

---

# A program for the Continuum Hypothesis

Jean-Louis Krivine

PPS Group, University Paris 7, CNRS

[krivine@pps.jussieu.fr](mailto:krivine@pps.jussieu.fr)

TLCA'09 Brasilia, July 2009

---

## Introduction

Classical realizability is a method to obtain programs from mathematical proofs in Analysis and set theory. In this talk, we limit ourselves to Analysis, a theory stated in second order logic.

Since the discovery, by T. Griffin in 1990, that a *control instruction* was typed with the law of Peirce, several frameworks have been given, to transform classical proofs in higher order logic into programs.

But higher order logic is not sufficient for Analysis ; we need some axioms : DC (dependent choice) is absolutely necessary.

AC, i.e. well ordering axioms would be very useful.

At least, weaker axioms, such as UF (the existence of ultrafilters on  $\mathbb{N}$ ).

Also CH (continuum hyp.) is a very interesting well ordering axiom.

---

## Introduction (cont.)

With **classical realizability**, we can associate programs not only with **proofs** but also with **axioms**.

In this talk, we shall deal with DC, UF and CH.

DC can be stated in second order order logic.

In order to state UF or CH, we need a slight extension of second order logic ;

i.e. a 3rd order predicate constant which represents :

a subset of  $\mathcal{P}(\mathbb{N})$  (ultrafilter)

a subset of  $\mathcal{P}(\mathbb{N})^2$  (well-ordering).

But the same method works for these axioms in higher order logic, or in ZF.

---

## Axioms for Analysis

For simplicity, we use second order logic instead of set theory.

Formulas are written with the only logical symbols  $\rightarrow, \forall$ .

We have *1st order* or *individual* variables and

*2nd order* or *predicate* variables of each arity.

The symbol  $\neq$  is a particular predicate constant, of arity 2.

There are three groups of axioms :

1. Equations such as  $\forall x(x + 0 = x), \forall x \forall y(x + sy = s(x + y)), \dots$

and inequations such as  $s0 \neq 0$ .

2. The comprehension axiom  $\exists X \forall \vec{x}(X\vec{x} \leftrightarrow F)$  for every formula  $F$ .

3. The axiom of dependent choice :

$$\forall X \exists Y F(X, Y) \rightarrow \exists Z \forall n \{ \text{int}(n) \rightarrow F(Z_n, Z_{n+1}) \}$$

for every formula  $F$  ( $X, Y$  are monadic,  $Z$  is dyadic).

---

## Axioms for Analysis (cont.)

It is also interesting to state axioms that *we don't want* because they are *always* false in non trivial realizability models.

- The extensionality axiom  $\forall X \forall Y [\forall x (Xx \leftrightarrow Yx) \rightarrow X = Y]$
- The recurrence axiom  $\forall x \text{int}(x)$

which says that each individual (1st order object) is an integer.

The formula for integers is  $\text{int}(x) \equiv \forall X \{ \forall y (Xy \rightarrow Xsy), X0 \rightarrow Xx \}$ .

Nevertheless, we get arithmetic by restricting formulas to integers.

Analysis is sufficient to formalize a very important part of mathematics including the theory of functions of real or complex variables, measure and probability theory, partial differential equations, analytic number theory, Fourier analysis, etc.

---

## Ultrafilter axiom

$\mathbf{U}$  is a 3rd order monadic predicate constant ; its argument is a unary predicate.

We state that  $\mathbf{U}$  is a non trivial ultrafilter on  $\mathbb{N}$  :

$$\forall X \forall Y (X \subset Y, \mathbf{U}(X) \rightarrow \mathbf{U}(Y)) ;$$

$$\forall X \forall Y (\mathbf{U}(X), \mathbf{U}(Y) \rightarrow \mathbf{U}(X \wedge Y)) ;$$

$$\forall X (\mathbf{U}(\mathbb{N} \setminus X) \leftrightarrow \neg \mathbf{U}(X)) ; \forall x \neg \mathbf{U}(\{x\}).$$

$$X \subset Y \text{ is } \forall n (\text{int}(n), Xn \rightarrow Yn).$$

The existence of ultrafilters greatly simplify some proofs in arithmetic or analysis. Ramsey theorem is a typical example.

Well orderings of  $\mathcal{P}(\mathbb{N})$  are very useful in analysis ;

for example, Hahn-Banach theorem ; or in order to apply Gelfand's theorem :

$\mathbb{C}$  is the only commutative Banach algebra which is a field.

