ECOLE NORMALE SUPERIEURE

Prototype proofs in type theory

Giuseppe LONGO

LIENS - 98 - 17

Département de Mathématiques et Informatique

CNRS URA 1327

Prototype proofs in type theory

Giuseppe LONGO

LIENS - 98 - 17

December 1998

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 01 44 32 30 00

Adresse électronique : longo@dmi.ens.fr

Prototype Proofs in Type Theory

Giuseppe Longo
LIENS(CNRS) and DMI
Ecole Normale Supérieure
45 rue d'Ulm
75005 Paris, France
http://www.dmi.ens.fr/users/longo

November 10, 1998

Abstract

The proofs of universally quantified statements, in mathematics, are given as "schemata" or as "prototypes" which may be applied to each specific instance of the quantified variable. Type Theory allows to turn into a rigorous notion this informal intuition described by many, including Herbrand. In this constructive approach where propositions are types, proofs are viewed as terms of $\lambda-$calculus and act as "proof-schemata", as for universally quantified types. We examine here the critical case of Impredicative Type Theory, i.e. Girard's system F, where type-quantification ranges over all types. Coherence and decidability properties are proved for prototype proofs in this impredicative context.

1 Introduction

1.1 Universal Quantification

In universally quantified propositions of everyday mathematics, such as $\forall x\mathcal{P}(x)$, x is meant to range on some intended collection of individuals (the reals, the complex numbers ...). This is a first order quantification. How do mathematicians prove a universal proposition of this kind? Assume, for instance, that in $\mathcal{P}(x)$, x is a real number variable. We seek to prove $\forall x\mathcal{P}(x)$, i.e. that for *any* real x, $\mathcal{P}(x)$ holds for this specific real. Any working mathematician would prove $\forall x\mathcal{P}(x)$ by saying: $\ll$ let x be an arbitrary or "generic" real number, then observe ...$\gg$ and would write *one* single proof, valid for *any* particular real, as there is no way to inspect all reals, one by one. In other words, his/her core (often implicit) remark would be that the proof does not depend on the specific (and arbitrary) real chosen, but only on *the assumption* that x is a real number.

In type-theoretic terms, a sound proof would only depend on the *type* of x, not on its value. Thus, if we want to prove $\mathcal{P}$ for a specific real, π say, we only need to replace everywhere in our proof, the "generic real" by the specific value, π. This is sound, as we used in the proof only properties that are verified by any real.

Herbrand called that kind of "uniform" proofs **prototype**:

> ≪... when we say that a theorem is true for all x, we mean that for each x individually it is possible to iterate its proof, which may just be considered a *prototype* of each individual proof.≫ (pp. 288-9, note 5 in [Gold87]).

In conclusion, the *proof* of a universally quantified statement is not understood by following the naif (tarskian style) interpretation of "$\forall x \ldots$", as "for all x ...": in no way, "$\forall x \ldots$" is used, in a proof, in the sense of the inspection of "all instances" in the intended model, yet its meaning and use refer to x as generic in a prototype proof.

The theory of types of $\lambda-$calculus provides a sound formal frame for this informal notion, in particular in the second order case. The reference to Type Theory and $\lambda-$calculus is related to the intuitionistic perspective, concerning truth and provability, which is assumed in this paper. In particular, in the "realizability interpretation" of Intuitionistic Systems, the meaning of a statement is given by the "set of its realizers" (that is, its possible or candidate proofs - if any, as this set may be empty). This constructive understanding of logical systems is also related the so called BHK explanation of the intuistionistic meaning of the logical connectives, [Troe73], [TroVan73] for both approaches. In either semantics, one constructively gives meaning to a statement, to a defined mathematical concept, by discussing its provability, as truth *is* provability. (Note: Prototype proofs in Type Theory were introduced in [FruLo98], whose first half is a preliminary version of the present paper. The decidability result in section 6 below is only conjectured in [FruLo98]).

