Forcing and Type Theory

Thierry Coquand

Oslo, June 11, 2009

Krivine's program

Starting point of this work: Krivine's program

To understand the computational meaning of mathematical proofs

Intuitionistic logic: reduction machine (Krivine abstract machine) with a term, an environment and a stack

Excluded-middle: one adds some new instructions, call-cc

Dependent choice: one adds a new instruction (a clock)

Krivine's program

Krivine has found a computational interpretation of principles such as

well-ordering of the reals

non principal ultrafilters over the natural numbers

continuum hypothesis

general axiom of choice

Furthermore the added instructions are remarkably simple (accessing and adding some value at the bottom of the stack)

Krivine's program

How does this work?

Krivine uses *forcing interpretation*

With forcing, we can explain

- well-ordering of the reals
- non principal ultrafilters over the natural numbers
- continuum hypothesis

using classical logic and dependent choice

This talk

We present a possible interpretation of non principal ultrafilters using dependent choice and excluded middle

No clear idea of the computational meaning

We take a simpler example: addition of one generic real Cohen to type theory

Clear computational interpretation and one mathematical application: definable functionals are uniformely continuous

Iterated forcing: we can interpret a forall functional

Reformulation of forcing

A.M. Levin "One conservative extension of formal mathematical analysis with a scheme of dependent choice" (1977)

Forcing over the system $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC}$ (for well-ordering of the reals)

Theorem: *If* $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC} + \mathsf{SUF} \vdash A$ *then* $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC} \vdash A$

The terms of the language are simply typed lambda terms. We have two basic types N (natural numbers) and N_2 (booleans). The atomic formulae are simply the terms of type N_2. There are two terms $0, 1$ of type N_2 and we identify 1 with the true formula $\top$ and 0 with the false formula $\bot$.

Reformulation of forcing

The formulae are

$$\varphi \ ::= \ \varphi \rightarrow \varphi \mid t \mid \forall x.\varphi$$

where t is a term of type N_2 (decidable atomic formula)

We use $n, m, \dots$ for variables over the type N. Example: $\forall n.\exists^c m.n < m$.

$\neg\varphi$ to be $\varphi \rightarrow \perp$

$\exists^c x.\varphi$ is $\neg\forall x.\neg\varphi$

Reformulation of forcing

The system HA^ω is intuitionistic with the usual rules of natural deduction and induction over natural numbers and boolean. The rule EM is $(\neg\neg\varphi) \rightarrow \varphi$ which is equivalent to $\varphi \vee \neg\varphi$. The rule DC is

$$\forall n.\forall x.\exists y.\varphi(n, x, y) \rightarrow \forall u.\exists f.\varphi(0, u, f(0)) \wedge \forall n.\varphi(n, f(n), f(n + 1))$$

The rule CC is

$$\forall n.\exists y.\varphi(n, y) \rightarrow \exists f.\forall n.\varphi(n, f(n))$$

Forcing

We add a new symbol μ and new atomic formula $\mu(f)$ for f of type $N \rightarrow N_2$

We consider now the extension of the theory HA^ω with the axioms (we could add the selectivity axiom)

$$\mu(1) \qquad \mu(fg) \leftrightarrow (\mu(f) \wedge \mu(g))$$

$$\mu(f) \vee^c \mu(1-f) \qquad \mu(f) \rightarrow \forall m.\exists^c n > m.f(m)$$

Forcing

We use letters $p, q, r, \dots$ to denote *forcing conditions*, here simply terms of type $N \rightarrow N_2$. One can think of forcing conditions as *decidable subsets* of $\mathbb{N}$.

We define a formula $p \Vdash \varphi$ by induction on φ where φ is an extended formula (which may contain the new symbol μ) and p is of type $N \rightarrow N_2$.