---

## Continuum hypothesis

**D** and **W** are 3rd order monadic and dyadic predicate constants ;  
their arguments are unary predicates.

We first state that each subset of  $\mathbb{N}$  is extensionally equivalent  
to some element of **D** :  $\forall X \exists Y (\mathbf{D}(Y) \wedge X \simeq Y)$

$X \simeq X'$  is  $\forall n (\text{int}(n) \rightarrow (Xn \leftrightarrow X'n))$ .

Then, we state that **W** is a well ordering of domain **D** :

$\forall X \forall Y (\mathbf{W}(X, Y) \rightarrow \mathbf{D}(X) \wedge \mathbf{D}(Y))$  ;

$\forall X \forall Y (\mathbf{D}(X), \mathbf{D}(Y) \rightarrow \mathbf{W}(X, Y) \vee \mathbf{W}(Y, X) \vee X = Y)$  ;

$\forall Y \{ \forall X (\mathbf{W}(X, Y) \rightarrow F(X)) \rightarrow F(Y) \} \rightarrow \forall Y F(Y)$  for every formula  $F$  ;

and finally, that each initial segment is countable :

$\forall Y \exists Z \{ \forall X (\mathbf{W}(X, Y) \rightarrow \exists n (\text{int}(n) \wedge X \simeq Z_n)) \}$  ( $Z$  is binary).

---

## Classical realizability

$\Lambda$  is the set of *closed  $\lambda$ -terms*, with some constants :

$cc, \sigma, \chi, \chi'$  and  $k_\pi$  (*continuations*) where  $\pi$  is any stack.

A *stack* is a finite sequence  $\pi = \xi_1 \cdot \dots \cdot \xi_n \cdot \pi_0$  with  $\xi_i \in \Lambda$  ;

$\pi_0$  is taken from a set of *stack constants* which mark the bottom of stacks.

If  $\xi \in \Lambda$ , then  $\xi \cdot \pi$  is the stack  $\xi \cdot \xi_1 \cdot \dots \cdot \xi_n \cdot \pi_0$

and  $\pi^\xi$  is the stack  $\xi_1 \cdot \dots \cdot \xi_n \cdot \xi \cdot \pi_0$ .

The set of stacks is denoted by  $\Pi$ .

A *process* is a couple  $(\xi, \pi) \in \Lambda \times \Pi$ . It is written as  $\xi \star \pi$ .

We fix also a recursive bijection  $\pi \mapsto n_\pi$  of  $\Pi$  onto  $\mathbb{N}$ .

---

## Classical realizability models

Processes can be performed, which is denoted by  $\succ$ . Here are the rules :

- $\xi\eta \star \pi \succ \xi \star \eta.\pi$  (push)
- $\lambda x t \star \xi.\pi \succ t[\xi/x] \star \pi$  (pop)
- $cc \star \xi.\pi \succ \xi \star k_\pi.\pi$  (save the stack)
- $k_\pi \star \xi.\pi' \succ \xi \star \pi$  (restore the stack)
- $\sigma \star \xi.\pi \succ \xi \star \underline{n}_\pi.\pi$  (signature)
- $\chi \star \xi.\pi^\tau \succ \xi \star \tau.\pi$  (read)
- $\chi' \star \xi.\tau.\pi \succ \xi \star \pi^\tau$  (write).

The variable part of a *classical realizability model* is a set  $\perp\!\!\!\perp$  of processes which is *saturated*, which means that :

$$\xi \star \pi \in \perp\!\!\!\perp, \xi' \star \pi' \succ \xi \star \pi \Rightarrow \xi' \star \pi' \in \perp\!\!\!\perp.$$

---

## Classical realizability models (cont.)

Now, we define models in which the truth value set is  $\mathcal{P}(\Pi)$  instead of  $\{0, 1\}$ .

We define two sets  $\|\Phi\| \subset \Pi$  and  $|\Phi| \subset \Lambda$ .

$\xi \in |\Phi|$  is also written  $\xi \Vdash \Phi$  (read  $\xi$  realizes  $\Phi$ ).

These two sets are connected as follows :  $\xi \Vdash \Phi \Leftrightarrow (\forall \pi \in \|\Phi\|) \xi \star \pi \in \perp$ .

$\Phi$  is a closed second order formula *with parameters*.