1.2 Impredicative Second Order Quantification

Impredicative definitions explicitly show up, when quantification is given over sets or predicates, not just individuals, as in the first order case: in this case the definiens may use the definiendum, which is a set or a predicate. That is, in order to deal formally with a "totality" of concepts or predicates or sets, one needs a second order language, namely a formalized language that may represent, internally, quantifications over collections of predicates or sets. Analysis, for example, may be viewed as second order Arithmetic, since real numbers may be defined as sets of integers.

More formally, a set S is impredicatively defined if for some predicate $\mathcal{P}$, one has $S = \{y/\forall X.\mathcal{P}(X, y)\}$, where y is an individual-variable and X is an set-variable (that is, X is intended to range over *sets* of individuals). Thus, S is a particular set and, if one wants to check wether a given a is in S, one needs to consider the property $\forall X \mathcal{P}(X, a)$, with X ranging over *all* sets (or predicates), including S itself. That is, one has to handle, in particular, the "circular" case $\mathcal{P}(S, a)$. One may then wonder if an analysis of $\forall X \mathcal{P}(X, a)$ is still possible in terms of "prototype proofs", in the sense given above for first order sentences. In particular, one may wonder what it may mean exactly for an instance of variable X to be "generic", in the second order case. As a matter of fact, the "generic" use of X is even more crucial here. Scanning all possible instances of X would indeed include $\forall X \mathcal{P}(X, a)$ itself and lead to an unbreakable circularity. But $\ll$ ...the belief that we must run through all individual cases rests on a confusion of "numerical" generality. [...] We do not establish specific generalities by running through individual cases but by logically deriving certain properties from certain others. $\gg$ as pointed out by Carnap in 1931, [Car31].

In our example, assume that S is a set of integers. In order to check whether n is in S, one should logically derive $\mathcal{P}(X, n)$ from the only assumption that X has the property of "being a set of integers" (i.e. from an assumption on its type and nothing else); thus, we do not run throughout all sets of integers, including the one we are defining, but we just make a logical derivation based on the properties of the "type of the sets of integers". In general,

> $\ll$ ...if we reject the belief that it is necessary to run through individual cases and rather make it clear to ourselves that the complete verification of a statement means nothing more than its logical validity for an arbitrary property, we will come to the conclusion that impredicative definitions are logically admissible. $\gg$, [Car31].

However, some technical questions need to be clarified and we will deal with them within the contructive frame of Girard's System F or II order $\lambda-$calculus (see section 2). How can we be sure that the proof given is actually independent from the specific X used throughout the proof? How can we prove that from the given proof of a specific case, we can actually reconstruct *uniformly* a proof of the general case, that is of $\forall X.\mathcal{P}(X, a)$? Or, is it decidable that X is truely generic and that the proof is a prototype, in the sense of Herbrand?

Mathematicians solve these questions, in the practice, by handwaving, experience and a common insight into proofs. This is perfectly sound in the first order case, where the stratification of individuals and predicates (or sets) poses no problem. It is a more delicate point in second order systems, when the issue of impredicativity actually raises.

Constructive logical systems, Type Theory in particular, by looking at proofs explicitly, as terms or computations, give a precise answer to these questions, as a simple corollary of a (difficult) result, as shown below. In our views, this may contribute to set on more solid grounds impredicatively given properties.

2 System Fc and Impredicativity

System F is known as Impredicative Type Theory, or Polymorphic $\lambda-$calculus; it was introduced by Girard, [Gir71] (see [GLT89] for a more recent presentation and [AL91] for its categorical semantics). It consists of *types* and *terms* (well typed terms). As a key component of its expressive power, the type system of F allows *explicit quantification on type variables* (second order quantification); its proof-theoretic strenght is such as to prove, by the normalization theorem, the consistency of (II order) Arithmetic.

