CHAPTER 3

Extracting computational content from proofs

3.1. A theory of computable functionals

3.1.1. Brouwer-Heyting-Kolmogorov and Gödel. The Brouwer-Heyting-Kolmogorov interpretation (BHK-interpretation for short) of intuitionistic (and minimal) logic explains what it means to prove a logically compound statement in terms of what it means to prove its components; the explanations use the notions of *construction* and *constructive proof* as unexplained primitive notions. For prime formulas the notion of proof is supposed to be given. The clauses of the BHK-interpretation are:

(i) p proves $A \wedge B$ if and only if p is a pair $\langle p_0, p_1 \rangle$ and p_0 proves A, p_1 proves B;
(ii) p proves $A \to B$ if and only if p is a construction transforming any proof q of A into a proof $p(q)$ of B;
(iii) $\bot$ is a proposition without proof;
(iv) p proves $\forall_{x \in D} A(x)$ if and only if p is a construction such that for all $d \in D$, $p(d)$ proves $A(d)$;
(v) p proves $\exists_{x \in D} A(x)$ if and only if p is of the form $\langle d, q \rangle$ with d an element of D, and q a proof of $A(d)$.

The problem with the BHK-interpretation clearly is its reliance on the unexplained notions of construction and constructive proof. Gödel was concerned with this problem for more than 30 years. In 1941 he gave a lecture at Yale university with the title "In what sense is intuitionistic logic constructive?". According to Kreisel, Gödel "wanted to establish that intuitionistic proofs of existential theorems provide explicit realizers" (Feferman et al., 1986, 1990, 1995, 2002, 2002, Vol II, p.219). Gödel published his "Dialectica interpretation" in (1958), and revised this work over and over again; its state in 1972 has been published in the same volume. Troelstra, in his introductory note to the latter two papers writes (loc. cit., pp.220/221):

> Gödel argues that, since the finististic methods considered are not sufficient to carry out Hilbert's program, one has

> to admit at least some abstract notions in a consistency proof; ... However, Gödel did not want to go as far as admitting Heyting's abstract notion of constructive proof; hence he tried to replace the notion of constructive proof by something more definite, less abstract (that is, more nearly finitistic), his principal candidate being a notion of "computable functional of finite type" which is to be accepted as sufficiently well understood to justify the axioms and rules of his system T, an essentially logic-free theory of functionals of finite type.

We intend to utilize the notion of a computable functional of finite type as an ideal in an information system, as explained in the previous chapter. However, Gödel noted that his proof interpretation is largely independent of a precise definition of computable functional; one only needs to know that certain basic functionals are computable (including primitive recursion operators in finite types), and that they are closed under composition. Building on Gödel (1958), we assign to every formula A a new one $\exists_x A_1(x)$ with $A_1(x)$ $\exists$-free. Then from a derivation of A we want to extract a "realizing term" r such that $A_1(r)$. Of course its meaning should in some sense be related to the meaning of the original formula A. However, Gödel explicitly states in (1958, p.286) that his Dialectica interpretation is *not* the one intended by the BHK-interpretation.

3.1.2. Formulas and predicates. When we want to make propositions about computable functionals and their domains of partial continuous functionals, it is perfectly natural to take, as initial propositions, ones formed inductively or coinductively. However, for simplicity we omit the treatment of coinductive definitions and deal with inductive definitions only. For example, in the algebra $\mathbf{N}$ we can inductively define *totality* by the clauses

$$T_{\mathbf{N}}0, \qquad \forall_n(T_{\mathbf{N}}n \to T_{\mathbf{N}}(\mathrm{S}n)).$$

Its least-fixed-point scheme will now be taken in the form

$$\forall_n(T_{\mathbf{N}}n \to A(0) \to \forall_n(T_{\mathbf{N}}n \to A(n) \to A(\mathrm{S}n)) \to A(n)).$$

The reason for writing it in this way is that it fits more conveniently with the logical elimination rules, which will be useful in the proof of the soundness theorem. It expresses that every "competitor" $\{\, n \mid A(n) \,\}$ satisfying the same clauses contains $T_{\mathbf{N}}$. This is the usual induction schema for natural numbers, which clearly only holds for "total" numbers (i.e., total ideals

in the information system for $\mathbf{N}$). Notice that we have used a "strengthened" form of the "step formula", namely $\forall_n(T_{\mathbf{N}}n \to A(n) \to A(\mathrm{S}n))$ rather than $\forall_n(A(n) \to A(\mathrm{S}n))$. In applications of the least-fixed-point axiom this simplifies the proof of the "induction step", since we have the additional hypothesis $T(n)$ available. Totality for an arbitrary algebra can be defined similarly. Consider for example the non-finitary algebra $\mathbf{O}$ (cf. 2.1.3), with constructors 0, successor S of type $\mathbf{O} \to \mathbf{O}$ and supremum Sup of type $(\mathbf{N} \to \mathbf{O}) \to \mathbf{O}$. Its clauses are

$$T_{\mathbf{O}}0, \qquad \forall_x(T_{\mathbf{O}}x \to T_{\mathbf{O}}(\mathrm{S}x)), \qquad \forall_f(\forall_{n\in T_{\mathbf{N}}}T_{\mathbf{O}}(fn) \to T_{\mathbf{O}}(\mathrm{Sup}(f))),$$

and its least-fixed-point scheme is

$$\begin{aligned}\forall_x(T_{\mathbf{O}}x \to A(0) \to\ & \\ & \forall_x(T_{\mathbf{O}}x \to A(x) \to A(\mathrm{S}x)) \to \\ & \forall_f(\forall_{n\in T}T_{\mathbf{O}}(fn) \to \forall_{n\in T}A(fn) \to A(\mathrm{Sup}(f))) \to \\ & A(x)).\end{aligned}$$

Generally, an inductively defined predicate I is given by k clauses, which are of the form

$$K_i := \forall_{\vec{x}}((A_\nu(I))_{\nu<n} \to I\vec{r}) \quad (i < k).$$

Our formulas will be defined by the operations of implication $A \to B$ and universal quantification $\forall_{x^\rho}A$ from inductively defined predicates $\mu_X\vec{K}$, where X is a "predicate variable", and the K_i are "clauses". Every predicate has an *arity*, which is a possibly empty list of types.

Definition (Formulas and predicates). By simultaneous induction we define formula forms

$$A, B ::= P\vec{r} \mid A \to B \mid \forall_x A$$

and predicate forms

$$P, Q ::= X \mid \{\, \vec{x} \mid A \,\} \mid \mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to X\vec{r}_i))_{i<k}$$

with X a predicate variable, $k \geq 1$ and $\vec{x}_i$ all free variables in $(A_{i\nu})_{\nu<n_i} \to X\vec{r}_i$ (it is not necessary to allow object parameters in inductively defined predicates, since they can be taken as extra arguments). Let C denote both formula and predicate forms. Let $\mathrm{FPV}(C)$ denote the set of free predicate variables in C. We define $\mathrm{SP}(Y, C)$ "Y occurs at most strictly positive in C" by induction on C.

$$\frac{\mathrm{SP}(Y,P)}{\mathrm{SP}(Y,P\vec{r})} \qquad \frac{Y \notin \mathrm{FPV}(A) \quad \mathrm{SP}(Y,B)}{\mathrm{SP}(Y, A \to B)} \qquad \frac{\mathrm{SP}(Y,A)}{\mathrm{SP}(Y,\forall_x A)}$$

$$\mathrm{SP}(Y,X) \qquad \frac{\mathrm{SP}(Y,A)}{\mathrm{SP}(Y,\{\,\vec{x}\mid A\,\})} \qquad \frac{\mathrm{SP}(Y,A_{i\nu}) \text{ for all } i{<}k,\ \nu{<}n_i}{\mathrm{SP}(Y,\mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i}\to X\vec{r}_i))_{i<k})}$$

Now we can define $\mathrm{F}(A)$ "A is a formula" and $\mathrm{Pred}(P)$ "P is a predicate", again by simultaneous induction.

$$\frac{\mathrm{Pred}(P)}{\mathrm{F}(P\vec{r})} \qquad \frac{\mathrm{F}(A)\quad \mathrm{F}(B)}{\mathrm{F}(A\to B)} \qquad \frac{\mathrm{F}(A)}{\mathrm{F}(\forall_x A)}$$

$$\mathrm{Pred}(X) \qquad \frac{\mathrm{F}(A)}{\mathrm{Pred}(\{\,\vec{x}\mid A\,\})}$$

$$\frac{\mathrm{F}(A_{i\nu}) \text{ and } \mathrm{SP}(X,A_{i\nu}) \text{ for all } i{<}k,\ \nu{<}n_i \qquad X\notin \mathrm{FPV}(A_{0\nu}) \text{ for all } \nu{<}n_0}{\mathrm{Pred}(\mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i}\to X\vec{r}_i))_{i<k})}$$

We call

$$I := \mu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i}\to X\vec{r}_i))_{i<k}$$

an inductive (or inductively defined) predicate. Sometimes it is helpful to display the predicate parameters and write $I(\vec{Y},\vec{Z}\,)$, where $\vec{Y},\vec{Z}$ are all predicate variables free in some $A_{i\nu}$ except X, and $\vec{Y}$ are the ones occuring only strictly positive. If we write the i-th component of I in the form $\forall_{\vec{x}}((A_\nu(X))_{\nu<n}\to X\vec{r}\,)$, then we call

$$K_i := \forall_{\vec{x}}((A_\nu(I))_{\nu<n}\to I\vec{r}\,) \tag{3.1}$$

the i-th *clause* (or *introduction axiom*) of I, denoted I_i^+.