2nd order parameters of arity  $k$  are functions  $\mathcal{X} : \mathbb{N}^k \rightarrow \mathcal{P}(\Pi)$ .

Induction steps to define  $\|\Phi\|$  :

Atomic formulas :  $\mathcal{X}(n_1, \dots, n_k), a \neq b, \mathcal{X} \neq \mathcal{Y}$ . Their truth value is obvious.

( $\|a \neq b\|$  and  $\|\mathcal{X} \neq \mathcal{Y}\|$  are either  $\top = \emptyset$  or  $\perp = \Pi$ ).

$\|\Phi \rightarrow \Psi\| = \{\xi \bullet \pi; \xi \Vdash \Phi, \pi \in \|\Psi\|\}$ .

$\|\forall x \Phi(x)\| = \bigcup \{\|\Phi(a)\|; a \in \mathbb{N}\}$

$\|\forall X \Phi(X)\| = \bigcup \{\|\Phi[\mathcal{X}/X]\|; \mathcal{X} : \mathbb{N}^k \rightarrow \mathcal{P}(\Pi)\}$  ( $X$  is a predicate variable of arity  $k$ ).

---

## Classical realizability models (cont.)

We see that realizability theory is exactly model theory, in which the truth value set is  $\mathcal{P}(\Pi)$  instead of  $\{0, 1\}$ .

We are indeed considering “standard” second order models :

the domain of individuals is  $\mathbb{N}$

the domain for  $k$ -ary predicate variables is  $\mathcal{P}(\Pi)^{\mathbb{N}^k}$  (instead of  $\{0, 1\}^{\mathbb{N}^k}$ ).

For each function  $f : \mathbb{N}^k \rightarrow \mathbb{N}$ , we have the  $k$ -ary function symbol  $f$  with its natural interpretation.

The truth values  $\emptyset$  and  $\Pi$  are denoted by  $\top$  and  $\perp$ . Therefore :

$t \Vdash \top$  for every term  $t$  ;  $t \Vdash \perp \Rightarrow t \Vdash F$  for every  $F$ .

**Warning.** In our *realizability* models, the set of individuals is  $\mathbb{N}$ .

But, the *2-valued* models we get from them are *non-standard*, i.e.

they may contain *non-standard integers* and even

individuals which are *not integers at all*.

---

---

## The adequation lemma

Realizability is compatible with intuitionistic deduction. We have :

### **Adequation lemma.**

*If  $\vdash t : \Phi$  in intuitionistic predicate logic, then  $t \Vdash \Phi$  for every choice of  $\perp$ .*

The proof is a simple induction on the length of the derivation of  $\vdash t : \Phi$ .

We extend this lemma to mathematical proofs, by adding new axioms.

If we want to use some axiom  $\mathcal{A}$ , we simply have to find

a term  $t$  and a formula  $\mathcal{B}$  such that  $t \Vdash \mathcal{B}$  and  $\mathcal{B} \vdash \mathcal{A}$ .

The only condition on  $t$  is to be a **proof-like term** i.e. *without continuation*.

In this way, we get immediately :

*Classical logic* because  $\mathbf{cc} \Vdash ((F \rightarrow G) \rightarrow F) \rightarrow F$  (law of Peirce).

*2nd order logic* because  $\lambda x(x)II \Vdash \exists X \forall \vec{x}(X\vec{x} \leftrightarrow F)$  (comprehension axioms).

*All true equations or inequations* because they are realized by  $I = \lambda x x$ .

---

## Dependent choice scheme

DC is more difficult to realize. In fact, we realize a stronger axiom scheme which we call **NEAC** (non extensional axiom of choice) :

$$(NEAC_F) \quad \forall X (F[X, \phi_F(X)] \rightarrow \forall Y F[X, Y])$$

$\phi_F : \mathcal{P}(\Pi)^{\mathbb{N}^m} \rightarrow \mathcal{P}(\Pi)^{\mathbb{N}^n}$  is a function which depends on the formula  $F$ .

NEAC is (much) weaker than AC because  $\phi_F$  is not supposed to be compatible with extensional equivalence.

The method is to show that  $\sigma \Vdash NEAC'_F$  where

$$NEAC'_F \equiv \forall X (\forall n (\text{int}(n) \rightarrow F[X, \psi_F(X, n)]) \rightarrow \forall Y F[X, Y])$$

and  $\psi_F : \mathcal{P}(\Pi)^{\mathbb{N}^m} \times \mathbb{N} \rightarrow \mathcal{P}(\Pi)^{\mathbb{N}^n}$ .