$I(p)$ is $\forall n. \exists m > n.\ p(m)$ $\qquad$ $F(p)$ is $\exists n. \forall m > n.\ \neg p(m)$

$\mu(f) \rightarrow I(f)$

$p \leqslant q$ is $F(p(1-q))$

Forcing

$p \Vdash \mu(f)$ is $p \leqslant f$

$p \Vdash \varphi$ is $I(p) \rightarrow \varphi$ if φ is a boolean

$p \Vdash \varphi_0 \rightarrow \varphi_1$ is $\forall q \leqslant p.(q \Vdash \varphi_0) \rightarrow (q \Vdash \varphi_1)$

$p \Vdash \forall x.\varphi$ is $\forall x.(p \Vdash \varphi)$

We can add other connectives and existential quantification

Not needed if we are only interested in classical logic

Forcing

Proposition: *If* $\varphi_1, \ldots, \varphi_n \vdash \varphi$ *and* $p \Vdash \varphi_1, \ldots, p \Vdash \varphi_n$ *then* $p \Vdash \varphi$

Using EM

Proposition: *We have* $p \Vdash \varphi_0 \vee^c \varphi_1$ *iff*

$$\forall q \leqslant p.\exists r \leqslant r.\ (r \Vdash \varphi_0) \vee^c (r \Vdash \varphi_1)$$

and $p \Vdash \exists^c x.\varphi$ *iff*

$$\forall q \leqslant p.\exists r \leqslant r.\exists^c x.\ r \Vdash \varphi$$

Forcing

Proposition: *We have (classical version of the comprehension axiom)*

$$p \Vdash (\forall n.\varphi(n,0) \vee^c \varphi(n,1)) \rightarrow \exists^c f.\forall n \varphi(n, f(n))$$

This expresses that there are no more decidable functions in the extension than in the ground model

Proposition: *We have (countable choice)*

$$p \Vdash (\forall n.\exists^c x.\varphi(n,x) \rightarrow \exists^c f.\forall n \varphi(n, f(n))$$

Forcing

All the axioms of non principal ultrafilters are forced

We have $\mathsf{HA}^\omega \vdash (p \to \varphi) \leftrightarrow (p \Vdash \varphi)$ if φ does not mention μ

$\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC} + \mathsf{SUF} \vdash \varphi$ implies $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC} \vdash \ (\Vdash \varphi)$ and hence $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{DC} \vdash \varphi$

So we have a computational interpretation of non principal ultrafilters

Levin (1977) does the same with a well-ordering of the reals, which justifies also the continuum hypothesis

Forcing and Type Theory

Difficult to understand the computational interpretation

Simpler framework: topological model (Beth semantics) with intuitionistic logic only

Main principle: we do not interpret the system by induction on types, but we do a direct "global" interpretation using the fact that type theory is essentially algebraic

Type Theory

Alternative to set theory for constructive mathematics

Identification of types and propositions, elements and proofs

Total functional programming language with dependent types

We present a mathematical application of forcing: any definable functional of type $(N \to N_2) \to N_2$ is uniformely continuous

A new way to program the universal quantification on $N \to N_2$

Type Theory

$$t \ ::= \ x \mid t\ t \mid \lambda x.t \mid h \mid c$$

$$A, B \ ::= \ (\Pi x : A)B \mid U \mid N \mid N_2 \mid Ord$$

$A \rightarrow B$ for $(\Pi x : A)B$ if x not free in B

$t_1\ t_2$ for $t_1(t_2)$ $\qquad$ $t_1\ t_2\ t_3$ for $(t_1\ t_2)\ t_3$

Type Theory

Intensional type theory

Judgements $\vdash t : A \qquad \vdash t_1 = t_2 : A \qquad \vdash A \qquad \vdash A_1 = A_2$

data types $N,\ N_2,\ Ord$

associated constructors $0 : N,\ S\ x : N\ [x : N],\ 0 : N_2,\ 1 : N_2$

Hypothetical judgement $\Gamma \vdash J$

$\Gamma, \Delta, \ldots$ *context* of the form $x_1 : A_1, \ldots, x_n : A_n$

Type Theory

$$\frac{\Gamma, x : A \vdash B}{\Gamma \vdash (\Pi x : A)B} \qquad \frac{\Gamma \vdash A : U}{\Gamma \vdash A} \qquad \frac{}{\Gamma \vdash U}$$

$$\frac{}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : (\Pi x : A)B} \qquad \frac{\Gamma \vdash t : (\Pi x : A)B \quad \Gamma \vdash a : A}{\Gamma \vdash t\ a : B(x/a)}$$

$$\frac{\Gamma \vdash A : U \quad \Gamma, x : A \vdash B : U}{\Gamma \vdash (\Pi x : A)B : U}$$

