Induction Principles Formalized in the Calculus of Constructions[1]

Gérard Huet

INRIA[2]

France

Abstract

The Calculus of Constructions is a higher-order formalism for writing constructive proofs in a natural deduction style, inspired from work of de Bruijn [2, 3], Girard [12], Martin-Löf [14] and Scott [18]. The calculus and its syntactic theory were presented in Coquand's thesis [7], and an implementation by the author was used to mechanically verify a substantial number of proofs demonstrating the power of expression of the formalism [9]. The Calculus of Constructions is proposed as a foundation for the design of programming environments where programs are developed consistently with formal specifications. The current paper shows how to define inductive concepts in the calculus.

A very general induction schema is obtained by postulating all elements of the type of interest to belong to the standard interpretation associated with a predicate map. This is similar to the treatment of D. Park [16], but the power of expression of the formalism permits a very direct treatment, in a language that is formalized enough to be actually implemented on computer. Special instances of the induction schema specialize to Nœtherian induction and Structural induction over any algebraic type. Computational Induction is treated in an axiomatization of Domain Theory in Constructions. It is argued that the resulting principle is more powerful than LCF's [13], since the restriction on admissibility is expressible in the object language.

Notations

We assume the reader is familiar with the Calculus of Constructions, as presented in [7, 9, 10, 11]. More precisely, we shall use in the present paper the extended system defined in Section 11 of [8]. The notation $[x : A]B$ stands for the algorithm with formal parameter x of type A and body B, whereas $(x : A)B$ stands for the product of types B indexed by x ranging over A. Thus square brackets are used for λ-abstraction, whereas parentheses stand for product formation. The atom $Prop$ is the type of logical propositions. The atom $Type$ stands for the first level in the predicative hierarchy of types (and thus we have $Prop : Type$). We abbreviate $(x : A)B$ into $A \to B$ whenever x does not occur in B. When $B : Prop$, we think of $(x : A)B$ as the universally quantified proposition $\forall x : A \cdot B$. When x does not occur in B and $A : Prop$,

---

[1]This article was presented at TAPSOFT 87, Pisa, March 1987, and appeared in the proceedings, published as Springer-Verlag Lecture Notes in Computer Science 249, pp 276–286.

[2]This research, done during a sabbatical at Carnegie Mellon University, was supported in part by the Office of Naval Research under contract N00014-84-K-0415 and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 5404, monitored by the Office of Naval Research under the same contract. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. Government.

we write it rather as an implication $A \Rightarrow B$. We assume known the logical constructions ∇ (absurdity), $\wedge$ (conjunction), $+$ (intuitionistic disjunction), $\vee$ (classical disjunction) and $\exists$ (existential quantification).

We use the symbol $:=$ for definitional equality of constants.

1 A constructive set theory

We assume a global context where we have declared: $[U : Type]$. We may think of U as the current *universe*, or as the domain of interpretation, in the sense of predicate calculus. *Sets* defined over U are represented as predicates of type $U \to Prop$, which we abbreviate from now on as Set_U, or even as Set when U is clear from the context. This may be formally justified by the type synthesis algorithm described in [11]. If $A : Set_U$ and $x : U$, we define $x \in A$ as the proposition $(A\ x)$. That is, the *elements* of U-sets are of type U. We abbreviate the quantification $(x : U)E$ as $\forall x \cdot E$, and the abstraction $[x : U]\,E$ as $\{x \mid E\}$. For successive bindings, we use respectively $\forall x, y \cdot E$ and $\{x, y \mid E\}$.

We define inclusion of sets A and B by:

$$A \subseteq B \ := \ \forall x \cdot x \in A \Rightarrow x \in B$$

and set equality by:

$$A = B \ := \ A \subseteq B \wedge B \subseteq A.$$

Note that this equality is extensional equality of sets. We could also define intensional equality between two elements x and y of type U, as:

$$x \equiv y \ := \ (A : Set)\ x \in A \Rightarrow y \in A.$$

If we decided to give a primitive equality $=$ on type U this would complicate matters quite a bit, since we would have to state that sets are predicates compatible with this equality, i.e. such that $(P\ x)$ and $x = y$ imply $(P\ y)$, and iterate this condition with classes, etc...