Then, we can prove easily in *classical* 2nd order logic that  $NEAC'_F \vdash NEAC_F$ .

---

## Usefulness of this stuff

- To get information on the programs associated with mathematical proofs.

By the adequation lemma, a proof of a formula  $F$  in Analysis

gives a term  $\theta$  such that  $\theta \Vdash F$  for every choice of  $\perp$ .

By suitable choices of  $\perp$ , we obtain important properties of  $\theta$  (see [3,4]).

Trivial example :  $F \equiv \exists n(\text{int}(n) \wedge R(n))$  with  $R$  recursive.

- To get programs for *axioms*.

In this talk, we do this for ultrafilter and well ordering axioms.

---

## UF, CH : first approach

At first sight, the problem for axioms such that AC or CH seems rather easy, using known set theoretic results about conservative extensions, which are valid for higher order logic or ZF :

Let  $F$  be a formula *in Analysis*. Any proof of  $F$ , which uses AC and GCH, can be transformed into a proof using only DC.

This new proof gives a program (proof-like term)  $\xi \Vdash F$ , since the axiom DC is already realized.

But this program was extracted from a proof of  $F$  *which is very far from the original one*.

In this way, we cannot get a program for AC or CH.

---

## CH : statement of the problem

The real problem is the following (we take CH as an example) :

Given a formula  $F$  in Analysis,

find a syntactic transformation  $\xi \mapsto \xi^*$  on  $\lambda$ -terms such that

if  $\vdash \theta : \text{CH} \rightarrow F$  is a proof of  $F$  in higher order logic with CH,

then  $\theta^* \Vdash F$  for every choice of  $\perp$ .

The important fact is that the transformation operates, not on the proof itself, but on the term  $\theta$ , which contains much less information.

---

## Statement of the problem (cont.)

Solving this problem gives us, as a by-product, a program  $\theta_{\text{CH}}$  for the axiom CH.

The real challenge is to **understand the behaviour of this program**.

In this talk, I simply give rough ideas on how to write it.

The conservative extension theorem stated above is proved by *set theoretical forcing*.

We formalise this proof in higher order logic with DC, and consider the term  $\theta$  obtained in this way.

The essential tool is the notion of *realizability structure* which contains as particular cases *classical realizability* and *sets of forcing conditions*.

---

## Realizability structures

A realizability structure  $\mathcal{S}$  is given as three sets  $\Lambda, \Pi, \Lambda \star \Pi$  (terms, stacks and processes), with the following operations :

- An application  $(\xi, \eta) \mapsto (\xi)\eta$  from  $\Lambda \times \Lambda$  into  $\Lambda$  (*application*).
- An application  $(\xi, \pi) \mapsto \xi \cdot \pi$  from  $\Lambda \times \Pi$  into  $\Pi$  (*push*).
- An application  $(\xi, \pi) \mapsto \xi \star \pi$  from  $\Lambda \times \Pi$  into  $\Lambda \star \Pi$  (*process*).
- An application  $\pi \mapsto \mathbf{k}_\pi$  from  $\Pi$  into  $\Lambda$  (*continuation*).

We have three distinguished terms  $\mathbf{K}, \mathbf{S}, \mathbf{cc} \in \Lambda$ .

There is also a subset  $\perp$  of  $\Lambda \star \Pi$  such that :

$$\xi\eta \star \pi \notin \perp \Rightarrow \xi \star \eta \cdot \pi \notin \perp ;$$

$$\mathbf{K} \star \xi \cdot \eta \cdot \pi \notin \perp \Rightarrow \xi \star \pi \notin \perp ;$$

$$\mathbf{S} \star \xi \cdot \eta \cdot \zeta \cdot \pi \notin \perp \Rightarrow \xi \star \zeta \cdot \eta \zeta \cdot \pi \notin \perp ;$$

$$\mathbf{cc} \star \xi \cdot \pi \notin \perp \Rightarrow \xi \star \mathbf{k}_\pi \cdot \pi \notin \perp ;$$

$$\mathbf{k}_\pi \star \xi \cdot \pi' \notin \perp \Rightarrow \xi \star \pi \notin \perp \text{ for every } \pi' \in \Pi.$$

---

## Example : Forcing

An obvious example is the classical realizability defined above.