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

Type Theory

If we have $c : C(0)$ and $g : (\Pi x : N)\ C(x) \rightarrow C(S(x))$

we can introduce a function $h : (\Pi x : N)C(x)$

with computation rules $h\ 0 = c \qquad\qquad h\ (S\ x) = g\ x\ (h\ x))$

Thinking of $C(x)$ as a proposition h is a proof of the universal proposition $(\Pi x : N)C(x)$ which we get by applying the principle of *mathematical induction*

In the case $C(x)$ does not depend explicitely on x we get the schema of primitive recursion (at higher types), schema introduced by Hilbert and used later by Gödel

Type Theory

We can introduce the type Ord, the type of *ordinal numbers*.

$0 : Ord,\ S\ x : Ord\ [x : N],\ L\ u : Ord\ [u : N \rightarrow Ord]$

The elimination rule expresses both the principle of *transfinite induction* over the second number class ordinals and definition of objects by transfinite recursion

Type Theory

In the formal theory the abstract entities (natural numbers, ordinals, functions, types, and so on) become represented by certain symbol configurations, called terms, *and the definitional schema, read from the left to the right, become mechanical* reduction rules *for these symbol configurations.*

Type theory *effectuates the computerization of abstract intuitionistic mathematics that above all Bishop has asked for*

It provides a framework in which we can express *conceptual* mathematics in a *computational* way.

Computability relation

$\varphi_A(t)$ "t is *computable* at type A" for $\vdash t : A$

$\varphi_N(t)$ iff $\vdash t = k : N$ for some numeral k

$\varphi_{N_2}(t)$ iff $\vdash t = b : N_2$ for some boolean b

$\varphi_{A \to B}(c)$ iff $\varphi_A(a)$ implies $\varphi_B(c\ a)$

Theorem: *If* $\vdash t : A$ *then* $\varphi_A(t)$

Computability relation

This can be defined for dependent type theory

One considers an inductive-recursive definition of

A computable type

and for A computable type, a predicate φ_A

For instance if A computable and $B(a)$ computable whenever $\varphi_A(a)$ then $(\Pi x : A)B(x)$ is computable and $\varphi_{(\Pi x:A)B(x)}(c)$ iff $\varphi_A(a)$ implies $\varphi_{B(a)}(c\ a)$

Type Theory

All well-typed terms are computable, hence normalisable

A programming language with *decidable* type-checking

Total functional programming language (D. Turner)

This implements the initial model (term model, free model, syntactical model) of type theory

Cartmell: generalised algebraic theory, (almost) equational presentation of type theory (category with families, P. Dybjer)

Forcing

First example in set theory: Cohen real where one adjoins to a model of set theory M a “generic” set of integers f

This model $M(f)$ negates the axiom of constructibility: the function f is “lawless”

Forcing

The model is constructed by transfinite induction on ordinals

For instance one define *names* by transfinite induction

$$N_\alpha = \bigcup_{\beta<\alpha} \mathcal{P}(N_\beta \times \mathsf{P})$$

where P is the set of conditions

Forcing

Boolean-valued model

$V_\alpha^{(B)} = \{x \mid Fun(x) \wedge Ran(x) \subseteq B \wedge \exists \beta < \alpha [Dom(x) \subseteq V_\beta^{(B)}]\}$

$V^{(B)} = \{x \mid \exists \alpha . [x \in V_\alpha^{(B)}]\}$

new collection of sets, also defined by transfinite induction

The elements in $V_\alpha = \cup_{\beta<\alpha} \mathcal{P}(V_\beta)$ are not in general elements of $V^{(B)}$

Cohen reals in type theory

Let us try to adjoin one Cohen real to the syntactical model of type theory

We try to adjoin a generic function: it will be represented by a new symbol $f : N \to N_2$