The empty U-set is defined as:

$$\emptyset \ := \ \{x \mid \nabla\}.$$

The usual set operations are available through the corresponding logical connectives:

$$A \cap B \ := \ \{x \mid x \in A \wedge x \in B\}$$

$$A \cup B \ := \ \{x \mid x \in A \vee x \in B\}$$

$$\sim A \ := \ \{x \mid \neg x \in A\}.$$

Remark. If we were completely formal, we should index all our notations with U, and write for instance $x \in_U A$, $\emptyset_U$, etc. We assume here no ambiguity arises as to which universe we are into.

Our sets resemble ordinary sets, except that the inclusion relation is defined constructively. Thus, we have $A \subseteq \sim\sim A$, but the converse in not true in general. That is, our sets behave more like open sets of a topological space, and classical sets are the analogue of closed sets, i.e. double-negation closed. The complement $\sim A$ of A is closed, and for every A we get its closure as $\sim\sim A$.

We now define classes as set predicates. That is, a U-class is of type $(Set_U \to Prop)$, abbreviated $Class_U$ or simply $Class$. For instance, the class of subsets of A is defined as:

$$(\mathcal{P}\ A) \ := \ [B : Set]\ B \subseteq A.$$

Class inclusion may be defined in the same way as set inclusion. Actually, all sets operations above extend to class operations, since U may be instantiated with Set_U.

If C is a U-class, we define the *intersection* of C as the U-set defined as follows:

$$\cap C \ := \ \{x \mid (A : Set)\ (C\ A) \Rightarrow x \in A\}.$$

For instance, we may define the singleton $\{x\}$ as follows:

$$\{x\} \ := \ \cap([A : Set]\ x \in A).$$

We say that set A is *universal* if it contains all the objects of the universe:

$$(Universal\ A) \ := \ \forall x \cdot x \in A.$$

A *mapping* maps a set to a set. More precisely, a U-map has type $Set_U \rightarrow Set_U$, abbreviated Map_U or simply Map. If φ is a U-map, we define:

$$(Stable\ \varphi) \ := \ [A : Set](\varphi\ A) \subseteq A$$

and

$$(Fixpt\ \varphi) \ := \ [A : Set](\varphi\ A) = A.$$

Note that these two constructions are of type $Class_U$. We now define the standard interpretation of map φ as the intersection of the class of sets for which φ is stable:

$$(Initial\ \varphi) \ := \ \cap(Stable\ \varphi)$$

that is, in expanded form:

$$\{u \mid (A : Set)(\forall x \cdot x \in (\varphi\ A) \Rightarrow x \in A) \Rightarrow u \in A\}.$$

2 Induction

We get an induction principle by restricting ourselves to the standard interpretation:

$$(Induction\ \varphi) \ := \ (A : Set)(Stable\ \varphi\ A) \Rightarrow (Universal\ A),$$

or, in an equivalent expanded formulation:

$$(A : Set)(\forall x \cdot x \in (\varphi\ A) \Rightarrow x \in A) \Rightarrow \forall u \cdot u \in A.$$

Note that $Initial$ and $Induction$ are really the same construction, up to permutation of independent hypotheses: the binding on u migrated from the outermost abstraction in $(Initial\ \varphi)$ to the innermost quantification in $(Induction\ \varphi)$.

This notion is especially important when φ is an *increasing* map:

$$(Incr\ \varphi) \ := \ (A : Set)(B : Set)\ A \subseteq B \Rightarrow (\varphi\ A) \subseteq (\varphi\ B)$$

since then we may apply Tarski's theorem, and thus consider $(Initial\ \varphi)$ as the least solution to φ considered as a recursive definition. Let us check out the details.