Here $\vec{A}\to B$ means $A_0\to\cdots\to A_{n-1}\to B$, associated to the right. The terms $\vec{r}$ are those introduced in section 2.2, i.e., typed terms built from variables and constants by abstraction and application, and (importantly) those with a common reduct are identified. In $\forall_{\vec{x}}((A_\nu(X))_{\nu<n}\to X\vec{r}\,)$ we call $A_\nu(X)$ a *parameter* premise if X does not occur in it, and a *recursive* premise otherwise. A recursive premise $A_\nu(X)$ is *nested* if it has an occurrence of X in a strictly positive parameter position of another (previously defined) inductive predicate, and unnested otherwise. An inductive predicate I is called *nested* if it has a clause with at least one nested recursive premise, and *unnested* otherwise.

A predicate of the form $\{\,\vec{x}\mid C\,\}$ is called a *comprehension term*. We identify $\{\,\vec{x}\mid C(\vec{x}\,)\,\}\vec{r}$ with $C(\vec{r}\,)$. The letter I will be used for predicates of the form $\mu_X(K_0,\dots,K_{k-1})$; they are called *inductively defined predicates*. An inductively defined predicate is *finitary* if its clauses have recursive premises of the form $X\vec{s}$ only.

Definition (Theory of computable functionals, TCF). TCF is the system in minimal logic for $\to$ and $\forall$, whose formulas are those in F above, and

whose axioms are the following. For each inductively defined predicate, there are "closure" or introduction axioms, together with a "least-fixed-point" or elimination axiom. In more detail, consider an inductively defined predicate $I := \mu_X(K_0, \dots, K_{k-1})$. For each of the k clauses we have the introduction axiom (3.1). Moreover, we have an *elimination axiom* I^-:

$$\forall_{\vec{x}}(I\vec{x} \to (\forall_{\vec{x}_i}((A_{i\nu}(I \wedge X))_{\nu<n_i} \to X\vec{r}_i))_{i<k} \to X\vec{x}\,) \tag{3.2}$$

where $I \wedge X$ abbreviates $\{\,\vec{x} \mid I\vec{x} \wedge X\vec{x}\,\}$ with $\wedge$ defined (inductively) below. Here X can be thought of as a "competitor" predicate.

3.1.3. Examples of inductive predicates. We first deal with the concept of an equality. A word of warning is in order here: we need to distinguish four separate but closely related equalities.

(i) Firstly, defined function constants D are introduced by computation rules, written $l = r$, but intended as left-to-right rewrites.
(ii) Secondly, we have Leibniz equality Eq inductively defined below.
(iii) Thirdly, pointwise equality between partial continuous functionals will be defined inductively as well.
(iv) Fourthly, if l and r have a finitary algebra as their type, $l = r$ can be read as a boolean term, where $=$ is the decidable equality defined in 2.2.3 as a boolean-valued binary function.

Leibniz equality. We define Leibniz equality by

$$\mathrm{Eq}(\rho) := \mu_X(\forall_x X(x^\rho, x^\rho)).$$

The introduction axiom is

$$\forall_x \mathrm{Eq}(x^\rho, x^\rho)$$

and the elimination axiom

$$\forall_{x,y}(\mathrm{Eq}(x, y) \to \forall_x Xxx \to Xxy),$$

where $\mathrm{Eq}(x, y)$ abbreviates $\mathrm{Eq}(\rho)(x^\rho, y^\rho)$.

Lemma (Compatibility of Eq). $\forall_{x,y}(\mathrm{Eq}(x, y) \to A(x) \to A(y))$.

Proof. Exercise. □

Using compatibility of Eq one easily proves symmetry and transitivity. Define *falsity* by $\mathbf{F} := \mathrm{Eq}(\mathtt{ff}, \mathtt{tt})$. Then we have

Theorem (Ex-falso-quodlibet). *For every formula A without predicate parameters we can derive $\mathbf{F} \to A$.*

PROOF. We first show that $\mathbf{F} \to \mathrm{Eq}(x^\rho, y^\rho)$. To see this, we first obtain $\mathrm{Eq}(\mathcal{R}^\rho_{\mathbf{B}}\mathtt{ff}xy, \mathcal{R}^\rho_{\mathbf{B}}\mathtt{ff}xy)$ from the introduction axiom. Then from $\mathrm{Eq}(\mathtt{ff}, \mathtt{tt})$ we get $\mathrm{Eq}(\mathcal{R}^\rho_{\mathbf{B}}\mathtt{tt}xy, \mathcal{R}^\rho_{\mathbf{B}}\mathtt{ff}xy)$ by compatibility. Now $\mathcal{R}^\rho_{\mathbf{B}}\mathtt{tt}xy$ converts to x and $\mathcal{R}^\rho_{\mathbf{B}}\mathtt{ff}xy$ converts to y. Hence $\mathrm{Eq}(x^\rho, y^\rho)$, since we identify terms with a common reduct.

The claim can now be proved by induction on $A \in \mathrm{F}$. *Case* $I\vec{s}$. Let K_i be the nullary clause, with final conclusion $I\vec{t}$. By induction hypothesis from $\mathbf{F}$ we can derive all parameter premises. Hence $I\vec{t}$. From $\mathbf{F}$ we also obtain $\mathrm{Eq}(s_i, t_i)$, by the remark above. Hence $I\vec{s}$ by compatibility. The cases $A \to B$ and $\forall_x A$ are obvious. □

A crucial use of the equality predicate Eq is that it allows us to lift a boolean term $r^{\mathbf{B}}$ to a formula, using $\mathrm{atom}(r^{\mathbf{B}}) := \mathrm{Eq}(r^{\mathbf{B}}, \mathtt{tt})$. This opens up a convenient way to deal with equality on finitary algebras. The computation rules ensure that, for instance, the boolean term $\mathrm{S}r =_{\mathbf{N}} \mathrm{S}s$, or more precisely $=_{\mathbf{N}}(\mathrm{S}r, \mathrm{S}s)$, is identified with $r =_{\mathbf{N}} s$. We can now turn this boolean term into the formula $\mathrm{Eq}(\mathrm{S}r =_{\mathbf{N}} \mathrm{S}s, \mathtt{tt})$, which again is abbreviated by $\mathrm{S}r =_{\mathbf{N}} \mathrm{S}s$, but this time with the understanding that it is a formula. Then (importantly) the two formulas $\mathrm{S}r =_{\mathbf{N}} \mathrm{S}s$ and $r =_{\mathbf{N}} s$ are identified because the latter is a reduct of the first. Consequently there is no need to prove the implication $\mathrm{S}r =_{\mathbf{N}} \mathrm{S}s \to r =_{\mathbf{N}} s$ explicitly.

Existence, conjunction and disjunction. One of the main points of TCF is that it allows the logical connectives existence, conjunction and disjunction to be inductively defined as predicates. This was first discovered by Martin-Löf (1971).

$$\begin{aligned}
\mathrm{Ex}(Y) &:= \mu_X(\forall_x(Yx^\rho \to X)),\\
\mathrm{And}(Y_1, Y_2) &:= \mu_X(Y_1 \to Y_2 \to X),\\
\mathrm{Or}(Y_1, Y_2) &:= \mu_X(Y_1 \to X,\ Y_2 \to X).
\end{aligned}$$

We will use the abbreviations

$$\begin{aligned}
\exists_x A &:= \mathrm{Ex}(\{\, x^\rho \mid A \,\}),\\
A \wedge B &:= \mathrm{And}(\{\mid A\,\}, \{\mid B\,\}),\\
A \vee B &:= \mathrm{Or}(\{\mid A\,\}, \{\mid B\,\}),
\end{aligned}$$

The introduction axioms are

$$\begin{aligned}
&\forall_x(Yx \to \exists_x Yx),\\
&Y_1 \to Y_2 \to Y_1 \wedge Y_2,\\
&Y_1 \to Y_1 \vee Y_2, \qquad Y_2 \to Y_1 \vee Y_2.
\end{aligned}$$

The elimination axioms are

$$\exists_x Yx \to \forall_x(Yx \to X) \to X,$$
$$Y_1 \wedge Y_2 \to (Y_1 \to Y_2 \to X) \to X,$$
$$Y_1 \vee Y_2 \to (Y_1 \to X) \to (Y_2 \to X) \to X.$$

We give some more familiar examples of inductively defined predicates.