The *conditions* $p, q, \ldots$ (finite amount of informations about this generic function) are finite sets of compatible equations of the form

$$f(3) = 0,\ f(4) = 0,\ f(0) = 1,\ f(5) = 1,\ f(7) = 1$$

Write $q \leqslant p$ if q refines p

Cohen reals in type theory

A condition p defines a basic open X_p of Cantor space C

Inductive definition of *covering relation* $p \lhd U$ where U is a finite set of conditions $p_1, \ldots, p_n$

Basic covering: if n is not in the domain of p then

$$p, f(n) = 0 \qquad\qquad p, f(n) = 1$$

covers p

Cohen reals in type theory

We define new judgements $\Gamma \vdash_p J$ indexed by conditions

We shall have $\Gamma \vdash_{p_1} J$ if $\Gamma \vdash_p J$ and $p_1 \leqslant p$

The new rules are

$\Gamma \vdash_p f : N \rightarrow N_2$

$\Gamma \vdash_p f\ n = b : N_2$ if $f\ n = b$ is in p

Cohen reals in type theory

If p is covered by $p_1, \ldots, p_n$

$$\frac{\Gamma \vdash_{p_1} J \;\; \ldots \;\; \Gamma \vdash_{p_n} J}{\Gamma \vdash_p J}$$

for instance

$$n : N \vdash_p n \; = \; if \; (f \; 0) \; then \; n \; else \; n : N$$

Cohen reals in type theory

This rule is reminiscent of Beth models

However we have

$$\frac{\Gamma, x : A \vdash_p t : B}{\Gamma \vdash_p \lambda x.t : A \rightarrow B}$$

which *does not* correspond to the usual Beth semantics of implication

Cohen reals in type theory

Connection between the standard model and the forcing extension?

If $\Gamma \vdash J$ then we have $\Gamma \vdash_p J$ for any p

No transfinite recursion

Conversely, assume $\vdash g : N \rightarrow N_2$

Proposition: *If* $\Gamma \vdash_p J$ *and* g *satisfies the condition* p *then* $\Gamma(f/g) \vdash J(f/g)$

Corollary: *If* $\Gamma \vdash A$ *and* $\Gamma \vdash_p a : A$ *then there exists* a' *such that* $\Gamma \vdash a' : A$

Cohen reals in type theory

If p is covered by $p_1, \ldots, p_n$

$$\frac{\Gamma \vdash_{p_1} J \quad \ldots \quad \Gamma \vdash_{p_n} J}{\Gamma \vdash_p J}$$

and g satisfies p then g satisfies exactly one of the p_i

Computability relation

We define $p \Vdash \varphi_A(t)$ for $\vdash_p t : A$

$p \Vdash \varphi_N(t)$ iff there a covering $p_1, \ldots, p_n$ of p and numerals $k_1, \ldots, k_n$ such that

$$\vdash_{p_1} t = k_1 : N \ \ \ldots \ \ \vdash_{p_n} t = k_n : N$$

$p \Vdash \varphi_{N_2}(t)$ iff there a covering $p_1, \ldots, p_n$ of p and booleans $b_1, \ldots, b_n$ such that

$$\vdash_{p_1} t = b_1 : N_2 \ \ \ldots \ \ \vdash_{p_n} t = b_n : N_2$$

Computability relation

$p \Vdash \varphi_{A\rightarrow B}(t)$ iff for any $p_1 \leqslant p$ we have

$p_1 \Vdash \varphi_A(u)$ implies $p_1 \Vdash \varphi_B(t\ u)$

Beth/topological model

Computability relation

Lemma: *The generic function* $f : N \to N_2$ *is computable*

$\Vdash \varphi_{N \to N_2}(f)$

We have $p \Vdash \varphi_{N_2}(f\ k)$ for any condition p and numeral k

Theorem: *If* $\vdash_p t : A$ *then* $p \Vdash \varphi_A(t)$

Uniform continuity

Corollary: *If* $\vdash t : (N \rightarrow N_2) \rightarrow N_2$ *then there exists a finite formal covering* $p_1, \dots, p_n$ *of Cantor space and booleans* $b_1, \dots, b_n$ *such that* $\vdash_{p_i} t\ f = b_i : N_2$

