Jan

On the propositions-as-types principle

Thierry Coquand

CMU and INRIA

draft

1 A reformulation of the propositions-as-types principle

We try to present the calculus of construction in such a way that it makes clear the connection with the first system of Martin-Löf and shows why it is simply an application of the Curry-Howard analogy between propositions and types to the Church calculus.

1.1 What were Martin-Löf's motivations for having a type of all types ?

Three principles entail this idea

1. we want quantification over predicates, propositions
2. Russel's doctrine of types: the range of significances of propositionnal functions form types
3. the idenfication of propositions and types

Indeed, by the first and second points propositions must form a type. If propositions and types are identified, then this type is at the same time the type of types. But there is something wrong somewhere, as the calculus we obtained by using a type of all types lead to inconsistency.

We shall try to formalise a calculus which respects the first and second points, but we shall restrict the third point. We retain only the idea that each proposition corresponds to one certain type, namely its type of proofs.

1.2 A formalisation of the notion of proposition's type of proofs

We start with the usual Church's type calculus. One has a special type o of all propositions, one constant $\Rightarrow: o \to o \to o$, and one polymorphic constant $\Pi : (\alpha \to o) \to o$.

We shall formalise the idea that propositions are types in the following way. Let introduce one operator T such that

1. if φ is a term of type o, then $T(\varphi)$ is a (new) type
2. if φ and ψ are two terms convertible of type o, then $T(\varphi)$ and $T(\psi)$ are equals
3. if φ and ψ are two terms convertible of type o, then $T(\varphi \Rightarrow \psi)$ is equal to $T(\varphi) \to T(\psi)$
4. if f is of type $\alpha \to o$, t is of type $T(\Pi(f))$ and u is of type α, then $(t\ u)$ is a well-formed term of type $T((f\ u))$
5. if t is of type $T(\varphi)$, then $\lambda x^\alpha t$ is of type $T(\Pi(\lambda x^\alpha \varphi))$

Note that these rules simply explain how to translate each proposistions in a type in such a way that the Heyting's semantic of propositions in intuitionnistic logic is preserved, i.e. that $T(\varphi)$ can be thougth as the types of all proofs of the proposition φ.

We can remark that the arrow between types in the range of T is definable. More precisely, one must add the rules

6. if φ and ψ are two terms of type o, then $T(\varphi) \to T(\psi)$ is equal to $T(\Pi(\lambda x^{T(\varphi)}\psi))$

Remark. This has a clear analogy with the new formulation by Martin-Löf of the universes formation in his new theory of type. We may be even more precise: take the calculus of Martin-Löf, and add a (special) universe U with the rules

$$\frac{A\ type \quad (x:A)B:U}{(\forall x:A)B:U}$$

$$\frac{A:U}{T(A)\ type}$$

$$\frac{A\ type \quad (x:A)B:U}{T((\forall x:A)B) conv (\forall x:A)T(B)}.$$

Note that the only difference between those rules and the ones of Martin-Löf is the general product formation $(\forall x : A)B : U$ with an arbitrary A *type* and not only one A of the form $T(a)$ for a term a of type U. Then, one get the calculus of construction and a non-predicative (but still coherent) calculus.

2 What is the calculus of construction ?

A careful study of the previous calculus shows that it contains the second-order calculus of Girard-Reynolds. For example, the generic identity is the term $\lambda A^o \lambda x^{T(A)} x$ which is of type $T(\Pi(\lambda A^o T(\Pi(\lambda x^{T(A)} A))))$. Indeed, if A is of type o and x of type $T(A)$, then x is of type $T(A)$, so, by the rule for the operator T, $\lambda x^{T(A)} x$ is of type $T(\Pi(\lambda x^{T(A)} A))$.

The nice point is that we do not need any more to state any axioms and deduction rules for our system of types. The logic becomes simply the expression of the fact that a proposition is true if, and only if, the type of its proofs is non empty.

Actually, our calculus contains a lot more. If we try to simplify the notation, and simply omit the operator T, one gets a part of the calculus of construction as presented in [Coquand, 85]. So this extension of the calculus of Church has still the normalisation property. The calculus of construction appears in this way to be the expression of the propositions-as-types principle for the system of [Church, 40].