Tarski's theorem may be stated in the constructions calculus as follows. Consider the following context Γ:
$[T : Type]$
$[Eq : T \rightarrow T \rightarrow Prop]$
$[Leq : T \rightarrow T \rightarrow Prop]$

$[Leqtrans : (x : T)(y : T)(z : T)(Leq\ x\ y) \Rightarrow (Leq\ y\ z) \Rightarrow (Leq\ x\ z)]$
$[Leqantisym : (x : T)(y : T)(Leq\ x\ y) \Rightarrow (Leq\ y\ x) \Rightarrow (Eq\ x\ y)]$
$[Lub : (T \to Prop) \to T]$
$[Upperb : (P : T \to Prop)(y : T)(P\ y) \Rightarrow (Leq\ y\ (Lub\ P))]$
$[Least : (P : T \to Prop)(y : T)((z : T)(P\ z) \Rightarrow (Leq\ z\ y)) \Rightarrow (Leq\ (Lub\ P)\ y)]$
$[f : T \to T]$
$[Incr : (x : T)(y : T)(Leq\ x\ y) \Rightarrow (Leq\ (f\ x)\ (f\ y))]$.

In the context Γ, we consider the set P_0 defined as:

$$P_0\ :=\ [u : T](Leq\ u\ (f\ u)),$$

and its least upper bound $x_0 : T$:

$$x_0\ :=\ (Lub\ P_0).$$

We first prove a few lemmas. We leave it to the reader to check that

$$\Gamma \vdash Lemma_1 : (x : T)(Leq\ x\ (f\ x)) \Rightarrow (Leq\ x\ (f\ x_0)),$$

with:

$$Lemma_1\ :=\ [x : T]\,[h : (Leq\ x\ (f\ x))]\,(Leqtrans\ x\ (f\ x)\ (f\ x_0)\ h\ (Incr\ x\ x_0\ (Upperb\ P_0\ x\ h))).$$

Similarly, we get $\Gamma \vdash Lemma_2 : (Leq\ x_0\ (f\ x_0))$, with

$$Lemma_2\ :=\ (Least\ P_0\ (f\ x_0)\ Lemma_1),$$

and also $\Gamma \vdash Lemma_3 : (Leq\ (f\ x_0)\ x_0)$, with

$$Lemma_3\ :=\ (Upperb\ P_0\ (f\ x_0)\ (Incr\ x_0\ (f\ x_0)\ Lemma_2)).$$

Now the proof is concluded easily, that is $\Gamma \vdash Tarski : (Eq\ (f\ x_0)\ x_0))$, with

$$Tarski\ :=\ (Leqantisym\ (f\ x_0)\ x_0\ Lemma_2\ Lemma_3).$$

The careful reader will check that this is the traditional proof of Tarski's theorem [20]. We may now use Tarski's theorem in the particular case of the subset relation. That is, we instantiate the type variable T with Set, $(Leq\ x\ y)$ becomes $y \subseteq x$, and Eq is set equality. The hypotheses $Leqtrans$ and $Leqantisym$ are easy to fulfill. We take for Lub the intersection operation, for which it is immediate to show $Upperb$ and $Least$. Note that it is essential here that Tarski's theorem be expressed over an arbitrary type. This allows us to instantiate T over the type of sets given with the inclusion relation, obtaining thus what is usually called the theorem of Knaster-Tarski.

Hence we get:

$$(\varphi : Map)(Incr\ \varphi) \Rightarrow (Fixpt\ \varphi\ (Initial\ \varphi)). \qquad (FIX)$$

Actually it is possible to refine Tarski's theorem and prove that the fixpoint obtained in the proof is actually the Lub of the set of all fixpoints. Here this shows that $(Initial\ \varphi)$ is the smallest fixpoint:

$$(\varphi : Map)(Incr\ \varphi) \Rightarrow (Initial\ \varphi) = \cap(Fixpt\ \varphi). \qquad (MIN)$$

Note the similarity of our approach with the treatment in Park[16], where $Induction$ is called a *convergence* formula.

3 Nœtherian induction

Let us use the abbreviation Rel_U, or simply Rel, for the type $U \to U \to Prop$. Rel_U is the type of binary relations on U.