In system F a **type** is either a type variable, an arrow type or a universally quantified (or polymorphic) type. (One may add atomic types, when dealing with specific extension of the "pure theory" presented here; however, most key predicates for Logic and Computer Science are codable in the pure system, see [GLT89].) Types then are constructed using the following schemes:

- *variables:* X, Y ... are types.
- *arrow types:* $\tau \rightarrow \sigma$ is a type, if σ and τ are types.
- *universal types:* $\forall X.\tau$ is a type, if τ is a type.

A **term** is either a variable, an abstraction, an application, a type abstraction, or a type application. Thus, terms are constructed using the following schemes:

- *variables:* x^τ of type τ, if τ is a type.
- *abstraction:* $\lambda x^\tau.M$ of type $\tau \rightarrow \sigma$, if σ is the type of M (λ *binds* term-variables in terms, or x^τ is not *free* in $\lambda x^\tau.M$).
- *application:* MN of type σ, if M is of type $\tau \rightarrow \sigma$ and N of type τ.
- *type abstraction:* $\Lambda X.M$ of type $\forall X.\tau$, if M is of type τ and X is not free in the type of any free term-variable of M (Λ binds type-variables in terms).
- *type application:* $M\sigma$ of type $\tau[\sigma/X]$, if M is of type $\forall X.\tau$ (where $\tau[\sigma/X]$ is the result of the replacement of all free occurrences of X by σ in τ.)

Reduction rules

$$\begin{array}{ll} (\lambda x^\tau.M)N \xrightarrow{\beta_1} M[N/x] & \lambda x^\tau.(Mx) \xrightarrow{\eta_1} M \text{ if } x \text{ is not free in } M \\ (\Lambda X.M)\tau \xrightarrow{\beta_2} M[\tau/X] & \Lambda X.(MX) \xrightarrow{\eta_2} M \text{ if } X \text{ is not free in } M \end{array}$$

We will write $\xrightarrow{F}$ for the reflexive and transitive closure of the union of these reductions, and $=_F$ for the symetric closure of $\xrightarrow{F}$.

Second order quantification is explicitly given by the rule of *type abstraction*, which binds second order variables, both in types and terms. Thus, $\forall X.\tau$ is defined, as a type, by a quantification over the collection of all types, which includes $\forall X.\tau$, the "definiendum". Moreover, by virtue of the Curry-Howard isomorphism, polymorphic types can be viewed as second order logical propositions. Under this "interpretation", a type of the form $\forall X.\tau$ "means" that the property τ is possessed by *all types* (in particular, by $\forall X.\tau$ itself). Finally, the terms of system F compute functions, either on *terms* (see the β_1 axiom) or on *types* (see the β_2 axiom). In particular, a polymorphic term, i.e. a term in a universally quantified type, takes types as inputs (including, possibly, its own type) and gives terms as outputs, by β_2. Here is then a the typical form of impredicativity, namely a type-theoretic formalization of the "vicious circle". It even shows up at two levels, for types and for terms. However, exactly because of this constructive frame, where proofs are seen as computations (are coded by terms), it will be possible to look closely at the nature of "prototype proofs". Indeed, prototype proofs in system F have strong "coherence" and decidability properties.

Axiom C and its meaning

The key point of our analysis relies on the observation that *all* types in system F are "generic", in the sense to be specified, and that, from a computational point of view, they act like variables. In short, in the constructive frame of second order Type Theory, outputs do not depend on inputs, when the input is a type: in this case, only the *type* of the output may depend on the input, not its "value" (see below). This is blatantly false for first order terms, in Type Theory or everywhere in mathematics: functions and computations do depend on inputs as first order individuals, as soon as a sufficiently expressive system is given.

This observation will be established in two steps. First, in this section, by a simple remark on the compatibility of an axiomatic extension of system F; later on, by the Genericity Theorem.