The even numbers. The introduction axioms are

$$\mathrm{Even}(0), \qquad \forall_n(\mathrm{Even}(n) \to \mathrm{Even}(\mathrm{S}(\mathrm{S}n)))$$

and the elimination axiom is

$$\forall_n(\mathrm{Even}(n) \to X0 \to \forall_n(\mathrm{Even}(n) \to Xn \to X(\mathrm{S}(\mathrm{S}n))) \to Xn).$$

Transitive closure. Let $\prec$ be a predicate variable representing a binary relation. The *transitive closure* $\mathrm{TC}_\prec$ of $\prec$ is inductively defined as follows. The introduction axioms are

$$\forall_{x,y}(x \prec y \to \mathrm{TC}_\prec(x,y)),$$
$$\forall_{x,y,z}(x \prec y \to \mathrm{TC}_\prec(y,z) \to \mathrm{TC}_\prec(x,z))$$

and the elimination axiom is

$$\forall_{x,y}(\mathrm{TC}_\prec(x,y) \to \forall_{x,y}(x \prec y \to Xxy) \to \forall_{x,y,z}(x \prec y \to \mathrm{TC}_\prec(y,z) \to Xyz \to Xxz) \to Xxy).$$

3.2. Realizability interpretation

At this point we come to the crucial step of identifying "computational content" in proofs, which can then be extracted. Recall that the BHK-interpretation (described in 3.1.1) left open what a proof of a prime formula is. However, in TCF we can be more definite, since a closed prime formula must be of the form $I\vec{r}$ with I an inductive predicate. The obvious idea is to view a proof of $I\vec{r}$ as a "generation tree", witnessing how the arguments $\vec{r}$ were put into I. For example, let Even be defined by the clauses Even(0) and $\forall_n(\mathrm{Even}(n) \to \mathrm{Even}(\mathrm{S}(\mathrm{S}n)))$. A generation tree for Even(6) should consist of a single branch with nodes Even(0), Even(2), Even(4) and Even(6). More formally, such a generation tree can seen as an ideal in an algebra ι_I associated naturally with I.

Consider the more general situation when parameters are involved, i.e., when we have a proof (in TCF) of a closed formula $\forall_{\vec{x}}(\vec{A} \to I\vec{r})$. It is of obvious interest which of the variables $\vec{x}$ and assumptions $\vec{A}$ are actually used

in the "solution" provided by the proof (in the sense of Kolmogorov (1932)). To be able to express dependence on and independence of such parameters we split each of our (only!) logical connectives $\to, \forall$ into two variants, a "computational" one $\forall^{\mathrm{c}}, \to^{\mathrm{c}}$ and a "non-computational" one $\forall^{\mathrm{nc}}, \to^{\mathrm{nc}}$. This distinction (for the universal quantifier) is due to Berger (1993, 2005). Then a proof of $\forall^{\mathrm{nc}}_{\vec{x}} \forall^{\mathrm{c}}_{\vec{y}} (\vec{A} \to^{\mathrm{nc}} \vec{B} \to^{\mathrm{c}} I\vec{r})$ provides a construction of an ideal in ι_I independent of $\vec{x}$ and assumed proofs of $\vec{A}$. One can view this "decoration" of $\to, \forall$ as turning our (minimal) logic into a "computational logic", which is able to express dependence on and independence of parameters. The rules for $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ are similar to the ones for $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$; they will be given in 3.2.2.

Now the clauses of inductive predicates can and should be decorated as well. Without loss of generality we can assume that they have the form

$$\forall^{\mathrm{nc}}_{\vec{x}} \forall^{\mathrm{c}}_{\vec{y}} (\vec{A} \to^{\mathrm{nc}} \vec{B} \to^{\mathrm{c}} X\vec{r}).$$

This will lead to a different (i.e., simplified) algebra ι_I associated with the inductive predicate I.

Of special importance is the case when we only have $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$, and there is only one clause. Such inductive predicates are called "uniform one-clause defined", and their associated algebra is the unit algebra $\mathbf{U}$. Examples are Leibniz equality, existence and conjunction when defined with $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$:

$$\begin{aligned}
\mathrm{Eq}(\rho) &:= \mu_X (\forall^{\mathrm{nc}}_x X(x^\rho, x^\rho)),\\
\mathrm{ExU}(Y) &:= \mu_X (\forall^{\mathrm{nc}}_x (Y x^\rho \to^{\mathrm{nc}} X)),\\
\mathrm{AndU}(Y_1, Y_2) &:= \mu_X (Y_1 \to^{\mathrm{nc}} Y_2 \to^{\mathrm{nc}} X).
\end{aligned}$$

From now on we only use this uniform one-clause definition of Leibniz equality Eq, and use the abbreviations

$$\begin{aligned}
\exists^{\mathrm{u}}_x A &:= \mathrm{ExU}(\{\, x^\rho \mid A \,\}).\\
A \wedge^{\mathrm{u}} B &:= \mathrm{AndU}(\{ \mid A \}, \{ \mid B \}).
\end{aligned}$$

Prime formulas $I\vec{r}$ with $\iota_I = \mathbf{U}$ only have a trivial generation tree, and in this sense are without computational content. Clearly this is also the case for formulas with such an $I\vec{r}$ as conclusion. These formulas are called non-computational (n.c.) or *Harrop formulas*. Moreover, a Harrop formula in a premise can be ignored when we are interested in the computational content of a proof of this formula: its only contribution would be of unit type. Therefore when defining the type of a formula in 3.2.5 we will use a "cleaned" form of such a type, not involving the unit type.

The next thing to do is to properly accomodate the BHK-interpretation and define what it means that a term t "realizes" the formula A, written

t **r** A. In the prime formula case $I\vec{r}$ this will involve a predicate "t realizes $I\vec{r}$", which will be defined inductively as well, following the clauses of I. But since this is a "meta" statement already containing the term t representing a generation tree, we are not interested in the generation tree for such realizing formulas and consider them as non-computational.

Finally we will define in 3.2.6 the "extracted term" $\mathrm{et}(M)$ of a proof M of a formula A, and prove the soundness theorem $\mathrm{et}(M)$ **r** A.

Remark. We have encountered two situations where inductive definitions do not have computational content: uniform one-clause defined predicates, and realizability predicates. There is a third occasion when this can happen and is in fact rather useful, namely when the all clauses have "invariant" premises A only; a formula A is called invariant if $\exists_x(x$ **r** $A)$ is equivalent to A. We write $\mu^{\mathrm{nc}}_X(K_0,\dots,K_{k-1})$ whenever an inductive predicate is n.c. The soundness theorem continues to hold if we restrict usage of the least-fixed-point (or elimination) axiom for such n.c. inductive predicates to Harrop formulas.

3.2.1. An informal explanation. The ideas that we develop here are illustrated by the following simple situation. The computational content of an implication $Pn \to^{\mathrm{c}} P(\mathrm{S}n)$ is that demanded of an implication by the BHK interpretation, namely a function from evidence for Pn to evidence for $P(\mathrm{S}n)$. The universal quantifier $\forall_n$ is non-computational if it merely supplies n as an "input", whereas to say that a universal quantifier is computational means that a construction of input n is also supplied. Thus a realization of

$$\forall^{\mathrm{nc}}_n(Pn \to^{\mathrm{c}} P(\mathrm{S}n))$$

will be a unary function f such that if r "realizes" Pn, then fr realizes $P(\mathrm{S}n)$, for every n. On the other hand, a realization of

$$\forall^{\mathrm{c}}_n(Pn \to^{\mathrm{c}} P(\mathrm{S}n))$$

will be a binary function g which, given a number n and a realization r of Pn, produces a realization $g(n,r)$ of $P(\mathrm{S}n)$. Therefore an induction with basis and step of the form

$$P0, \qquad \forall^{\mathrm{nc}}_n(Pn \to^{\mathrm{c}} P(\mathrm{S}n))$$

will be realized by iterates $f^{(n)}(r_0)$, whereas a computational induction

$$P0, \qquad \forall^{\mathrm{c}}_n(Pn \to^{\mathrm{c}} P(\mathrm{S}n))$$

will be realized by the primitive recusively defined $h(n,r_0)$ where $h(0,r_0) = r_0$ and $h(\mathrm{S}n,r_0) = g(n,h(n,r_0))$.

Finally, a word about the non-computational implication: a realizer of $A \to^{\mathrm{nc}} B$ will depend solely on the existence of a realizer of A, but will be completely independent of which one it is. An example would be an induction

$$P0, \qquad \forall^{\mathrm{c}}_n(Pn \to^{\mathrm{nc}} P(\mathrm{S}n))$$

where the realizer $h(n, r_0)$ is given by $h(0, r_0) = r_0$, $h(\mathrm{S}n, r_0) = g(n)$, without recursive calls. The point is that in this case g does not depend on a realizer for Pn, only upon the number n itself.