Note that every relation may be seen as an indexed family of sets. Thus if $\geq$ is a preorder, $(\geq\ x)$ is the set of elements below x.

Let $R : Rel$. We define the *adjoint* map associated with R as the U-map:

$$(Adjoint\ R)\ :=\ [A : Set]\ \{x \mid (R\ x) \subseteq A\}.$$

It is a simple exercise (left to the reader) to prove that this map is always increasing:

$$(R : Rel)\ (Incr\ (Adjoint\ R)) \qquad\qquad (Adjoint\_Incr).$$

The class of *R-inductive* sets is defined as:

$$(Inductive\ R)\ :=\ (Stable\ (Adjoint\ R)).$$

The induction associated with the adjoint map states that the inductive sets are universal. This is just what is usually called Nœtherian induction[5]:

$$(Noetherian\ R)\ :=\ (Induction\ (Adjoint\ R))$$

or, in expanded form:

$$(A : Set)\ (\forall x \cdot (\forall y \cdot (R\ x\ y) \Rightarrow y \in A) \Rightarrow x \in A) \Rightarrow \forall u \cdot u \in A.$$

We recognize the definition used in [9] to prove Newman's lemma:

$$(R : Rel)\ (Noetherian\ R) \Rightarrow (Loc\_Confluent\ R) \Rightarrow (Confluent\ R).$$

Thus we see that this very powerful transfinite induction principle is but a special case of the very general *Induction* above. Usual complete induction principles are in turn obtained by further specialization. For instance, we shall see below that complete induction over the naturals is simply $(Noetherian\ >)$.

4 Structural Induction

It is now time to introduce some further notation. Let A be a U-set, and E be any construction expression. We let $\forall x \in A \cdot E$ to stand for an abbreviation of $\forall x \cdot x \in A \Rightarrow E$, and similarly we let $\{x \in A \mid E\}$ stand for $\{x \mid x \in A \wedge E\}$. We shall also use the notation $\exists x \in A \cdot E$ to stand for $\exists [x : U](x \in A \wedge E)$.

We shall now show how to express structural induction[4] in the calculus. First we define the relation "f preserves A", when f is a U-function (i.e. $f : U \to U$) and A is a U-set:

$$(Preserve\ f\ A)\ :=\ \forall x \in A \cdot (f\ x) \in A.$$

Now we define what it means for the element $y : U$ to be *reachable* from element $x : U$ using function f. This notion is axiomatized by the relation:

$$(Iter\ f)\ :=\ \{x, y \mid (A : Set)\,(Preserve\ f\ A) \Rightarrow x \in A \Rightarrow y \in A\}.$$

The general method consists in defining, in the structure under consideration, the suitable generalization of reachability expressing what are the elements *expressible* using the operations of the structure. The expressibility predicate may then be seen as a set (the initial algebra, or standard model). Similarly to above we get an induction principle by postulating that every element of type U is in this set.

Let us consider for instance arithmetic. The structure is given here by a successor operation $S : U \to U$ and a zero constant $0 : U$. A universe presented with this structure we call a *Peano algebra*. On any Peano algebra, we may define a relation

$$\leq \ := \ (Iter\ S)$$

and it is easy to show that $\leq$ is reflexive and transitive (using as proofs respectively identity and composition of the proper type). Now the set

$$\{n \mid 0 \leq n\}$$

is the characteristic predicate of the standard model $\mathcal{N}$:

$$\mathcal{N} \ := \ \{n \mid (A : Set)\ (\forall m \in A \cdot (S\ m) \in A) \Rightarrow 0 \in A \Rightarrow n \in A\}.$$

The corresponding universally quantified sentence is Peano's induction principle:

$$Peano \ := \ (A : Set)\ (\forall m \in A \cdot (S\ m) \in A) \Rightarrow 0 \in A \Rightarrow \forall n \cdot n \in A.$$

Note how the binding on n migrated from $\mathcal{N}$ to $Peano$, similarly to the transformation between $Initial$ and $Induction$.