The first remark is inspired by a result in [Gir71]: in system F, there is no definable term that *discriminates* between types. That is, there is no term J_σ such that J_σ applied to type ρ is 1 if $\sigma = \rho$, and is 0 if $\sigma \neq \rho$. In other words, there is no term whose output values are all in the same type (the type of integers, or any other type with at least two elements) and depend on the input type. This idea was taken up in [LMS93] by extending system F with the following axiom, some sort of a "generalized dual" of Girard's result [1]:

Axiom C: *If* $M : \forall X.\sigma$ *and* $X \notin FV(\sigma)$ *then for all* τ, τ',$M\tau = M\tau' : \sigma$.

[1]Independently of Girard's remark, in [CMMS91] a similar extension was proposed, for the purposes of subtyping, a notion motivated by programming (see also [CGL95]).

Axiom C intuitively means that an input type (τ), which is not used to establish the type (as σ does not depend on X) of the corresponding output value ($M\tau$), bears no information as input. So if M has the type $\forall X.\sigma$ and X is not free in the type σ (i.e. σ is not a function of X), then it does not matter whether one applies M to τ or τ' and one may consider both results to be equal. Equivalently, since there are no type discriminators by Girard's remark, Axiom C forces terms of universally quantified type, whose outputs live in the same type, to be constant. Informally, this is sound, because we are in a constructive frame and types have the intended meaning of a possibly infinite domain of interpretation. Thus a term, as effective computation, cannot compare nor discriminate on the grounds of possibly infinite information. Or,to put it otherwise, Girard's result above or Axiom C are not limitations or "negative" properties of system F: they instead soundly stress the constructive nature of this intuitionistic system whose computations handle possibly infinite inputs like "blackboxes".

We write Fc for the extension of F by Axiom C and $=_{Fc}$ for the corresponding extension of $=_F$. Axiom C is not derivable in system F. Indeed, let $x : \forall X.\sigma$ and $X \notin FV(\sigma)$ then for any τ and τ', $x\tau$ is a normal form, different from $x\tau'$. However, Axiom C is formally *compatible* with F, as there are models of Fc. As a matter of fact, all known and non-trivial models (e.g. not term-models nor models of Type:Type) realize Axiom C, see [Lon95]. In short, all "parametric" models of F, in the sense of Reynolds, [MR92], the coherent domains, [Gir86], and the PER models are all models of Fc [2]. It is also possible to extend system F by a reduction relation which is strongly normalizing, Church-Rosser and induces exactly the Fc-equality (which is thus decidable, see [Bel97]).

The soundness of Fc gives the first hint towards the "generic" nature of types as inputs, in system F. That is, we may consistently consider each type exactly as a variable, at least *under the special circumstances* that it is an input for a term M of type $\forall X.\sigma$, where $X \notin FV(\sigma)$. The Genericity Theorem extends this remark to all universally quantified types and their terms.

3 The Genericity Theorem.

In [LMS93], Axiom C was introduced in order to prove the Genericity Theorem below (note that there is no restriction on σ).

Theorem (Genericity). *Let M and N have type $\forall X.\sigma$. Then:*

$$(\text{Exists } \tau,\ M\tau =_{Fc} N\tau) \Longrightarrow M =_{Fc} N.$$

This theorem shows that two polymorphic terms that are equal on *one* input

[2] The categorical significance of the PER models, i.e. the meaning of "quantification as product" as well as the meaning of Axiom C are both given by the validity in the Effective Topos of the Uniformity Principle, see [Lon87], [LM91].

type are equal on *any* input type. In other words, the behaviour of polymorphic terms is so "uniform" that one can reduce Fc equality on *every* possible types to Fc equality on one *single* type τ (no matter which one!). That is, if ($M\tau =_{Fc} N\tau$) then $MX =_{Fc} NX$. The proof is far from obvious.

With reference to very different matters, as an analogy, one may remember the "regular behaviour" of analytic functions of complex variable: when known on the border of a regular shape, they are known everywhere inside. Thus, if two of these functions coincide on the border, they coincide everywhere [Rud80].