3.2.2. Decorating $\to$ and $\forall$. We adapt the definition in 3.1.2 of predicates and formulas to the newly introduced decorated connectives $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$. Let $\to$ denote either $\to^{\mathrm{c}}$ or $\to^{\mathrm{nc}}$, and similarly $\forall$ either $\forall^{\mathrm{c}}$ or $\forall^{\mathrm{nc}}$. Then the definition in 3.1.2 can be read as it stands.

We also need to adapt our definition of TCF to the decorated connectives $\to^{\mathrm{c}}, \to^{\mathrm{nc}}$ and $\forall^{\mathrm{c}}, \forall^{\mathrm{nc}}$. The introduction and elimination rules for $\to^{\mathrm{c}}$ and $\forall^{\mathrm{c}}$ remain as before, and also the elimination rules for $\to^{\mathrm{nc}}$ and $\forall^{\mathrm{nc}}$. However, the introduction rules for $\to^{\mathrm{nc}}$ and $\forall^{\mathrm{nc}}$ must be restricted: the abstracted (assumption or object) variable must be "non-computational", in the following sense. Simultaneously with a derivation M we define the sets $\mathrm{CV}(M)$ and $\mathrm{CA}(M)$ of *computational* object and assumption variables of M, as follows. Let M^A be a derivation. If A is non-computational (n.c.), i.e., the type $\tau(A)$ of A (defined below in 3.2.5) is the "nulltype" symbol $\circ$, then $\mathrm{CV}(M^A) := \mathrm{CA}(M^A) := \emptyset$. Otherwise

$$\begin{aligned}
&\mathrm{CV}(c^A) := \emptyset \quad (c^A \text{ an axiom}),\\
&\mathrm{CV}(u^A) := \emptyset,\\
&\mathrm{CV}((\lambda_{u^A} M^B)^{A\to^{\mathrm{c}}B}) := \mathrm{CV}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}}B}) := \mathrm{CV}(M),\\
&\mathrm{CV}((M^{A\to^{\mathrm{c}}B} N^A)^B) := \mathrm{CV}(M) \cup \mathrm{CV}(N),\\
&\mathrm{CV}((M^{A\to^{\mathrm{nc}}B} N^A)^B) := \mathrm{CV}(M),\\
&\mathrm{CV}((\lambda_x M^A)^{\forall^{\mathrm{c}}_x A}) := \mathrm{CV}((\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}) := \mathrm{CV}(M) \setminus \{x\},\\
&\mathrm{CV}((M^{\forall^{\mathrm{c}}_x A(x)} r)^{A(r)}) := \mathrm{CV}(M) \cup \mathrm{FV}(r),\\
&\mathrm{CV}((M^{\forall^{\mathrm{nc}}_x A(x)} r)^{A(r)}) := \mathrm{CV}(M),
\end{aligned}$$

and similarly

$$\begin{aligned}
&\mathrm{CA}(c^A) := \emptyset \quad (c^A \text{ an axiom}),\\
&\mathrm{CA}(u^A) := \{u\},
\end{aligned}$$

$$\mathrm{CA}((\lambda_{u^A} M^B)^{A\to^{\mathrm{c}}B}) := \mathrm{CA}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}}B}) := \mathrm{CA}(M^A)\setminus\{u\},$$
$$\mathrm{CA}((M^{A\to^{\mathrm{c}}B}N^A)^B) := \mathrm{CA}(M)\cup\mathrm{CA}(N),$$
$$\mathrm{CA}((M^{A\to^{\mathrm{nc}}B}N^A)^B) := \mathrm{CA}(M),$$
$$\mathrm{CA}((\lambda_x M^A)^{\forall^{\mathrm{c}}_x A}) := \mathrm{CA}((\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}) := \mathrm{CA}(M),$$
$$\mathrm{CA}((M^{\forall^{\mathrm{c}}_x A(x)}r)^{A(r)}) := \mathrm{CA}((M^{\forall^{\mathrm{nc}}_x A(x)}r)^{A(r)}) := \mathrm{CA}(M).$$

The introduction rules for $\to^{\mathrm{nc}}$ and $\forall^{\mathrm{nc}}$ then are

(i) If M^B is a derivation and $u^A \notin \mathrm{CA}(M)$ then $(\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}}B}$ is a derivation.

(ii) If M^A is a derivation, x is not free in any formula of a free assumption variable of M and $x \notin \mathrm{CV}(M)$, then $(\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}$ is a derivation.

An alternative way to formulate these rules is simultaneously with the notion of the "extracted term" $\mathrm{et}(M)$ of a derivation M. This will be done in 3.2.6.

3.2.3. Decorating inductive definitions. Now we can and should decorate inductive definitions. The introduction axioms are

$$K_i := \forall^{\mathrm{c/nc}}_{\vec{x}}((A_\nu(I))_{\nu<n} \to^{\mathrm{c/nc}} I\vec{r}) \tag{3.3}$$

and the elimination axiom is

$$\forall^{\mathrm{nc}}_{\vec{x}}(I\vec{x} \to^{\mathrm{c}} (\forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu}(I \wedge^{\mathrm{d}} X))_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k} \to^{\mathrm{c}} X\vec{x}) \tag{3.4}$$

where $I \wedge^{\mathrm{d}} X$ abbreviates $\{\,\vec{x} \mid I\vec{x} \wedge^{\mathrm{d}} X\vec{x}\,\}$ with $\wedge^{\mathrm{d}}$ defined below.

Let us decorate the inductively defined predicates in 3.1.3, that is, take computational aspects into account. For $\exists$, $\wedge$ and $\vee$ we obtain $\exists^{\mathrm{d}}, \exists^{\mathrm{l}}, \exists^{\mathrm{r}}, \exists^{\mathrm{u}}$, $\wedge^{\mathrm{d}}, \wedge^{\mathrm{l}}, \wedge^{\mathrm{r}}, \wedge^{\mathrm{u}}$ $\vee^{\mathrm{d}}, \vee^{\mathrm{l}}, \vee^{\mathrm{r}}, \vee^{\mathrm{u}}$ with d for "double", l for "left", r for "right" and u for "uniform". They are defined by their introduction axioms, which involve both $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$.

$$\forall^{\mathrm{c}}_x(A \to^{\mathrm{c}} \exists^{\mathrm{d}}_x A), \qquad \exists^{\mathrm{d}}_x A \to^{\mathrm{c}} \forall^{\mathrm{c}}_x(A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P,$$
$$\forall^{\mathrm{c}}_x(A \to^{\mathrm{nc}} \exists^{\mathrm{l}}_x A), \qquad \exists^{\mathrm{l}}_x A \to^{\mathrm{c}} \forall^{\mathrm{c}}_x(A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P,$$
$$\forall^{\mathrm{nc}}_x(A \to^{\mathrm{c}} \exists^{\mathrm{r}}_x A), \qquad \exists^{\mathrm{r}}_x A \to^{\mathrm{c}} \forall^{\mathrm{nc}}_x(A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P,$$
$$\forall^{\mathrm{nc}}_x(A \to^{\mathrm{nc}} \exists^{\mathrm{u}}_x A), \qquad \exists^{\mathrm{u}}_x A \to^{\mathrm{c}} \forall^{\mathrm{nc}}_x(A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P,$$

and similar for $\wedge$:

$$A \to^{\mathrm{c}} B \to^{\mathrm{c}} A \wedge^{\mathrm{d}} B, \qquad A \wedge^{\mathrm{d}} B \to^{\mathrm{c}} (A \to^{\mathrm{c}} B \to^{\mathrm{c}} P) \to^{\mathrm{c}} P,$$
$$A \to^{\mathrm{c}} B \to^{\mathrm{nc}} A \wedge^{\mathrm{l}} B, \qquad A \wedge^{\mathrm{l}} B \to^{\mathrm{c}} (A \to^{\mathrm{c}} B \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P,$$
$$A \to^{\mathrm{nc}} B \to^{\mathrm{c}} A \wedge^{\mathrm{r}} B, \qquad A \wedge^{\mathrm{r}} B \to^{\mathrm{c}} (A \to^{\mathrm{nc}} B \to^{\mathrm{c}} P) \to^{\mathrm{c}} P$$
$$A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} A \wedge^{\mathrm{u}} B, \qquad A \wedge^{\mathrm{u}} B \to^{\mathrm{c}} (A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P$$

and for $\vee$:

$$\begin{array}{ll} A \to^{\mathrm{c}} A \vee^{\mathrm{d}} B, & B \to^{\mathrm{c}} A \vee^{\mathrm{d}} B, \\ A \to^{\mathrm{c}} A \vee^{\mathrm{l}} B, & B \to^{\mathrm{nc}} A \vee^{\mathrm{l}} B, \\ A \to^{\mathrm{nc}} A \vee^{\mathrm{r}} B, & B \to^{\mathrm{c}} A \vee^{\mathrm{r}} B, \\ A \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B, & B \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B \end{array}$$

with elimination schemes

$$\begin{array}{l} A \vee^{\mathrm{d}} B \to^{\mathrm{c}} (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} (B \to^{\mathrm{c}} P) \to^{\mathrm{c}} P, \\ A \vee^{\mathrm{l}} B \to^{\mathrm{c}} (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} (B \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P, \\ A \vee^{\mathrm{r}} B \to^{\mathrm{c}} (A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} (B \to^{\mathrm{c}} P) \to^{\mathrm{c}} P, \\ A \vee^{\mathrm{u}} B \to^{\mathrm{c}} (A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} (B \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P. \end{array}$$