Any definable functional is uniformely continuous

Proof: Since we have $\Vdash \varphi_{(N\rightarrow N_2)\rightarrow N_2}(t)$ and $\Vdash \varphi_{N\rightarrow N_2}(f)$ we also have $\Vdash \varphi_{N_2}(t\ f)$

If $\vdash g : N \rightarrow N_2$ is a standard function, it will satisfy exactly one condition p_i and then $\vdash t\ g = b_i : N_2$ by substitution of f by g

Evaluation of expressions

In standard type theory we have only one “process” (Krivine’s terminology) running

For forcing extensions of type theory the result of the evaluation of a term t at stage p will be a formal sum $\Sigma p_i t_i$ where $p_1, \ldots, p_n$ is a covering (partition) of p

Evaluation of expressions

The new rules are

$p\ (f\ n) = p\ b$ if $f(n) = b$ in p

$p\ (f\ n) = p_0\ 0 + p_1\ 1$ otherwise where p_i extends p with $f(n) = i$

Otherwise we have

$p\ ((\lambda x.t)\ u) = p\ t(x/u)$

Evaluation of expressions

In general we evaluate formal sums $\Sigma p_i\ t_i$ where p_i is a "partition of unity"

Several independent computations in parallel

Natural notion of equality, and the reduction is still Church-Rosser

Evaluation of expressions

If f does not appear in t then the evaluation proceeds as in standard type theory

We have an *extension* of standard type theory. We can apply any standard term $t : (\Pi g : N \to N_2)C(g)$ to the generic function f

Conversion and type-checking are still decidable

Evaluation of expressions

If $\vdash t : (N \rightarrow N_2) \rightarrow N_2$ we can decide if $\forall g.t\ g$ is true or not by computing

$t\ f = \Sigma p_i b_i$

Then $\forall g.t\ g$ is true iff $b_1 = \cdots = b_n = 1$

Evaluation of expressions

Thus we can evaluate

$\forall : ((N \rightarrow N_2) \rightarrow N_2) \rightarrow N_2$

when applied to standard expressions

Can we build a model where this functional is always evaluated?

Addition of infinitely many Cohen reals

Iterated forcing

We can add finitely many generic functions $f_1, \dots, f_n$

The conditions are now of the form

$f_1(3) = 0, f_1(4) = 1,\ f_2(0) = 1, f_2(3) = 1, f_2(4) = 0,\ f_3(5) = 1$

The conditions define now basic open X_p of C^n

Addition of infinitely many Cohen reals

$\vdash_{n,p} \forall\ t = 1 : N_2$ iff there is a covering $p_1, \ldots, p_l$ of p adding f_{n+1} such that

$$\vdash_{n+1,p_i} t\ f_{n+1} = 1$$

$\vdash_{n,p} \forall\ t = 0 : N_2$ iff there is a condition $q \leqslant p$ using f_{n+1} such that

$$\vdash_{n+1,q} t\ f_{n+1} = 0$$

Evaluation of expressions

Consider $\pi_1 : X_p \times C \to X_p$. If we evaluate

$p\ (t\ f_{n+1}) = \Sigma q_i c_i$

then $q_1, \dots, q_l$ is a covering of $X_p \times C$, which can be seen as a boolean valued continuous function $\psi : X_p \times C \to N_2$

We can find $p_1, \dots, p_n$ covering of X_p such that ψ depends only on its second component on $X_{p_i} \times C$ and then

$p\ (\forall\ t) = \Sigma p_j b_j$

Addition of infinitely many Cohen reals

In this way we get a computation of the functional

$\forall : ((N \rightarrow N_2) \rightarrow N_2) \rightarrow N_2$

Addition of infinitely many Cohen reals

In this model we use a "varying space"

$C \leftarrow C^2 \leftarrow C^3 \leftarrow \dots$

Each space C^n has a notion of covering

We do not need to consider the projection map $C^{n+1} \rightarrow C^n$ to be a covering