This regularity or uniformity could be surprising, and perhaps it is, but this is actually what makes second order impredicative Type Theory become safe and interesting, as a constructive theory, similarly as the regularities of analytic functions make them relevant.

For the purposes of our forthcoming application, observe that the Genericity Theorem in [LMS93] is actually shown by proving the following Main Lemma:

Main Lemma (to Genericity Theorem). *Let M and N have type σ. Then:*

$$(\text{Exists } \tau, [\tau/X]M =_{Fc} [\tau/X]N) \Longrightarrow M =_{Fc} N.$$

Recall now that, if M is a polymorphic term, it can take as input *any* type, and in particular types that are more complex that its own type. One can then wonder what happens in these "circular" cases. The informal answer inspired by the Genericity theorem is, so far: "It happens the same as on a simpler input type, because the computational behaviour of M "in extenso" is determined by its behaviour on a single type". In this precise and strong sense, the term or computation does not depend on the input type, as possibly "carrying" infinite information.

As shown in [FruLo98], the proof of Genericity is (difficult but) "elementary". More precisely, it is possible to code it into PRA (Primitive Recursive Arithmetic), provided that the Church-Rosser property for system F is assumed (but its proof is elementary and easy). Indeed, PRA is an elementary arithmetic theory, where one can handle basic mathematical computations and deductions. Thus, in spite of the circularity generated by polymorphism, a strong "regularity" property of terms in system Fc is established by the Genericity Theorem and the proof of this regularity is elementary, i.e. logicaly complex reasonings are not necessary to deal with this key property of the impredicativity of Fc. Moreover, in [FruLo98] some hints are given on how Genericity may help to understand technically the surprising "uniformities" in the Normalization Theorem for system F, the core of this system: it displays why (sets of) candidates of reducibility let the proof work $\ll$ ... as if the rule of universal [second order] abstraction (which forms functions defined for arbitrary types) **were so *uniform* that it operates without any information at all about its arguments.**$\gg$ [GLT89], [p.115]. (The compatibility of Axiom C and) the Genericity Theorem show, precisely, in which sense functions defined for arbitrary types operate without any information about their arguments.

4 Prototype Proofs in Type Theory

In Type Theory, or according to the realizability or BHK interpretations, the constructive meaning of $\forall X.\sigma$, the crucial, impredicatively given type, is defined as follows. A proof-term $M\!:\forall X.\sigma$ is a computation or function that takes *any type* ρ to a proof $M\rho\!:[\rho/X]\sigma$. Thus, from a term $M\!:\forall X.\sigma$ one can reconstruct the terms or proofs $M\rho$ for *each* specific instance $[\rho/X]\sigma$ of σ. Yet, as already stressed, a proof of $\forall X.\sigma$ is not constructed by running through all specific cases or input types ρ, but by giving a prototype proof, in the sense of Herbrand, which uniformely works on each instance ρ. The most obvious prototype proof and generic case is given by $MX\!:\sigma$. However, this does not save us from the circularity of impredicativity, as variables in Type Theory have a double "status": they are atomic entities (types in this case) but they also formally represent the mathematical use of "variables" as arbitrary elements of the intended domain of variation, since they may be instantiated by any element of that domain, the collection of all types, in this case. Then, in particular, the variable X which may occur in σ can be instantiated by σ or even $\forall X.\sigma$.

Our thesis though, in view of the Genericity Theorem, is that an arbitrary specific instance type, possibly simpler than σ, may suffice to determine a fully general proof. The idea then is to start from a specific instance $[\rho/X]\sigma$ and discuss the prototype nature of its proof, if any. In a sense we want to describe the backwards process, w.r.t. the one described above, as we want to go from a proof of $[\rho/X]\sigma$ to one of $\forall X.\sigma$.