Let $\prec$ be a predicate variable representing a binary relation. A computational variant of the inductively defined *transitive closure* $\mathrm{TC}_{\prec}$ of $\prec$ has introduction axioms

$$\begin{array}{l} \forall^{\mathrm{c}}_{x,y}(x \prec y \to^{\mathrm{nc}} \mathrm{TC}_{\prec}(x,y)), \\ \forall^{\mathrm{c}}_{x,y}\forall^{\mathrm{nc}}_{z}(x \prec y \to^{\mathrm{nc}} \mathrm{TC}_{\prec}(y,z) \to^{\mathrm{c}} \mathrm{TC}_{\prec}(x,z)), \end{array}$$

and the elimination scheme is

$$\begin{array}{l} \forall^{\mathrm{nc}}_{x,y}(\mathrm{TC}_{\prec}(x,y) \to^{\mathrm{c}} \forall^{\mathrm{c}}_{x,y}(x \prec y \to^{\mathrm{nc}} Pxy) \to^{\mathrm{c}} \\ \qquad \forall^{\mathrm{c}}_{x,y}\forall^{\mathrm{nc}}_{z}(x \prec y \to^{\mathrm{nc}} \mathrm{TC}_{\prec}(y,z) \to^{\mathrm{c}} Pyz \to^{\mathrm{c}} Pxz) \to^{\mathrm{c}} \\ \qquad Pxy). \end{array}$$

Consider the accessible part of a binary relation $\prec$. A computational variant $\mathrm{Acc}_{\prec}$ is determined by the introduction axioms

$$\begin{array}{l} \forall^{\mathrm{c}}_{x}(\mathbf{F} \to^{\mathrm{nc}} \mathrm{Acc}_{\prec}(x)), \\ \forall^{\mathrm{nc}}_{x}(\forall^{\mathrm{c}}_{y \prec x}\mathrm{Acc}_{\prec}(y) \to^{\mathrm{c}} \mathrm{Acc}_{\prec}(x)), \end{array}$$

where $\forall^{\mathrm{c}}_{y \prec x}A$ stands for $\forall^{\mathrm{c}}_{y}(y \prec x \to^{\mathrm{nc}} A)$. The elimination scheme is

$$\begin{array}{l} \forall^{\mathrm{nc}}_{x}(\mathrm{Acc}_{\prec}(x) \to^{\mathrm{c}} \forall^{\mathrm{c}}_{x}(\mathbf{F} \to^{\mathrm{nc}} Px) \to^{\mathrm{c}} \\ \qquad \forall^{\mathrm{nc}}_{x}(\forall^{\mathrm{c}}_{y \prec x}\mathrm{Acc}_{\prec}(y) \to^{\mathrm{c}} \forall^{\mathrm{c}}_{y \prec x}Py \to^{\mathrm{c}} Px) \to^{\mathrm{c}} \\ \qquad Px). \end{array}$$

3.2.4. Totality and induction. In 2.1.6 we have defined what the total and structure-total ideals of a finitary algebra are. We now inductively define general totality predicates. Let us first look at some examples. The clauses defining totality for the algebra $\mathbf{N}$ are

$$T_{\mathbf{N}}0, \qquad \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to^{\mathrm{c}} T_{\mathbf{N}}(\mathrm{S}n)).$$

The least-fixed-point axiom is

$$\forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to^{\mathrm{c}} X0 \to^{\mathrm{c}} \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to^{\mathrm{c}} Xn \to^{\mathrm{c}} X(\mathrm{S}n)) \to^{\mathrm{c}} Xn).$$

Clearly the partial continuous functionals with $T_{\mathbf{N}}$ interpreted as the total ideals for $\mathbf{N}$ provide a model of TCF extended by these axioms.

For the algebra $\mathbf{D}$ of derivations totality is inductively defined by the clauses

$$T_{\mathbf{D}}0^{\mathbf{D}}, \qquad \forall^{\mathrm{nc}}_{x,y}(T_{\mathbf{D}}x \to^{\mathrm{c}} T_{\mathbf{D}}y \to^{\mathrm{c}} T_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}xy)),$$

with least-fixed-point axiom

$$\begin{aligned}\forall^{\mathrm{nc}}_x(T_{\mathbf{D}}x \to^{\mathrm{c}} X0^{\mathbf{D}} \to^{\mathrm{c}}\ & \\ & \forall^{\mathrm{nc}}_{x,y}(T_{\mathbf{D}}x \to^{\mathrm{c}} T_{\mathbf{D}}y \to^{\mathrm{c}} Xx \to^{\mathrm{c}} Xy \to^{\mathrm{c}} X(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}xy)) \to^{\mathrm{c}} \\ & Xx).\end{aligned}$$

Again, the partial continuous functionals with $T_{\mathbf{D}}$ interpreted as the total ideals for $\mathbf{D}$ (i.e., the finite derivations) provide a model.

Generally we define

(i) RT_ρ called *relative totality*, and its special cases
(ii) T_ρ called (absolute) *totality* and
(iii) ST_ρ called *structural totality*.

The least-fixed-point axiom for ST_ι will provide us with the induction axiom for the algebra ι.

The definition of RT_ρ is relative to an assigment of predicate variables Y of arity (α) to type variables α.

Definition (Relative totality RT). Let $\iota = \mu_\xi(\kappa_0, \dots, \kappa_{k-1}) \in \mathrm{Alg}(\vec{\alpha}\,)$ with $\kappa_i = (\rho_\nu(\vec{\alpha}, \xi))_{\nu<n} \to \xi$. Then $\mathrm{RT}_\iota := \mu_X(K_0, \dots, K_{k-1})$, with

$$K_i := \forall^{\mathrm{nc}}_{\vec{x}}((\mathrm{RT}_{\rho_\nu}(\vec{Y}, X)x_\nu)_{\nu<n} \to^{\mathrm{c}} X(\mathrm{C}_i\vec{x}\,))$$

and

$$\begin{aligned}\mathrm{RT}_{\alpha_j}(\vec{Y}, X) &:= Y_j,\\ \mathrm{RT}_{\xi}(\vec{Y}, X) &:= X,\\ \mathrm{RT}_{\sigma\to\rho}(\vec{Y}, X) &:= \{\, f \mid \forall^{\mathrm{nc}}_{\vec{x}}(\mathrm{RT}_\sigma\vec{x} \to^{\mathrm{c}} \mathrm{RT}_\rho(\vec{Y}, X)(f\vec{x}\,))\,\}.\end{aligned}$$

For important special cases of the parameter predicates $\vec{Y}$ we introduce a separate notation. Suppose we want to argue about total ideals only. Note that this only makes sense when when no type variables occur. However, to allow a certain amount of abstract reasing (involving type variables to be substituted later by concrete closed types), we introduce special predicate variables T_α which under a substitution $\alpha \mapsto \rho$ with ρ closed turn into the inductively defined predicate T_ρ. Using this convention we define totality for an arbitrary algebra by specializing Y of arity (ρ) to T_ρ.

Definition (Absolute totality T). Let $\iota = \mu_\xi(\kappa_0, \dots, \kappa_{k-1}) \in \mathrm{Alg}(\vec{\alpha}\,)$ with $\kappa_i = (\rho_\nu(\vec{\alpha}, \xi))_{\nu<n} \to \xi$. Then $T_\iota := \mu_X(K_0, \dots, K_{k-1})$, with

$$K_i := \forall^{\mathrm{nc}}_{\vec{x}}((T_{\rho_\nu}(X)x_\nu)_{\nu<n} \to^{\mathrm{c}} X(\mathrm{C}_i\vec{x}\,))$$

and

$$\begin{aligned} T_{\alpha_j}(X) &:= T_{\alpha_j}, \\ T_\xi(X) &:= X, \\ T_{\sigma\to\rho}(X) &:= \{\, f \mid \forall^{\mathrm{nc}}_{\vec{x}}(T_\sigma\vec{x} \to^{\mathrm{c}} T_\rho(X)(f\vec{x}\,))\,\}. \end{aligned}$$

Another important special case occurs when we substitute the predicate variables Y by truth predicates. The resulting totality predicate is called structural totality.