We consider now a (slight) generalization of forcing :

The *set of conditions*  $P$  has a binary operation  $(p, q) \mapsto pq$  and a distinguished element  $\mathbf{1}$ .

A predicate  $C[p]$  defines the set of *non trivial conditions*.

We define a preorder on  $P$  :  $p \leq q \Leftrightarrow \forall r (C[pr] \rightarrow C[qr])$  ;

and an equivalence relation :  $p \simeq q \Leftrightarrow \forall r (C[pr] \leftrightarrow C[qr])$ .

We suppose the following formulas are true :

i)  $C[p(qr)] \leftrightarrow C[(pq)r]$  ;  $C[p] \leftrightarrow C[p\mathbf{1}]$ .

ii)  $C[(p(qr))s] \leftrightarrow C[((pq)r)s]$  ;  $C[(pq)r] \leftrightarrow C[(qp)r]$  ;

$C[pq] \rightarrow C[(pp)q]$  ;  $C[pq] \leftrightarrow C[(p\mathbf{1})q]$ .

iii)  $C[(pq)r] \rightarrow C[pr]$ .

---

## Forcing (cont.)

Then  $(P, \leq)$  is a semi-lattice with greatest element  $\mathbf{1}$  and  $\inf(p, q) = pq$ .

$C$  is an final segment of  $P$ , i.e. :  $p \leq q, C[p] \rightarrow C[q]$ .

The associated realizability structure is defined as follows :

$$\Lambda = \Pi = \Lambda \star \Pi = P ; \mathbf{K} = \mathbf{S} = \mathbf{cc} = \mathbf{1} ;$$

$$\mathbf{k}_p = p ; p \cdot q = p \star q = pq.$$

$\perp$  is defined as the complement of  $C$  :

$$p \notin \perp \Leftrightarrow C[p].$$

---

## General realizability models

Exactly as in the particular case of classical realizability, we can define the realisability model associated with a given realizability structure. Its truth value set is  $\mathcal{P}(\mathbf{\Pi})$ . We have again :

### **Adequation lemma.**

*If  $\vdash t : \Phi$  in classical higher order logic, then  $\mathbf{t} \Vdash \Phi$ .*

$t$  is a proof-like term, written with the combinators  $K, S, cc$  ;

$\mathbf{t} \in \mathbf{\Lambda}$  is its obvious translation with **K, S, cc**.

We have given so far two examples of realizability structures :  
**classical realizability** and **forcing**.

We need now a third one, which combines both.

---

## Product realizability structure

We consider a classical realizability structure  $\mathcal{S}$  and a predicate  $C[p]$  where  $p$  is a unary predicate variable.

We take  $P = \mathcal{P}(\Pi)^{\mathbb{N}}$  (the domain of  $p$ ).

The operation  $(p, q) \mapsto pq$  is some given binary function on  $P$ , and  $\mathbf{1}$  is fixed in  $P$ .

We assume all forcing axioms are realized by proof-like terms  $\alpha_i$ . Thus

$\alpha_0 \Vdash C[(pq)r] \rightarrow C[p(qr)]$ ,  $\alpha_1 \Vdash C[p(qr)] \rightarrow C[(pq)r]$  etc.

We define a realizability structure  $\mathcal{S}^+$  by putting :

$\Lambda = \Lambda \times P$  ;  $\Pi = \Pi \times P$  ;  $\Lambda \star \Pi = (\Lambda \star \Pi) \times P$ .

The operations  $\cdot$  and  $\star$  are straightforward :

$(\xi, p) \cdot (\pi, q) = (\xi \cdot \pi, pq)$  ;  $(\xi, p) \star (\pi, q) = (\xi \star \pi, pq)$ .

---

## Product realizability structure (cont.)

The operation on terms and the definition of constants are less obvious.