Assume then that from a proof N of an instance $[\rho/X]\sigma$ one tries to reconstruct a proof of the universal proposition $\forall X.\sigma$. In general, this may not be possible. It is possible, though, when the *structure* of a specific proof N of $[\rho/X]\sigma$, that is of $N\!:[\rho/X]\sigma$, is "parametric" in ρ or it may be described uniformely as a substitution of a type variable by ρ. In that case, we call N a prototype proof:

Definition. *Given a type σ, we say that a type ρ is* **generic** *and a proof $N\!:[\rho/X]\sigma$ is a* **prototype** *if there exists $M\!:\sigma$, such that X is not free in the type of a free term variable of M and $[\rho/X]M =_{Fc} N\!:[\rho/X]\sigma$.*

Notice that if ρ and a proof $N\!:[\rho/X]\sigma$ are, respectively, generic and prototype, by $M\!:\sigma$, then $\Lambda X.M\!:\forall X.\sigma$. That is, the construction (existence) of M, from the prototype and generic proof and type N and ρ, immediately gives a proof of the universal statement. As, by the reduction rule β_2, the converse is trivial, i.e. $(\Lambda X.M)\rho =_{Fc} [\rho/X]M\!:[\rho/X]\sigma$, then, given a type, there exist generic and prototype type and proof *if and only if* the corresponding universal statement is provable. Clearly, not any proof nor type of a specific instanciated type need be prototype and generic: for example, $[\rho/X]X$ has no prototype proof with ρ generic, otherwise the universal statement $\forall X.X$, the absurdum or empty type, would be provable. We prove next the "coherence" of this notion and, in section 6, that, given N,σ, X and ρ, it is decidable whether N is a prototype proof with ρ generic.

5 Coherence for second order Prototype Proofs

We focused on a second order notion of prototype proof and generic type. Of course, the definition can be easily extended to first order statements, the more usual ground of "prototype" proofs in mathematics: if r is an arbitrary real number and the proof of $P(r) = [r/x]P$ does not depend on r, but *only* on its type - the type R of reals - (or r is generic and the proof is a prototype similarly as in the definition above), any mathematician would say that we actually proved $\forall x.P(x)$. Type-theoretically, but informally here, one should just display the proof-term $[r/x]M = N{:}\,[r/x]P$, for $r{:}\,R$. (Note that ρ is generic in a second order prototype proof $N{:}\,[\rho/X]\sigma$, if the proof depends *only* on "*ρ being a type*").

As already mentioned, the first order case is not problematic at all: individuals are distinct from propositions and there is no apparent vicious circle. This is not so in the impredicative case, which motivates the doubts of many in the use of impredicative second order quantifications (and variables). Thus, the very simple notions of generic types and prototype proof turn out to be a more delicate issue in impredicative systems.

However, exactly because of the relevant property of System F given by the Genericity Theorem, we are now able to assure that prototype proofs are sound, also when the generic type may be as complex as the universal assertion to be proved. The soundness is given, similarly as in Category Theory, by a "*coherence* result", which states the unicity of the reconstruction of the proof of the universal statement from a specific prototype proof and generic type.

Theorem (Coherence). *Given a type σ, let ρ and a proof $N{:}\,[\rho/X]\sigma$ be generic and prototype, respectively. Then,*
if $[\rho/X]M =_{Fc} N =_{Fc} [\rho/X]M'{:}\,[\rho/X]\sigma$, one has $M =_{Fc} M'$ and, thus,

$$\Lambda X.M =_{Fc} \Lambda X.M'{:}\,\forall X.\sigma.$$

The proof is immediate corollary to the Genericity Theorem or, more precisely, of its Main Lemma, stated above. The meaning of this fact should be clear: it says that no matter how we extract a proof of a universal statement from a prototype one of a specific instance, in any case we obtain just one proof (modulo "$=_{Fc}$"). Thus, also the type ρ does not matter, or it is truely generic, since from the **unique** proof $\Lambda X.M$ of $\forall X.\sigma$ we can obtain, uniformely and effectively, proofs for each instance $[\rho'/X]\sigma$, just by application $(\Lambda X.M)\rho'$. (It is fair to call this fact a "coherence" property, by an informal analogy to commuting diagrams, up to "$=$", in coherence theorems of Category Theory).