Definition (Structural totality ST). Let $\iota = \mu_\xi(\kappa_0, \dots, \kappa_{k-1}) \in \mathrm{Alg}(\vec{\alpha}\,)$ with $\kappa_i = (\rho_\nu(\vec{\alpha}, \xi))_{\nu<n} \to \xi$. Then $\mathrm{ST}_\iota := \mu_X(K_0, \dots, K_{k-1})$, with

$$K_i := \forall^{\mathrm{nc}}_{\vec{x}}((\mathrm{ST}_{\rho_\nu}(X)x_\nu)_{\nu<n} \to^{\mathrm{c}} X(\mathrm{C}_i\vec{x}\,))$$

and

$$\begin{aligned} \mathrm{ST}_{\alpha_j}(X) &:= \{\, x \mid \top\,\} \quad \text{(omitted whenever possible)}, \\ \mathrm{ST}_\xi(X) &:= X, \\ \mathrm{ST}_{\sigma\to\rho}(X) &:= \{\, f \mid \forall^{\mathrm{nc}}_{\vec{x}}(\mathrm{ST}_\sigma\vec{x} \to^{\mathrm{c}} \mathrm{ST}_\rho(X)(f\vec{x}\,))\,\}. \end{aligned}$$

For example, the main clause for the predicate $\mathrm{ST}_{\mathbf{L}(\alpha)}$ expressing structural totality of lists of elements of type α is

$$\forall^{\mathrm{nc}}_{x,l}(\underbrace{\mathrm{ST}_\alpha(X)x}_{\top;\ \text{omit}} \to^{\mathrm{c}} \underbrace{\mathrm{ST}_\xi(X)}_{X}\, l \to^{\mathrm{c}} X(x :: l))$$

where $x :: l$ is shorthand for $\mathrm{Cons}(x, l)$. It leads to the introduction axiom

$$\forall^{\mathrm{nc}}_{x,l}(\mathrm{ST}_{\mathbf{L}(\alpha)}l \to^{\mathrm{c}} \mathrm{ST}_{\mathbf{L}(\alpha)}(x :: l))$$

with no assumptions on x.

The least-fixed-point axiom for $\mathrm{ST}_{\mathbf{L}(\alpha)}$ is according to (3.4)

$$\forall^{\mathrm{nc}}_l(\mathrm{ST}(l) \to^{\mathrm{c}} X(\mathrm{Nil}) \to^{\mathrm{c}} \forall^{\mathrm{nc}}_{x,l}((\mathrm{ST} \wedge^{\mathrm{d}} X)l \to^{\mathrm{c}} X(x :: l)) \to^{\mathrm{c}} Xl^{\mathbf{L}(\rho)}).$$

Written differently (with "duplication") we obtain the induction axiom

$$\forall^{\mathrm{nc}}_l(\mathrm{ST}(l) \to^{\mathrm{c}} X(\mathrm{Nil}) \to \forall^{\mathrm{nc}}_{x,l}(\mathrm{ST}(l) \to^{\mathrm{c}} Xl \to^{\mathrm{c}} X(x :: l)) \to^{\mathrm{c}} Xl^{\mathbf{L}(\rho)})$$

denoted $\mathrm{Ind}_{l,X}$.

Note that in all these definitions we allow usage of totality predicates for previously introduced algebras ι'. An example is totality $T_{\mathbf{T}}$ for the algebra $\mathbf{T}$ of finitely branching trees. It is defined by the single clause

$$\forall^{\mathrm{nc}}_{as}(\mathrm{RT}_{\mathbf{L}(\mathbf{T})}(T_{\mathbf{T}})(as) \to^{\mathrm{c}} T_{\mathbf{T}}(\mathrm{Branch}(as))).$$

Clearly all three notions of totality coincide for algebras without type parameters. Abbreviating $\forall^{\mathrm{nc}}_x(Tx \to^{\mathrm{c}} A)$ by $\forall^{\mathrm{c}}_{x\in T} A$ we obtain from the elimination axioms *computational induction schemes*, for example

$$\mathrm{Ind}_{p,P}\colon \forall^{\mathrm{c}}_{p\in T}(P\mathtt{tt} \to^{\mathrm{c}} P\mathtt{ff} \to^{\mathrm{c}} Pp^{\mathbf{B}}),$$
$$\mathrm{Ind}_{n,P}\colon \forall^{\mathrm{c}}_{n\in T}(P0 \to^{\mathrm{c}} \forall^{\mathrm{c}}_{n\in T}(Pn \to^{\mathrm{c}} P(\mathrm{S}n)) \to^{\mathrm{c}} Pn^{\mathbf{N}}).$$

The types of these formulas (as defined in 3.2.5) will be the types of the recursion operators of the respective algebras.

3.2.5. The type of a formula, realizability and witnesses. For every formula or predicate C we define $\tau(C)$ (a type or the "nulltype" symbol $\circ$). In case $\tau(C) = \circ$ proofs of C have no computational content; such C are called *non-computational* (n.c.) (or *Harrop*); the other ones are called *computationally relevant* (c.r.). The definition can be conveniently written if we extend the use of $\rho \to \sigma$ to the nulltype symbol $\circ$:

$$(\rho \to \circ) := \circ, \quad (\circ \to \sigma) := \sigma, \quad (\circ \to \circ) := \circ.$$

Definition (Type $\tau(C)$ of a formula or predicate C, and ι_I). Assume a global injective assignment of a type variable ξ to every predicate variable X.

$$\tau(P\vec{r}) := \tau(P),$$
$$\tau(A \to^{\mathrm{c}} B) := (\tau(A) \to \tau(B)), \quad \tau(A \to^{\mathrm{nc}} B) := \tau(B),$$
$$\tau(\forall^{\mathrm{c}}_{x^\rho} A) := (\rho \to \tau(A)), \quad \tau(\forall^{\mathrm{nc}}_{x^\rho} A) := \tau(A),$$
$$\tau(X) := \xi,$$
$$\tau(\{\, \vec{x} \mid A \,\}) := \tau(A),$$
$$\tau(\mu^{\mathrm{nc}}_X(K_0, \dots, K_{k-1})) := \circ,$$

$$\tau(\mu_X(\forall^{\mathrm{nc}}_{\vec{x}_i}\forall^{\mathrm{c}}_{\vec{y}_i}(\vec{A}_i \to^{\mathrm{nc}} \vec{B}_i \to^{\mathrm{c}} X\vec{r}_i))_{i<k}) := \mu_\xi(\tau(\vec{y}_i) \to \tau(\vec{B}_i) \to \xi)_{i<k}.$$

We call $\iota_I := \mu_\xi(\tau(\vec{y}_i) \to \tau(\vec{B}_i) \to \xi)_{i<k}$ the algebra associated with the inductive predicate $I := \mu_X(\forall^{\mathrm{nc}}_{\vec{x}_i}\forall^{\mathrm{c}}_{\vec{y}_i}(\vec{A}_i \to^{\mathrm{nc}} \vec{B}_i \to^{\mathrm{c}} X\vec{r}_i))_{i<k}$.

Hence

$$\tau(I\vec{r}) = \begin{cases} \iota_I & \text{if } I \text{ is c.r.,} \\ \circ & \text{if } I \text{ is n.c.} \end{cases}$$

We now define *realizability*. It will be convenient to introduce a special "nullterm" symbol ε to be used as a "realizer" for n.c. formulas. We extend term application to the nullterm symbol by

$$\varepsilon t := \varepsilon, \quad t\varepsilon := t, \quad \varepsilon\varepsilon := \varepsilon.$$

Definition (t realizes A). Let A be a formula and t either a term of type $\tau(A)$ if the latter is a type, or the nullterm symbol ε for n.c. A. We assume an injective global assignment, giving for every predicate variable X of arity $\vec{\rho}$ a predicate variable $X^{\mathbf{r}}$ of arity $(\tau(X), \vec{\rho})$.

$$\begin{aligned}
&t\ \mathbf{r}\ X\vec{r} := X^{\mathbf{r}}t\vec{r},\\
&t\ \mathbf{r}\ (A \to^{\mathrm{c}} B) := \forall^{\mathrm{nc}}_x(x\ \mathbf{r}\ A\ \to^{\mathrm{nc}}\ tx\ \mathbf{r}\ B),\\
&t\ \mathbf{r}\ (A \to^{\mathrm{nc}} B) := \forall^{\mathrm{nc}}_x(x\ \mathbf{r}\ A\ \to^{\mathrm{nc}}\ t\ \mathbf{r}\ B),\\
&t\ \mathbf{r}\ \forall^{\mathrm{c}}_x A := \forall^{\mathrm{nc}}_x(tx\ \mathbf{r}\ A),\\
&t\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A := \forall^{\mathrm{nc}}_x(t\ \mathbf{r}\ A),\\
&t\ \mathbf{r}\ (\mu_X(\forall^{\mathrm{nc}}_{\vec{x}_i}\forall^{\mathrm{c}}_{\vec{y}_i}((A_{i\nu})_{\nu<n_i} \to^{\mathrm{nc}} (B_{i\nu})_{\nu<m_i} \to^{\mathrm{c}} X\vec{r}_i))_{i<k})\vec{s} := I^{\mathbf{r}}t\vec{s}
\end{aligned}$$

with

$$\begin{aligned}
I^{\mathbf{r}} := \{\, w, \vec{x} \mid (\mu^{\mathrm{nc}}_X(\forall^{\mathrm{nc}}_{\vec{x}_i,\vec{y}_i,\vec{u}_i}((\exists_{u_{i\nu}} u_{i\nu}\ \mathbf{r}\ A_{i\nu})_{\nu<n_i} \to^{\mathrm{nc}} (v_{i\nu}\ \mathbf{r}\ B_{i\nu})_{\nu<m_i} \to^{\mathrm{nc}}\\
X(\mathrm{C}_i\vec{y}_i\vec{v}_i)\vec{r}_i))_{i<k})w\vec{x}\,\}.
\end{aligned}$$