Extension of type theory

One *cannot* program quantification or the fan functional in standard type theory (R. Gandy, Howard)

It is possible to program these functionals with general recursion (W. Tait, R. Gandy, U. Berger, A. Simpson, M. Escardo) and the normalization of this program usually relies on the continuity property

Extension of type theory

We suggest here a *different* way to implement these functionals without relying on general recursion

These models, with one or infinitely many Cohen reals, still have the normalization property

Conversion and type-checking are still decidable

Probably, Cantor space $N \rightarrow N_2$ is spatial in this model

Algebraic numbers

Algebraic closure of a field: usual justification is done by transfinite induction/Zorn's Lemma

This is used for instance in the theory of algebraic curves: the theory is simpler/more uniform if one starts from a field which is algebraically closed

Algebraic numbers

The necessity of using an algebraically closed ground field introduced -and has perpetuated for 110 years- a fundamentally transcendental *construction at the foundation of the theory of* algebraic curves*. Kronecker's approach, which calls for adjoining new constants algebraically as they are needed, is much more consonant with the nature of the subject*

H. Edwards *Mathematical Ideas, Ideals, and Ideology*, Math. Intelligencer 14 (1992), no. 2, 6–19.

Algebraic numbers

Extension of type theory with a type K of *algebraic numbers*

Algebraic closure of the rationals

We want to add quantities with conditions

An algebraic extension of the rational field $\mathbb{Q}$ *is desribed by* adjunction relations *of the form* $\varphi_1(q_1) = 0,\ \varphi_2(q_1, q_2) = 0, \ldots$ *Adjunction of each* q_j *extends the field of "known" quantities*

The Kronecker-Duval Philosophy

Teo Mora's book *Solving Polynomial Equations*

Kronecker's model gives a powerful tool for computing with algebraic numbers provided *we have an algorithm for factorizing polynomials over a given algebraic extension of the rationals*

Such an algorithm exists but its practical complexity is so unsatisfactory that the solution provided by Kronecker's ideas has no practical impact.

In 1987 Duval added an unexpected twist to Kronecker's proposal, showing how factorization can be easily avoided. Her proposal threw light on Kronecker's ideas, clarifying the philosophy behind them.

Algebraic numbers

We have to build a model in which we can can realize

$$(\Pi\ u : K)\ [Id(u,0) + (\Sigma v : K) Id(uv,1)]$$

$$(\Pi\ u_1 \ldots\ u_l : K)(\Sigma\ v : K)\ Id(v^l + u_1 v^{l-1} + \cdots + u_l, 0)$$

Algebraic numbers

The conditions are now given by finitely many indeterminates $x_1, \ldots, x_n$ and finitely many polynomials conditions $P_1(x_1) = 0, \; P_2(x_1, x_2) = 0, \ldots$

Equational extension of type theory

For instance $x_1^2 - 2 = 0, \; x_2^2 - 2 = 0$

At each stage, the canonical value of a closed element of type K is a polynomial in $x_1, \ldots, x_n$

Algebraic numbers

We explain how to realize the field axiom

$$(\Pi\ u : K)\ [Id(u, 0) + (\Sigma v : K) Id(uv, 1)]$$

Algebraic numbers

We compute u: it evaluates to a polynomial $P(x_1)$

We can assume the defining condition $P_1(x_1) = 0$ to be square free

We compute the gcd of P with P_1: we find a spliting of $P_1 = P_{11}P_{12}$ with P_{11} divides P and P_{12} prime to P

Since P_{12} prime to P we find a relation $AP_{12} + BP = 1$ and $B(x_1)$ is an inverse of $P(x_1)$ for $P_{12}(x_1) = 0$

Since P_{11} divides P we have $P(x_1) = 0$ if $P_{11}(x_1) = 0$

Algebraic numbers

Example: with the condition p

$x_1^2 - 2 = 0,\ x_2^2 - 2 = 0$

is the element $u = x_1 - x_2$ invertible?