The independence of the proof of the universal statement from the specific "structure" of a proof of a specific instance, as well as from the generic type used, garanties that, exactly in the "shaky" second order case, the mathematical soundness of those statements. The system is "coherent" both in the categorical-technical sense, and in the sense of the possibility of disregarding the complexity of the instantiating type, since all types are generic and act like variables.

As already mentioned, this garanty is given "exactly" in the critical impredicative second order case, as the (obvious variant of the) Genericity Theorem is clearly (and fortunately) false in the first order case (first order terms, as computable functions, must allow lots of computations and strictly depend on inputs).

6 Decidability

In this section it is shown that it is decidable, given types σ, ρ, X and a term N, whether ρ is generic and N is a prototype proof of $[\rho/X]\sigma$.

Clearly, one has to to decide if there exists $M{:}\,\sigma$, such that X is not free in the type of a free term variable of M and

$$[\rho/X]M =_{Fc} N{:}\,[\rho/X]\sigma.$$

In [Bel97], it is shown that system Fc is Church-Rosser and (strongly) normalizes, i.e. that each term has a unique normal form (no reduction rules are applicable). Thus $=_{Fc}$ is decidable, as

$$[\rho/X]M =_{Fc} N \Longrightarrow [\rho/X]M' \equiv N'$$

where M' and N' are the normal forms of M and N, respectively. The reverse implication is obvious. (Note that $[\rho/X]M'$ is the normal form of $[\rho/X]M$, as replacing a type variable by a type does not changes the redexes, see [Bar90]).

Thus, we have to decide whether there exists an M satisfying a decidable property. This is a semi-decidable predicate in general; yet, by re-adapting to terms a notion in [LMS93] for types (the X-contexts), one may reduce it to a decidable one.

Definition. *M is an X-context of N for ρ, if $[\rho/X]M \equiv N$.*

Lemma. *Assume that ρ occurs exactly k-times in N and chose X not free in N, then there are 2^k different X-contexts M of N for ρ.*
Proof. Obvious.

Theorem. *Given σ, ρ, X and $N{:}\,[\rho/X]\sigma$, it is decidable whether ρ is generic and N is prototype.*
Proof. We have to find an $M{:}\,\sigma$, if any, such that X is not free in the type of a free term variable of M and $[\rho/X]M =_{Fc} N{:}\,[\rho/X]\sigma$. In case such an $M{:}\,\sigma$ exists, let M' and N' be the (unique) normal forms of M and N. Then $[\rho/X]M' \equiv N'$ and M' and N' differ only by, possibly, the occurences of ρ; that is, M' belongs to the set of X-contexts of N' for ρ. Thus, any witness for a proof to be prototype (and a type generic) must belong to a finite set.

Recall now that typing for (typed) terms in system $F(c)$ is decidable, i.e. one can check whether a given term is well-typed and recostruct its (unique) type, if it exists. Then, the decision algorithm we seek, works as follow:
- reduce N to its normal form N',
- compute its associated finte set of X-contexts of N' for ρ,
- check whether any of these terms has type σ and that X is not free in the type of one of its free term variables,

if not, N is not prototype and ρ is not generic; otherwise, the X-context $M' : \sigma$ is the required witness and, by the Coherence Theorem, it is unique, up to $=_{Fc}$.

This concludes the proof (or, more precisely, it gives the guidelines for a ... *prototype* proof of the Theorem.)

Conclusion

Within the constructive approach to provability proposed by (impredicative) Type Theory, we could formalize the informal notion of prototype proof and generic argument. This formal treatment was then applied to justify impredicative types as propositions, by coherence and decidability properties. As already pointed out, coherence does not hold for first order prototype proofs. Problem: in the first order case, can one have a decidability result, at least? Of course, the context and the precise conditions should be fully specified: the Calculus of Constructions could be a suitable environment for this, and one may expect a negative answer in general, but a positive one in some key cases. The challenge would be to prove that the positive cases are the ones of mathematical interest.