Let us now indicate the relationship with our general induction principle above. The map φ needed here may be defined as sending A to: $\{n \mid \exists m \in A \cdot n \equiv (S\ m) + n \equiv 0\}$, or equivalently, we define:

$$(Nat\_map\ A) \ := \ \{n \mid (P : Set)\ \forall u \cdot (u \in A \Rightarrow (S\ u) \in P) \Rightarrow 0 \in P \Rightarrow n \in P\}.$$

It is easy to prove that Nat_map is increasing, and that:

$$(Stable\ Nat\_map\ A) \Leftrightarrow (Nat\_stable\ A)$$

with

$$(Nat\_stable\ A) \ := \ (\forall n \in A \cdot (S\ n) \in A) \wedge (0 \in A)$$

and thus we get that

$$(Induction\ Nat\_map) \Leftrightarrow Peano.$$

Indeed, a simple Curryfication suffices to show that:

$$\mathcal{N} = \cap Nat\_stable.$$

Remark. The equivalence between $(Stable\ Nat\_map)$ and Nat_stable boils down to recognizing the following propositional equivalence:

$$(Q : Prop)((P \Rightarrow Q) \Rightarrow Q) \Leftrightarrow P.$$

Intuitively, it means that every proposition is equivalent to its operational contents.

Actually $Peano$ is only one half of initiality. What it says is that the universe contains only elements which are definable with the algebra operators. The other half is to postulate that different operators give rise to distinct elements. In the case of arithmetic, for instance, this amounts to adding the following two postulates:

$$Peano1 \ := \ \forall n \cdot \neg(S \ n) \equiv 0$$

$$Peano2 \ := \ \forall m, n \cdot (S \ m) \equiv (S \ n) \Rightarrow m \equiv n.$$

A standard model of arithmetic is thus any universe verifying $Peano$, $Peano1$ and $Peano2$.

Remark. We recall that the natural numbers may be expressed logically by the second-order proposition:

$$Nat \ := \ (X : Prop)(X \Rightarrow X) \Rightarrow X \Rightarrow X,$$

with the successor function, of type $Nat \Rightarrow Nat$, defined as:

$$S \ := \ [n : Nat]\,[X : Prop]\,[s : X \Rightarrow X]\,[z : X](s \ (n \ X \ s \ z))$$

and the zero, of type Nat, defined as:

$$0 \ := \ [X : Prop]\,[s : X \Rightarrow X]\,[z : X]\,z.$$

It is possible to apply the whole theory above, with Nat standing for the universe U. However, even in Nat we need to postulate the $Peano$ axioms. This is a bit puzzling, since we know that the normal forms of constructions (with η conversion allowed) of type Nat are isomorphic to the standard model of natural numbers. But this knowledge is from meta-theoretic analysis, and cannot be internalized in the system. However, it is a simple matter to define in the meta-language of constructions appropriate macros, so that the Peano axioms are automatically generated from the signature Nat.

The method above is of course generalizable in a straightforward way to any algebraic type, leading to structural induction over a wide variety of structures.

Finally, complete induction is easily seen a direct application of Nœtherian induction. For instance, over integers, with

$$x > y \ := \ (S \ y) \leq x$$

we get complete induction (course-of-values induction) as $(Noetherian \ >)$.

5 Computational Induction

We now show how to imbed in Constructions Scott's computational induction method, as presented for instance in LCF[13].

5.1 The domain postulates

We assume axioms on the universe U giving it the structure of a pre-ordering:

$$[\sqsubseteq : Rel]$$

$$[Refl : \forall u \cdot u \sqsubseteq u]$$

$$[Trans : \forall u, v, w \cdot u \sqsubseteq v \Rightarrow v \sqsubseteq w \Rightarrow u \sqsubseteq w].$$

We define $\doteq$ as the associated equivalence:

$$\doteq := \ \forall u, v \cdot u \sqsubseteq v \wedge v \sqsubseteq u.$$

We say that the U-set A is *directed* whenever:

$$(Directed\ A) \ := \ \forall x \in A \cdot \forall y \in A \cdot \exists z \in A \cdot x \sqsubseteq z \wedge y \sqsubseteq z.$$