the system splits in two systems

$x_1^2 - 2 = 0,\ x_1 - x_2 = 0$ over which we have $u = 0$

$x_1^2 - 2 = 0,\ x_1 + x_2 = 0$ over which we have $uv = 1$ with $v = x_1/4$

Algebraic numbers

We need to realize also the axiom of algebraic closure

$$(\Pi\ u_1 \dots\ u_l : K)(\Sigma\ v : K)\ Id(v^l + u_1 v^{l-1} + \cdots + u_l, 0)$$

For this, we add a *new kind of covering*: any condition

$$P_1(x_1) = 0,\ \ \dots, P_n(x_1, \dots, x_n) = 0$$

is covered by the condition obtained by adding a *new* indeterminate x_{n+1} and a new polynomial condition

$$P_{n+1} = x_{n+1}^l + Q_1 x_{n+1}^{l-1} + \cdots + Q_l = 0$$

where $Q_1, \dots, Q_l$ are polynomials in $x_1, \dots, x_n$

Algebraic numbers

We add two (Skolem) functions

$$\alpha(u_1,\dots,u_l) : K \ [u_1 \ \dots \ u_l : K]$$

$$\beta(u_1,\dots,u_l) : Id(\alpha^l + u_1\alpha^{l-1} + \cdots + u_l, 0) \ [u_1 \ \dots \ u_l : K]$$

Reminiscent of Hilbert's ϵ-symbol

Algebraic numbers

They should behave as functions: $\alpha(u_1, \ldots, u_l) = \alpha(v_1, \ldots, v_l) : K$ whenever $u_1 = v_1 : K, \ldots, u_l = v_l : K$

For this we have to check if the equations

$$x^l + u_1 x^{l-1} + \cdots + u_l = 0 : K$$

has already received a solution in the condition, and if not one extends the condition with a new the indeterminate solution x of this equation

The computation rule is then $\alpha(u_1, \ldots, u_l) = x$

Algebraic numbers

Evaluation rule for $\alpha(u_1,\dots,u_l)$ at p

We evaluate $u_1,\dots,u_l$ (this may involve a spliting of the condition p)

if $x^l + u_1x^{l-1} + \cdots + u_l = 0 : K$ has received a solution x_m in p the value is x_m

otherwise we introduce a new indeterminate x_{n+1} with the equation

$x_{n+1}^l + u_1x_{n+1}^{l-1} + \cdots + u_l = 0$

and the value is x_{n+1}

Algebraic numbers

We have a computational interpretation of type theory extended with a type of algebraic numbers K

We still have the normalization property and decidability of type checking

We can use all the general results proved about decidable field in the *standard* theory and instantiate them on the field K

Algebraic numbers

In this model we have a canonical example of a *topological system* (G. Sambin)

$(X, \mathcal{A}, \models)$

X a set of *points*, here $X = K$

$\mathcal{A}$ is a formal topology, here the Zariski lattice of $K[x]$, of basic open $D(a)$ with a in $K[x]$

$u \models D(a)$ is defined to be $\neg Id(a(u), 0)$

Algebraic numbers

We then have a *spatial* or *extensional* topological system

$D(a_1) \wedge \cdots \wedge D(a_n) \leqslant D(b_1) \vee \cdots \vee D(b_m)$

iff

$(\Pi u : X)\ u \models D(a_1) \wedge \cdots \wedge u \models D(a_n) \rightarrow u \models D(b_1) \vee \cdots \vee u \models D(b_m)$

Algebraic numbers

This can be generalised to any algebraic curves (algebraic extension of $K(x)$): X is then the set of *places* of the curve and $\mathcal{A}$ the space of *valuations* associated to this curve

For instance for $\mathcal{A} = Val(K(x), K)$ the set of points is $X = K \cup \{\infty\}$

Generalizations

The same method can be applied for other theories in algebra

real algebraic closure, separable algebraic closure (Galois theory, we add *all* the roots of a polynomial at the same time), differential closure

Krivine's work *Structure de réalisabilité, RAM et ultrafiltre sur* N reduces non effective principles (non principal ultrafilters, well-ordering of the reals) to classical dependent choice. (See also previous work of A.M. Levin.) This gives a computational interpretation of such principles.

Connection with Goodman's combination of forcing + realizability