References

[AL91] A. Asperti and G. Longo. *Categories, Types, and Structures.* MIT Press, 1991. (Currrently downloadable from: http://www.dmi.ens.fr/users/longo)

[Bar90] H. Barendregt. The Typed λ-calculus. in *Handbook of theoretical computer science.* Vol. B : formal models and semantics. van Leeuwen, Jan (ed.). The MIT Press, 1990.

[Bel97] G. Bellè. Syntactical properties of an extension of Girard's System F where types can be taken as "generic" inputs. Preliminary note, DISI - Università di Genova. E-mail: gbelle@disi.unige.it.

[Car31] R. Carnap. The logicist foundation of mathematics, in P. Benacerraf, H. Putnam, *Philosophy of mathematics; selected readings*, Prentice-Hall philosophy series, 1964.

[CGL95] G. Castagna, G. Ghelli, and G. Longo. A Calculus for Overloaded Functions with Subtyping. *Information and Computation*, 117(1):115–135, February 1995. (Prelim. version: ACM Conference on LISP and Functional Programming, pp.182-192, San Francisco, 1993).

[CMMS91] L. Cardelli, S. Martini, J.C. Mitchell, and A. Scedrov. An extension of system F with subtyping. *Information and Computation* 94, pages 4–56, 1994. First appeared in the proceedings of the Conference on Theoretical Aspects of Computer Software (Sendai, Japan), T. Ito and R. Meyer, eds., Lecture Notes in Computer Science 526, pages 750–770, Springer-Verlag, 1991.

[FruLo98] T. Fruchart and G. Longo. Carnap's remarks on Impredicative Definitions and the Genericity Theorem. In *Logic, Methodology and Philosophy of Science: Logic in Florence, 1995.* Cantini et al. eds., Kluwer, 1998.

[Gir71] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. Proceedings of the *2nd Scandinavian Logic Symposium*, J.E. Fenstad, ed., pages 63–92, North-Holland, 1971.

[Gir86] J.-Y. Girard. The system F of variable types, fifteen years later, *Theoretical Computer Science*, vol 45, pages 159–192.

[GLT89] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and types.* Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press, 1989.

[Gold87] H. Goldfarb, *Jacques Herbrand: logical writings*, 1987.

[Lon87] G. Longo. Some aspects of impredicativity: notes on Weyl's philosophy of Mathematics and on todays Type Theory, *Logical Colloquium 87*, Studies in Logic (Ebbinghaus et al. eds), North Holland, 1989.

[Lon95] G. Longo. Parametric and type-dependent polymorphism. *Fundamenta Informaticae*, 22(1-2):69–92, 1995.

[LM91] G. Longo and E. Moggi. Constructive natural deduction and its ω-set interpretation. *Mathematical Structures in Computer Science* vol. 1, pages 215–253, 1991.

[LMS93] G. Longo, K. Milsted, and S. Soloviev. The Genericity Theorem and the notion of parametricity in the polymorphic λ-calculus. *Theoretical Computer Science* 121, pages 323–349, 1993.

[MR92] Q. Ma and J.C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. Proceedings of the *Conference on Mathematical Foundations of Programming Semantics*, S. Brookes, M. Main, A.

Melton, M. Mislove, and D. Schmidt, eds., Lecture Notes in Computer Science 598, pages 1–40, Springer-Verlag, 1992.

[Rud80] F. Rudin. *Real and Complex Analysis*, MacGraw Hill, 1980.

[Troe73] A. Troelstra. *Metamathematical investigation of intuitionistic arithmetic and analysis*, Lecture Notes in Mathematics 344, Springer Verlag, 1973.

[TroVan73] A. Troelstra and D. VanDalen. *Constructivism in mathematics.* Vol. I and II, North-Holland, 1988.