Now we postulate the partial order U to be *complete*, in the sense that every directed set possesses a limit, its lub: $(A : Set)\ (Directed\ A) \Rightarrow \exists u \cdot u \in (Lub\ A)$, with:

$$(Lub\ A) \ := \ \{u \mid \forall x \in A \cdot x \sqsubseteq u \wedge \forall v \cdot (\forall x \in A \cdot x \sqsubseteq v) \Rightarrow u \sqsubseteq v\}.$$

For ease of application, we shall Skolemize the limit u as a function $(lim\ A)$. We could have lim depend on an extra argument of type $(Directed\ A)$, but this extra generality is not needed; this is an application of the principle of "proof irrelevance". Thus we postulate:

$$[lim : Set_U \rightarrow U]$$

$$[Complete : (A : Set)\ (Directed\ A) \Rightarrow (lim\ A) \in (Lub\ A)].$$

It is easy to show that the elements of $(Lub\ A)$ are equivalent:

$$\forall u \in (Lub\ A) \cdot \forall v \in (Lub\ A) \cdot u \doteq v.$$

The empty set $\emptyset$ is directed, and thus every complete pre-order possesses a minimum element:

$$\bot \ := \ (lim\ \emptyset).$$

It is straightforward to prove that $\bot$ is indeed minimum:

$$\forall u \cdot \bot \sqsubseteq u. \qquad (Bot)$$

5.2 Computational induction

Let $f : U \rightarrow U$. We define the set of (finite) *f-approximants* as:

$$(Approx\ f) \ := \ (Iter\ f\ \bot)$$

that is:

$$\{u \mid (A : Set)\ (Preserve\ f\ A) \Rightarrow \bot \in A \Rightarrow u \in A\}.$$

Remark the similarity with the definition of the standard model $\mathcal{N}$ above. Similarly to maps above, we define the notion of *increasing* function:

$$(Increasing\ f) \ := \ \forall u, v \cdot u \sqsubseteq v \Rightarrow (f\ u) \sqsubseteq (f\ v)$$

and we may show that:

$$(f : U \rightarrow U)\ (Increasing\ f) \Rightarrow (Directed\ (Approx\ f)). \qquad (Dir\_Approx)$$

The proof of this proposition, left as an exercise, is analogous to defining inductively the function computing the maximum of two natural numbers. We may now define, for any increasing f:

$$(Y\ f) \ := \ (lim\ (Approx\ f)).$$

The limit of finite approximants $(Y\ f)$ is intuitively $\sqcup_n\ f^n(\bot)$.

We now define an *admissible* U-set as one which contains all the limits of its directed subsets:

$$(Adm\ A)\ :=\ (B : Set)\ B \subseteq A \Rightarrow (Directed\ B) \Rightarrow (lim\ B) \in A.$$

The restriction of A to admissible sets in the definition of approximant permits to iterate f in the transfinite, which gives the notion of transfinite f-approximation:

$$(\infty\ f)\ :=\ \{u \mid (A : Set)\ (Adm\ A) \Rightarrow (\forall x \in A \cdot (f\ x) \in A) \Rightarrow u \in A\}.$$

Note that $(\infty\ f)$ is the intersection of the class of admissible sets preserved by f, whereas $(Approx\ f)$ is the intersection of the class of sets containing $\bot$ and preserved by f. In some sense $(\infty\ f)$ is to $(Approx\ f)$ what ordinals are to natural numbers.