Let us first explain what we need :

$$\mathbf{K} = (\mathbf{K}^*, \mathbf{1}) ; \mathbf{S} = (\mathbf{S}^*, \mathbf{1}) ; \mathbf{cc} = (\mathbf{cc}^*, \mathbf{1}) ;$$

$$(\xi, p)(\eta, q) = ((\xi\eta)^*, pq) ; \mathbf{k}_{(\pi, p)} = (\mathbf{k}_\pi^*, p).$$

$\mathbf{K}^*, \mathbf{S}^*, \mathbf{cc}^*$  must be proof-like terms, with the following reduction rules :

$$\mathbf{K}^* \star \xi \cdot \eta \cdot \pi^\tau > \xi \star \pi^{\alpha\tau} \quad \text{with } \alpha \Vdash \mathbf{C}[\mathbf{1}(p(qr))] \rightarrow \mathbf{C}[pr] ;$$

$$\mathbf{S}^* \star \xi \cdot \eta \cdot \zeta \cdot \pi^\tau > \xi \star \zeta \cdot \eta \zeta \cdot \pi^{\beta\tau} \quad \text{with } \beta \Vdash \mathbf{C}[\mathbf{1}(p(q(rs)))] \rightarrow \mathbf{C}[p(r(qr)s)].$$

$$\mathbf{cc}^* \star \xi \cdot \pi^\tau > \xi \star \mathbf{k}_\pi^* \cdot \pi^{\gamma\tau} \quad \text{with } \gamma \Vdash \mathbf{C}[p(qr)] \rightarrow \mathbf{C}[q(rr)].$$

$(\xi\eta)^*$  and  $\mathbf{k}_\pi^*$  must be terms, with the following reduction rules :

$$(\xi\eta)^* \star \pi^\tau > \xi \star \eta \cdot \pi^{\alpha_0\tau} \quad \text{with } \alpha_0 \Vdash \mathbf{C}[(pq)r] \rightarrow \mathbf{C}[p(qr)].$$

$$\mathbf{k}_\pi^* \star \xi \cdot \omega^\tau > \xi \star \pi^{\delta\tau} \quad \text{with } \delta \Vdash \mathbf{C}[p(qr)] \rightarrow \mathbf{C}[qp].$$

---

## Product realizability structure (cont.)

For each term  $\alpha \in \Lambda$ , we set  $\bar{\alpha} = \lambda x(\chi)\lambda y(\chi'x)(\alpha)y$ .

The behaviour of  $\bar{\alpha}$  is as follows :  $\bar{\alpha} \star \xi \cdot \pi^\tau > \xi \star \pi^{\alpha\tau}$ .

Then, we can define :

$K^* = \bar{\alpha}K ; S^* = \bar{\beta}S ; cc^* = \lambda x(\chi)\lambda y(cc)\lambda k((\chi'x)(\gamma)y)(k^*)k$ .

$(\xi\eta)^* = \bar{\alpha}_0\xi\eta ; k_\pi^* = k^*k_\pi$  with  $k^* = \lambda k\lambda x(\chi)\lambda y(k)(\chi'x)(\delta)y$ .

We see that the last element  $\tau$  of the stack is a *global memory* or a *heap*.

The new control instruction  $cc^*$  does not store it.

The new continuation is  $k_\pi^*$  ; it throws away the current stack *except its last element*, which is the current heap.

$\chi$  and  $\chi'$  are the *read* and *write* instructions for this heap.

---

## Product realizability structure (cont.)

Finally, we define  $\perp\!\!\!\perp^+ \subset \mathbf{\Pi}$  :

$$(\xi \star \pi, p) \in \perp\!\!\!\perp^+ \Leftrightarrow (\forall \tau \in \Lambda)(\tau \Vdash C[p] \Rightarrow \xi \star \pi^\tau \in \perp\!\!\!\perp).$$

Then it can be shown that we have just defined a realizability structure that we call a *product realizability structure*.

Let  $t$  be a combinator written with  $\mathbf{K}, \mathbf{S}, \mathbf{cc}$ .

If we substitute  $\mathbf{K}, \mathbf{S}, \mathbf{cc}$ , we get  $\mathbf{t} = (t^*, \mathbf{1}_t) \in \Lambda$  ;

$t^*$  is obtained from  $t$  by the rules above ;

$\mathbf{1}_t$  is a condition composed with  $\mathbf{1}$  and parenthesis, with the same tree structure as  $t$ .

The notation for  $\mathbf{t}$  realizes  $F$  in this realizability structure will be  $\mathbf{t} \Vdash^+ F$

---

## A term for CH (sketch)

We use now a well known result about forcing :

With a suitable ordering on the set  $\mathcal{P}(\mathbb{N})$ ,

we can prove in higher order logic with DC that  $\mathbf{1} \Vdash \text{CH}$ .

Moreover,  $\mathbf{1} \Vdash (\mathcal{P}(\mathbb{N}) \text{ is not changed})$ .