In case A is n.c., $\forall^{\mathrm{nc}}_x(x\ \mathbf{r}\ A \to^{\mathrm{nc}} B(x))$ means $\varepsilon\ \mathbf{r}\ A \to^{\mathrm{nc}} B(\varepsilon)$. For a general n.c. inductively defined predicate (with restricted elimination scheme) we define $\varepsilon\ \mathbf{r}\ I\vec{s}$ to be $I\vec{s}$. For the special n.c. inductively defined predicates $I^{\mathbf{r}}$, Eq, $\exists^{\mathrm{u}}$ and $\wedge^{\mathrm{u}}$ introduced above realizability is defined by

$$\begin{aligned}
\varepsilon\ \mathbf{r}\ I^{\mathbf{r}}t\vec{s} &:= I^{\mathbf{r}}t\vec{s},\\
\varepsilon\ \mathbf{r}\ \mathrm{Eq}(t,s) &:= \mathrm{Eq}(t,s),\\
\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_x A &:= \exists^{\mathrm{u}}_{x,y}(y\ \mathbf{r}\ A),\\
\varepsilon\ \mathbf{r}\ (A \wedge^{\mathrm{u}} B) &:= \exists^{\mathrm{u}}_x(x\ \mathbf{r}\ A) \wedge^{\mathrm{u}} \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ B).
\end{aligned}$$

NOTE. Call two formulas A and A' *computationally equivalent* if each of them computationally implies the other, and in addition the identity realizes each of the two derivations of $A' \to^{\mathrm{c}} A$ and of $A \to^{\mathrm{c}} A'$. It is an easy exercise to verify that for n.c. A, the formulas $A \to^{\mathrm{c}} B$ and $A \to^{\mathrm{nc}} B$ are computationally equivalent, and hence can be identified. In the sequel we shall simply write $A \to B$ for either of them. Similarly, for n.c. A the two formulas $\forall^{\mathrm{c}}_x A$ and $\forall^{\mathrm{nc}}_x A$ are n.c., and both $\varepsilon\ \mathbf{r}\ \forall^{\mathrm{c}}_x A$ and $\varepsilon\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A$ are defined to be $\forall^{\mathrm{nc}}_x(\varepsilon\ \mathbf{r}\ A)$. Hence they can be identified as well, and we shall simply write $\forall_x A$ for either of them. Since the formula $t\ \mathbf{r}\ A$ is n.c., under this convention the $\to, \forall$-cases in the definition of realizability can be written

$$\begin{aligned}
t\ \mathbf{r}\ (A \to^{\mathrm{c}} B) &:= \forall_x(x\ \mathbf{r}\ A \to t x\ \mathbf{r}\ B),\\
t\ \mathbf{r}\ (A \to^{\mathrm{nc}} B) &:= \forall_x(x\ \mathbf{r}\ A \to t\ \mathbf{r}\ B),\\
t\ \mathbf{r}\ \forall^{\mathrm{c}}_x A &:= \forall_x(t x\ \mathbf{r}\ A),\\
t\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A &:= \forall_x(t\ \mathbf{r}\ A).
\end{aligned}$$

Here are some examples. Consider the totality predicate T for $\mathbf{N}$ inductively defined by the clauses

$$T0, \qquad \forall^{\mathrm{nc}}_n(Tn \to^{\mathrm{c}} T(\mathrm{S}n)).$$

More precisely $T := \mu_X(K_0, K_1)$ with $K_0 := X0$, $K_1 := \forall^{\mathrm{nc}}_n(Xn \to^{\mathrm{c}} X(\mathrm{S}n))$. These clauses have types $\kappa_0 := \tau(K_0) = \tau(X0) = \xi$ and $\kappa_1 := \tau(K_1) = \tau(\forall^{\mathrm{nc}}_n(Xn \to^{\mathrm{c}} X(\mathrm{S}n))) = \xi \to \xi$. Therefore the algebra of witnesses is $\iota_T := \mu_\xi(\xi, \xi \to \xi)$, that is, $\mathbf{N}$ again. The witnessing predicate $T^{\mathbf{r}}$ is defined by the clauses

$$T^{\mathbf{r}}00, \qquad \forall_{n,m}(T^{\mathbf{r}}mn \to T^{\mathbf{r}}(\mathrm{S}m, \mathrm{S}n))$$

and it has as its elimination scheme

$$\begin{aligned}
\forall^{\mathrm{nc}}_n \forall^{\mathrm{c}}_m (T^{\mathbf{r}}mn \to Q(0,0) \to^{\mathrm{c}}\ & \\
& \forall^{\mathrm{nc}}_{n,m}(T^{\mathbf{r}}mn \to Qmn \to^{\mathrm{c}} Q(\mathrm{S}m, \mathrm{S}n)) \to^{\mathrm{c}}\\
& Qmn.
\end{aligned}$$

As an example involving parameters, consider the formula $\exists^{\mathrm{d}}_x A$ with a c.r. formula A, and view $\exists^{\mathrm{d}}_x A$ as inductively defined by the clause

$$\forall^{\mathrm{c}}_x(A \to^{\mathrm{c}} \exists^{\mathrm{d}}_x A).$$

More precisely, $\mathrm{Ex}^{\mathrm{d}}(Y) := \mu_X(K_0)$ with $K_0 := \forall^{\mathrm{c}}_x(Yx^\rho \to^{\mathrm{c}} X)$. Then $\exists^{\mathrm{d}}_x A$ abbreviates $\mathrm{Ex}^{\mathrm{d}}(\{\, x^\rho \mid A \,\})$. The single clause has type $\kappa_0 := \tau(K_0) = \tau(\forall^{\mathrm{c}}_x(Yx^\rho \to^{\mathrm{c}} X)) = \rho \to \alpha \to \xi$. Therefore the algebra of witnesses is $\iota := \iota_{\exists^{\mathrm{d}}_x A} := \mu_\xi(\rho \to \alpha \to \xi)$, that is, $\rho \times \alpha$. We write $\langle x, u \rangle$ for the values

of the (only) constructor of ι, i.e., the pairing operator. The witnessing predicate $(\exists^{\mathrm{d}}_x A)^{\mathbf{r}}$ is defined by the clause $K^{\mathbf{r}}_0((\exists^{\mathrm{d}}_x A)^{\mathbf{r}}, \{\, x^\rho \mid A \,\}) :=$

$$\forall_{x,u}(u\ \mathbf{r}\ A \to (\exists^{\mathrm{d}}_x A)^{\mathbf{r}}\langle x, u\rangle)$$

and its elimination scheme is

$$\forall^{\mathrm{c}}_w((\exists^{\mathrm{d}}_x A)^{\mathbf{r}} w \to \forall^{\mathrm{nc}}_{x,u}(u\ \mathbf{r}\ A \to Q\langle x, u\rangle) \to^{\mathrm{c}} Qw).$$

Definition (Leibniz equality Eq and $\exists^{\mathrm{u}}$, $\wedge^{\mathrm{u}}$). The introduction axioms are

$$\forall^{\mathrm{nc}}_x \mathrm{Eq}(x, x), \qquad \forall^{\mathrm{nc}}_x(A \to^{\mathrm{nc}} \exists^{\mathrm{u}}_x A), \qquad A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} A \wedge^{\mathrm{u}} B,$$

and the elimination schemes are

$$\begin{aligned}
&\forall^{\mathrm{nc}}_{x,y}(\mathrm{Eq}(x, y) \to \forall^{\mathrm{nc}}_x Pxx \to^{\mathrm{c}} Pxy),\\
&\exists^{\mathrm{u}}_x A \to \forall^{\mathrm{nc}}_x(A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P,\\
&A \wedge^{\mathrm{u}} B \to (A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P.
\end{aligned}$$

An important property of the realizing formulas $t\ \mathbf{r}\ A$ is that they are *invariant*.

Proposition. *$\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ A)$ is the same formula as $t\ \mathbf{r}\ A$.*

Proof. By induction on the simultaneous inductive definition of formulas and predicates in 3.1.2.

Case $t\ \mathbf{r}\ I\vec{s}$. By definition the formulas $\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ I\vec{s})$, $\varepsilon\ \mathbf{r}\ I^{\mathbf{r}}t\vec{s}$, $I^{\mathbf{r}}t\vec{s}$ and $t\ \mathbf{r}\ I\vec{s}$ are identical.

Case $I^{\mathbf{r}}t\vec{s}$. By definition $\varepsilon\ \mathbf{r}\ (\varepsilon\ \mathbf{r}\ I^{\mathbf{r}}t\vec{s})$ and $\varepsilon\ \mathbf{r}\ I^{\mathbf{r}}t\vec{s}$ are identical.