Let us now show that $(\infty\ f)$ is admissible:
$[f : U \to U]\ [B : Set]\ [h_1 : B \subseteq (\infty\ f)]\ [h_2 : (Directed\ B)] \vdash l_1 : (lim\ B) \in (\infty\ f)$
where l_1 is proved by:
$[C : Set]\ [h_3 : (Adm\ C)]\ [h_4 : (Stable\ C)](h_3\ B\ l_2\ h_2)$
where $l_2 : B \subseteq C$ is proved by:
$[u : U]\ [h_5 : u \in B](h_1\ u\ h_5\ C\ h_3\ h_4)$.
Discharging all this temporary context, we get:

$$(f : U \to U)(Adm\ (\infty\ f)). \qquad (Adm\_\infty)$$

Now it is a simple matter to prove:
$[f : U \to U]\ [i : (Increasing\ f)] \vdash (Adm\_\infty\ f\ (Approx\ f)\ incl\ (Dir\_Approx\ f)) : (Y\ f) \in (\infty\ f)$,
where the proof of $incl : (Approx\ f) \subseteq (\infty\ f)$ is left to the reader. Thus we get finally:

$$(f : U \to U)(Increasing\ f) \Rightarrow (Y\ f) \in (\infty\ f).$$

By unwinding this proposition it is easy to see that this is precisely Scott's computational induction principle. Writing it in long form:

$$(f : U \to U)(Increasing\ f) \Rightarrow (A : Set)\ (Adm\ A) \Rightarrow (\forall x \in A \cdot (f\ x) \in A) \Rightarrow (Y\ f) \in A. \qquad (Comp\_Ind)$$

Two remarks are in order. Firstly, note that this principle is *provable* from our postulates on the domain U (i.e., the complete partial ordering axioms). Secondly, the notion of admissible set is axiomatized inside the calculus, and thus we can use all the power of the logical system to prove that a given set is indeed admissible, whereas in LCF[13] the notion of admissible predicate is weakened to a syntactic check of the meta-linguistic support. Finally note that the hypothesis $\bot \in A$ is not needed above, since it is implicit from the hypothesis $(Adm\ A)$.

5.3 Continuity and fixpoints

It may seem curious that it is not necessary in the justification of computational induction to assume that f is continuous. But this assumption is indeed needed for recursion. Let us now make this point precise.

First, let us define the image by f of a U-set A:

$$(Image\ f\ A)\ :=\ \{y \mid \exists x \in A \cdot y \doteq (f\ x)\}.$$

It is easy to show that:

$$(Directed\ A) \Rightarrow (Increasing\ f) \Rightarrow (Directed\ (Image\ f\ A)). \qquad (Dir\_Im)$$

Thus, for every increasing f and directed set A, we may define:

$$(Lim\ f\ A)\ :=\ (lim\ (Image\ f\ A)).$$

Now let us call *diagram* any non-empty directed set:

$$(Diagram\ A)\ :=\ \exists u \in A \cdot (Directed\ A).$$

Next we define what it means for an increasing f to be continuous:

$$(Continuous\ f)\ :=\ (A : Set)\ (Diagram\ A) \Rightarrow (Lim\ f\ A) \doteq (f\ (lim\ A)).$$

Note that we must restrict A to be a non-empty directed set, since we do not demand our functions to be strict.

Exercise. Prove that for all f, $(Continuous\ f) \Rightarrow (Increasing f)$.

Now, defining the fixpoints of f in a similar way as for maps:

$$(Fixpoints\ f)\ :=\ \{u \mid (f\ u) \doteq u\},$$

we can prove:

$$(f : U \to U)(Continuous\ f) \Rightarrow (Y\ f) \in (Fixpoints\ f)$$

and:

$$(f : U \to U)(Continuous\ f) \Rightarrow \forall x \in (Fixpoints\ f) \cdot (Y\ f) \sqsubseteq x.$$

In other words, $(Y\ f) \in (Min\ (Fixpoints\ f))$, with

$$(Min\ A)\ :=\ \{u \in A \mid \forall x \in A \cdot u \sqsubseteq x\}.$$

This is analogous to Tarski's theorem, but still significantly different.

A variant of Tarski's theorem would say here is that if f is increasing (and not necessarily continuous), then $(Z\ f)$ is the minimum fixpoint of f, where

$$(Z\ f)\ :=\ (lim\ (\infty\ f)).$$

Problem. Show the above statement. In particular, you will need to prove that $(\infty\ f)$ is itself directed.