This extension of Church's calculus presents some novelty. There exists now dependant types. For example, the expression $(T(\varphi) \to o) \to o)$ is a type which depends on a term $\varphi : o$. It seems natural to add the rules

if t is a type, and α is a type, then $\Pi(\lambda x^\alpha t)$ is a type
and also

if u is of type t, then $\lambda x^\alpha u$ is of type $\Pi(\lambda x^\alpha t)$.

With these new rules, the calculus becomes exactly the calculus of constructions as presented in [Coquand, 85].

3 A new formulation of the calculus of contruction

Definition. the class of term of the calculus of construction is the class defined by the following inductive rules

1. *Type* and *Prop* are terms
2. an identifier is a term
3. an integer (de Bruijn index) is a term
4. if M and N are terms, then $(M\ N)$ is a term
5. if M is a term, then $\lambda(M)$ is a term
6. if M and N are terms, then $\Pi(M,N)$ is a term .

We shall not need then any special constant such as $\Rightarrow$, neither special rules of inference. The motivation is that we shall simply express the proposition-as-types principle: we identify a proposition with the type of its proofs, so that the previous quantifier $(\forall x : A)\varphi$ becomes the product $\Pi(A,\varphi)$. The $A \to B$ is definable as $\Pi(A,B)$. It seems so that for building a type system with an associate logic, all we need is to have the λ-operation and the product formation. All the (semantic) rule about true formulae of Church's calculus appear as derived rules of a very simple typing mechanism.

We shall generalize the previous rules of the construction calculus by the introduction of *Type*, the type of so-called "context", and we shall try to extend this calculus with four levels.

Assignments

$$\overline{\text{the empty assignment is valid}}$$

$$\frac{\Gamma \text{ is valid} \quad \Gamma \vdash M : Type \quad x \text{ does not appear in } \Gamma}{\Gamma, x : M \text{ is valid}}$$

Type Inference Rule

$$\frac{\Gamma \vdash M : Prop}{\Gamma \vdash M : Type}$$

$$\frac{\Gamma \text{ is valid}}{\Gamma \vdash Prop : Type}$$

$$\frac{\Gamma \text{ is valid} \quad x \text{ occurs in } \Gamma \text{ with the type M}}{\Gamma \vdash x : M}$$

$$\frac{\Gamma, x : M \vdash N : P}{\Gamma \vdash \lambda x.N : (\Pi x : M)P}$$

$$\frac{\Gamma, x : M \vdash N : Prop}{\Gamma \vdash (\Pi x : M)N : Prop}$$

$$\frac{\Gamma, x : M \vdash N : Type}{\Gamma \vdash (\Pi x : M)N : Type}$$

$$\frac{\Gamma \vdash M : (\Pi x : A)R \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : [N/1]R}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash R : Type \quad A \text{ conv } R}{\Gamma \vdash M : R}$$

4 Application: Peano Arithmetic

Let us introduce a new type $Int : Type$, the special constants $0 : Int$, and $S : [n : Int]Int$, with a recursion operator $rec : [u : Int][P : [x : Int]Prop][h_1 : (P0)][h_2 : [x : Int][h_3 : (Px)](P(Sx))](Pu)$ with $(rec0af)$ convertible to a, and $(rec(Sn)af)$ convertible to $(f(recnaf))$ (this appears as a generalisation of the system T of Gödel). Then, our identification of a proposition and types is sufficient for the derivation of all Peano axiom, without any other assumption.

and a recursion operator

5 The calculus of Martin-Löf 1971

It appears now as an attempt of one reflection's principle. We can view, by the operator T, the propositions as names for certain types. One can ask if each type is so named, and this question turns out to be equivalent to the question whether the type o is so named. If one introduce a special constant V of type o such that $T(V)$ is equal to o, then one gets exactly the system of [Martin-Löf, 71].

In this presentation, this appears a bit artificial, and, as a matter of fact, [Girard, 72] has shown that this assumption leads to a non-normalisable term.

Conclusion

The calculus of construction is the natural expression of the propositions-as-types principle for the system of [Church, 40]. Once we express it, the axioms and inference rules become surpefluous and are a consequence of the identification of a truth of a proposition with the fact that its type of proofs is non-empty.

It appears so that if we believe
propositions = types
then we must be predicative. But it is possible to be non predicative if we simply view propositions as special kind of types.