If we formalize correctly this proof, we get

a forcing realizability structure on  $\mathcal{P}(\Pi)^{\mathbb{N}}$  (the domain of unary predicates)

and a proof-like term  $\theta_{\text{CH}}$  such that  $\theta_{\text{CH}} \Vdash \mathbf{1} \Vdash \text{CH}$ .

The first symbol  $\Vdash$  is for *classical realizability*,

the second symbol  $\Vdash$  is *forcing*.

---

## A term for CH (sketch)

But it can be shown that classical realizability followed by forcing is somehow equivalent to product realisability. Precisely, we have :

$$\xi \Vdash p \Vdash F \Rightarrow (\chi_F \xi, p) \Vdash^+ F ;$$

$$(\xi, p) \Vdash^+ F \Rightarrow \chi'_F \xi \Vdash p \Vdash F$$

where  $\chi_F$  and  $\chi'_F$  are proof-like terms, which depends only on the arrow structure of the formula  $F$ .

Thus, we have shown that  $((\chi_{CH})\theta_{CH}, \mathbf{1}) \Vdash^+ CH$ .

---

## A term for CH (sketch)

Consider now a formula  $F$  in Analysis and a proof  $\vdash t : \text{CH} \rightarrow F$ .

By the adequation lemma, we get  $\mathbf{t} \Vdash^+ \text{CH} \rightarrow F$

i.e.  $(t^*, \mathbf{1}_t) \Vdash^+ \text{CH} \rightarrow F$ . Therefore

$(t^*, \mathbf{1}_t)((\chi_{\text{CH}})\theta_{\text{CH}}, \mathbf{1}) \Vdash^+ F$  that is

$((\bar{\alpha}_0 t^*)(\chi_{\text{CH}})\theta_{\text{CH}}, \mathbf{1}_t \mathbf{1}) \Vdash^+ F$  and by product realizability

$(\chi'_F)(\bar{\alpha}_0 t^*)(\chi_{\text{CH}})\theta_{\text{CH}} \Vdash \mathbf{1}_t \mathbf{1} \Vdash F$ .

But, because  $F$  is a formula *in Analysis*,

and because *the forcing does not change*  $\mathcal{P}(\mathbb{N})$

$\mathbf{1}_t \mathbf{1} \Vdash F$  is  $\mathbf{C}[\mathbf{1}_t \mathbf{1}] \rightarrow F$ .

Now  $\mathbf{1}_t \mathbf{1}$  is a product of  $\mathbf{1}$ ,

thus there is a proof-like term  $\tau_0$  such that  $\vdash \tau_0 : \mathbf{C}[\mathbf{1}_t \mathbf{1}]$ .

---

## A term for CH (sketch)

Finally, we get the following proof-like term which realizes  $F$  :

$$((\chi'_F)(\bar{\alpha}_0 t^*)(\chi_{CH})\theta_{CH})\tau_0 \Vdash F.$$

For example, when  $F$  is an arithmetical formula, this term implements a winning strategy in the associated game.

The term  $\lambda x((\chi'_F)(\bar{\alpha}_0 t^*)(x)\tau_0)$  is obtained by a syntactic transformation on the original term  $t$ .

The term  $(\chi_{CH})\theta_{CH}$  is the program associated with *the continuum hypothesis*.

---

## Conclusion

The term  $(\chi_{CH})\theta_{CH}$  is the program associated with *the continuum hypothesis*.

The real challenge is now to understand the behaviour of such programs.

Up to now, I have no idea for the continuum hypothesis.

For the ultrafilter axiom, there are some good reasons to believe that the program behaves as a scheduler in an operating system.

---

## References

1. **T. Griffin** *A formulæ-as-type notion of control.*

Conf. Record of the 17th A.C.M. Symp. on Principles of Progr. Languages, 1990.

2. **J.-L. Krivine** *Typed lambda-calculus in classical Zermelo-Fraenkel set theory.*

Arch. Math. Log. 40, 3, 189-205, 2001.

3. **J.-L. Krivine** *Dependent choice, 'quote' and the clock.*

Th. Comp. Sc. 308, 259-276, 2003.

4. **J.-L. Krivine** *Realizability in classical logic.*

To appear in Panoramas et Synthèses. Société mathématique de France.

5. **J.-L. Krivine** *Structures de réalisabilité, RAM et ultrafiltre sur  $\mathbb{N}$ .*

To appear.

Pdf files at <http://www.pps.jussieu.fr/~krivine>