Case $\mathrm{Eq}(t, s)$. By definition $\varepsilon\ \mathbf{r}\ (\varepsilon\ \mathbf{r}\ (\mathrm{Eq}(t, s)))$ and $\varepsilon\ \mathbf{r}\ (\mathrm{Eq}(t, s))$ are identical.

Case $\exists^{\mathrm{u}}_x A$. The following formulas are identical.

$$\begin{aligned}
&\varepsilon\ \mathbf{r}\ (\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_x A),\\
&\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_x \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ A),\\
&\exists^{\mathrm{u}}_x(\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ A)),\\
&\exists^{\mathrm{u}}_x \exists^{\mathrm{u}}_y(\varepsilon\ \mathbf{r}\ (y\ \mathbf{r}\ A)),\\
&\exists^{\mathrm{u}}_x \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ A) \quad \text{by induction hypothesis},\\
&\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_x A.
\end{aligned}$$

Case $A \wedge^{\mathrm{u}} B$. The following formulas are identical.

$$\begin{aligned}
&\varepsilon\ \mathbf{r}\ (\varepsilon\ \mathbf{r}\ (A \wedge^{\mathrm{u}} B)),\\
&\varepsilon\ \mathbf{r}\ (\exists^{\mathrm{u}}_x(x\ \mathbf{r}\ A) \wedge^{\mathrm{u}} \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ B)),
\end{aligned}$$

$$\varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_x(x\ \mathbf{r}\ A) \wedge^{\mathrm{u}} \varepsilon\ \mathbf{r}\ \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ B)),$$
$$\exists^{\mathrm{u}}_x(\varepsilon\ \mathbf{r}\ (x\ \mathbf{r}\ A)) \wedge^{\mathrm{u}} \exists^{\mathrm{u}}_y(\varepsilon\ \mathbf{r}\ (y\ \mathbf{r}\ B)),$$
$$\exists^{\mathrm{u}}_x(x\ \mathbf{r}\ A) \wedge^{\mathrm{u}} \exists^{\mathrm{u}}_y(y\ \mathbf{r}\ B) \quad \text{by induction hypothesis},$$
$$\varepsilon\ \mathbf{r}\ (A \wedge^{\mathrm{u}} B).$$

Case $A \to^{\mathrm{c}} B$. The following formulas are identical.

$$\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ (A \to^{\mathrm{c}} B)),$$
$$\varepsilon\ \mathbf{r}\ \forall_x(x\ \mathbf{r}\ A\ \to\ tx\ \mathbf{r}\ B),$$
$$\forall_x(\varepsilon\ \mathbf{r}\ (x\ \mathbf{r}\ A)\ \to\ \varepsilon\ \mathbf{r}\ (tx\ \mathbf{r}\ B)),$$
$$\forall_x(x\ \mathbf{r}\ A\ \to\ tx\ \mathbf{r}\ B) \quad \text{by induction hypothesis},$$
$$t\ \mathbf{r}\ (A \to^{\mathrm{c}} B).$$

Case $A \to^{\mathrm{nc}} B$. The following formulas are identical.

$$\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ (A \to^{\mathrm{nc}} B)),$$
$$\varepsilon\ \mathbf{r}\ \forall_x(x\ \mathbf{r}\ A\ \to\ t\ \mathbf{r}\ B),$$
$$\forall_x(\varepsilon\ \mathbf{r}\ (x\ \mathbf{r}\ A)\ \to\ \varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ B)),$$
$$\forall_x(x\ \mathbf{r}\ A\ \to\ t\ \mathbf{r}\ B) \quad \text{by induction hypothesis},$$
$$t\ \mathbf{r}\ (A \to^{\mathrm{nc}} B).$$

Case $\forall^{\mathrm{c}}_x A$. The following formulas are identical.

$$\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ \forall^{\mathrm{c}}_x A),$$
$$\varepsilon\ \mathbf{r}\ \forall_x(tx\ \mathbf{r}\ A),$$
$$\forall_x(\varepsilon\ \mathbf{r}\ (tx\ \mathbf{r}\ A)),$$
$$\forall_x(tx\ \mathbf{r}\ A) \quad \text{by induction hypothesis},$$
$$t\ \mathbf{r}\ \forall^{\mathrm{c}}_x A.$$

Case $\forall^{\mathrm{nc}}_x A$. The following formulas are identical.

$$\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A),$$
$$\varepsilon\ \mathbf{r}\ \forall_x(t\ \mathbf{r}\ A),$$
$$\forall_x(\varepsilon\ \mathbf{r}\ (t\ \mathbf{r}\ A)),$$
$$\forall_x(t\ \mathbf{r}\ A) \quad \text{by induction hypothesis},$$
$$t\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A.$$

This completes the proof. □

3.2.6. Extracted terms. For a derivation M of a formula A we define its *extracted term* $\mathrm{et}(M)$, of type $\tau(A)$. This definition is relative to a fixed assignment of object variables to assumption variables: to every assumption variable u^A for a formula A we assign an object variable x_u of type $\tau(A)$.

Definition (Extracted term $\mathrm{et}(M)$ of a derivation M). For derivations M^A with A n.c. let $\mathrm{et}(M^A) := \varepsilon$. Otherwise

$$
\begin{aligned}
\mathrm{et}(u^A) &:= x_u^{\tau(A)} \quad (x_u^{\tau(A)} \text{ uniquely associated with } u^A),\\
\mathrm{et}((\lambda_{u^A} M^B)^{A\to^{\mathrm{c}}B}) &:= \lambda_{x_u^{\tau(A)}}\mathrm{et}(M),\\
\mathrm{et}((M^{A\to^{\mathrm{c}}B}N^A)^B) &:= \mathrm{et}(M)\mathrm{et}(N),\\
\mathrm{et}((\lambda_{x^\rho} M^A)^{\forall^{\mathrm{c}}_x A}) &:= \lambda_{x^\rho}\mathrm{et}(M),\\
\mathrm{et}((M^{\forall^{\mathrm{c}}_x A(x)}r)^{A(r)}) &:= \mathrm{et}(M)r,\\
\mathrm{et}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}}B}) &:= \mathrm{et}(M),\\
\mathrm{et}((M^{A\to^{\mathrm{nc}}B}N^A)^B) &:= \mathrm{et}(M),\\
\mathrm{et}((\lambda_{x^\rho} M^A)^{\forall^{\mathrm{nc}}_x A}) &:= \mathrm{et}(M),\\
\mathrm{et}((M^{\forall^{\mathrm{nc}}_x A(x)}r)^{A(r)}) &:= \mathrm{et}(M).
\end{aligned}
$$

Here $\lambda_{x_u^{\tau(A)}}\mathrm{et}(M)$ means just $\mathrm{et}(M)$ if A is n.c.

It remains to define extracted terms for the axioms. Consider a (c.r.) inductively defined predicate I. For its introduction axioms (3.3) and elimination axiom (3.4) define $\mathrm{et}(I_i^+) := \mathrm{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where both the constructor C_i and the recursion operator $\mathcal{R}$ refer to the algebra ι_I associated with I.

Now consider the special non-computational inductively defined predicates. Since they are n.c., we only need to define extracted terms for their elimination axioms. For the witnessing predicate $I^{\mathbf{r}}$ we define $\mathrm{et}((I^{\mathbf{r}})^-) := \mathcal{R}$ (referring to the algebra ι_I again), and for Leibniz equality Eq, the n.c. existential quantifier $\exists^{\mathrm{u}}_x A$ and conjunction $A \wedge^{\mathrm{u}} B$ we take identities of the appropriate type.

Remark. If derivations M are defined simultaneously with their extracted terms $\mathrm{et}(M)$, we can formulate the introduction rules for $\to^{\mathrm{nc}}$ and $\forall^{\mathrm{nc}}$ by

(i) If M^B is a derivation and $x_{u^A} \notin \mathrm{FV}(\mathrm{et}(M))$, then $(\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}}B}$ is a derivation.

(ii) If M^A is a derivation, x is not free in any formula of a free assumption variable of M and $x \notin \mathrm{FV}(\mathrm{et}(M))$, then $(\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}$ is a derivation.

3.2.7. Soundness. One can prove that every theorem in $\mathrm{TCF} + \mathrm{Ax}_{\mathrm{nci}}$ has a realizer: the extracted term of its proof. Here $(\mathrm{Ax}_{\mathrm{nci}})$ is an arbitrary set of non-computational invariant formulas viewed as axioms.

Theorem (Soundness). *Let M be a derivation of A from assumptions $u_i \colon C_i$ $(i < n)$. Then we can derive* $\mathrm{et}(M)$ $\mathbf{r}$ *A from assumptions x_{u_i}* $\mathbf{r}$ *C_i (with $x_{u_i} := \varepsilon$ in case C_i is n.c.).*

For the proof is (by induction on M) we have to refer to the literature.