Thus, continuity is needed for finiteness, i.e. computability. This concludes our incursion into domain theory.

6 Nœtherian as a well-foundedness principle

We are going to show in this section that $(Noetherian\ R)$ implies that there are no infinite R-chains, relating induction to well-foundedness.

Let A be a U-set. We say that A is *R-eternal* iff:

$$(Eternal\ R\ A)\ :=\ \exists x \in A \wedge \forall x \in A \cdot \exists y \in A \cdot (R\ x\ y).$$

It is straightforward to show, with the definition of Nœtherian given above, that:

$$(Eternal\ R\ A)\ |\ (Noetherian\ R\ \sim A)$$

where the incompatibility connective | is Sheffer's stroke.

Thus (*Noetherian* R) implies (classically) that R is *well-founded*, in the sense that there are no infinite R-chains:

$$(WFR)\ :=\ (A : Set)\ \neg(Eternal\ R\ A).$$

Intuitively, the set $(Initial\ (Adjoint\ R))$ contains all elements which have only finite R-chains issued from them, and (*Noetherian* R) says that this set is universal.

Acknowledgements

We thank Thierry Coquand for many stimulating discussions.

References

[1] R. Boyer, J Moore. "A Computational Logic." Academic Press (1979).

[2] N.G. de Bruijn. "Automath a language for mathematics." Les Presses de l'Université de Montréal, (1973).

[3] N.G. de Bruijn. "A survey of the project Automath." (1980) in to H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Eds Seldin J. P. and Hindley J. R., Academic Press (1980).

[4] R. Burstall. "Proving Properties of Programs by Structural Induction." Comp. J. **12** (1969), 41–48.

[5] P. M. Cohn. "Universal Algebra." Reidel, 1965.

[6] R.L. Constable, N.P. Mendler. "Recursive Definitions in Type Theory." Private Communication (1985).

[7] Th. Coquand. "Une théorie des constructions." *Thèse de troisième cycle,* Université Paris VII, Janvier 85.

[8] Th. Coquand. "An analysis of Girard's paradox." First IEEE Symposium on Logic in Computer Science, Boston (June 1986), 227–236.

[9] Th. Coquand and G. Huet. "Constructions: A Higher Order Proof System for Mechanizing Mathematics." *EUROCAL85,* Linz, Springer-Verlag LNCS 203 (1985).

[10] Th. Coquand and G. Huet. "The Calculus of Constructions." To appear, Information and Control.

[11] Th. Coquand and G. Huet. "Concepts Mathématiques et Informatiques Formalisés dans le Calcul des Constructions." Logic Colloquium, Orsay (July 85). To appear, North-Holland.

[12] J.Y. Girard. "Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieure." Thèse d'Etat, Université Paris VII (1972).

[13] M. Gordon, R. Milner and C. Wadsworth. "Edinburgh LCF." Springer-Verlag LNCS 78 (1979).

[14] P. Martin-Löf. "A theory of types." Report 71-3, Dept. of Mathematics, University of Stockholm, Feb. 1971, revised (Oct. 1971).

[15] N.P. Mendler. "First and Second-Order Lambda Calculi with Recursive Types." Technical Report TR 86-764, Dept. of Computer Science, Cornell University (July 1986).

[16] D. Park. "Fixpoint Induction and Proofs of Program Properties." *Machine Intelligence 5*, Eds. B. Meltzer & D. Michie, 59–77, Edinburgh University Press.

[17] L. C. Paulson. "Constructing Recursion Operators in Intuitionistic Type Theory." Tech. Report 57, Computer Laboratory, University of Cambridge (Oct. 1984). To appear, J. of Symbolic Computation.

[18] D. Scott. "Constructive validity." Symposium on Automatic Demonstration, Springer-Verlag Lecture Notes in Mathematics, **125** (1970).

[19] D. Scott. "Data Types as Lattices." SIAM Journal of Computing **5** (1976) 522–587.

[20] A. Tarski. "A Lattice-Theoretical Fixpoint Theorem and its Applications." *Pacific J. Math.* **5** (1955), 285–309.