COMPUTATIONAL ASPECTS OF BISHOP'S CONSTRUCTIVE MATHEMATICS

HELMUT SCHWICHTENBERG

Abstract. We view the Scott-Ershov type structure $(\mathfrak{C}_\tau)_\tau$ of partial continuous functionals as an appropriate model for Bishop's constructive mathematics. It allows infinite objects as data (i.e., base type objects). Pointwise equality is defined by induction on types, and an object is called extensional if it is pointwise equal to itself. A formal theory TCF describing $(\mathfrak{C}_\tau)_\tau$ is sketched, with inductive and coinductive predicates as its main ingredient. Using realizability we define the computational content of a formal proof M as a term $\mathrm{et}(M)$. We prove that $\mathrm{et}(M)$ realizes the end formula of M (soundness theorem), and that $\mathrm{et}(M)$ is extensional. Since $\mathrm{et}(M)$ is in TCF's language we can formally prove the soundness theorem.

Keywords: Program extraction, realizability, extensionality
2010 Mathematics Subject Classification: 03B35, 03F50, 03B70

Constructive mathematics, as its name says, puts an emphasis on the constructions involved in mathematical arguments. It therefore is tempting to have a closer look at these constructions, as advocated in Bishop (1970). The goal would be to extract them from mathematical proofs. Then one can view them as programs, obtained not by a programming effort but by searching for a perspicious mathematical argument, which of course has to use constructive logic. Clearly this requires formal proofs, but the benefit is that the program arising from a proof that can be machine checked for its correctness. In this sense we have a "certified" program.

One can even go a step further and ask for a formal proof that the extracted program correctly solves the original problem proved constructively. This can be expressed by the notion of realizability; see Troelstra (1998) for a survey on this subject. We then need to view the extracted program as a term in a formal language (an extension of the system T of Gödel (1958)), and give a formal proof that the term t realizes the formula A, written $t \mathbf{r} A$. Constructively to state A in a sense means the same as to say that A has a realizer. This statement $A \leftrightarrow \exists_x (x \mathbf{r} A)$ was called "to assert is to realize" in Feferman (1979). Here we call it invariance axiom, since it expresses invariance of A under the realizability interpretation. Using the invariance axioms one can prove a soundness theorem, saying that for any proof M of a formula A one can find another proof that the term $\mathrm{et}(M)$ extracted from the proof M is s realizer of A, i.e., $\mathrm{et}(M) \mathbf{r} A$. This step can be seen as a kind of reflection of what was done in the original (realizability-free) problem area. In this way we obtain a higher degree of reliability of the extracted term viewed as a program. We not only know that it came from a formal proof, but can even provide another formal soundness proof stating that the program satisfies its specification.

In the present paper we describe the main steps to carry this program out. It involves the setup of an appropriate theoretical framework TCF (theory of computable functionals). To ensure that TCF is a meaningful theory it is designed to describe a particular model suitable to deal with computable higher type objects. For case studies we use the proof assistant Minlog[1] designed to support the generation of formal proofs in TCF.

1. Partial continuous functionals

Prior to the setup of a formal theory we define the model our theory is supposed to describe. It will be a model accomodating higher type objects, from a constructive point of view. The main idea is to view an object of an arbitrary type as given by its finite approximations. This approach has the advantage that the notion of computability of our functional objects is unproblematic: it means that the set of its finite approximations can be enumerated by an elementary function.

To allow for applications in exact real arithmetic with real numbers represented as streams of signed digits we admit infinite data already at base types. A benefit of this approach is that it brings down the type level of other concepts of constructive analysis, for instance continuity of real functions.

1.1. **Information systems.**

We aim at describing higher type functionals by their finite approximations. For this purpose we use Dana Scott's information systems. The basic idea is to provide an axiomatic setting to describe approximations of abstract objects (like functions or functionals) by concrete, finite ones. We take an arbitrary countable set A of "bits of data" or *tokens* as a basic notion to be explained axiomatically. In order to use such data to build approximations of abstract objects, we need a notion of *consistency*, which determines when the elements of a finite set of tokens are consistent with each other. We also need an *entailment* relation between consistent finite sets U of data and single tokens a, which intuitively expresses the fact that the information contained in U is sufficient to compute the bit of information a. The axioms below are a minor modification of Scott's (1982), due to Larsen and Winskel (1991).

Definition. An *information system* is a structure $(A, \mathrm{Con}, \vdash)$ where A is an at most countable non-empty set (the *tokens*), Con is a set of finite subsets of A (the *consistent* sets) and $\vdash$ is a subset of $\mathrm{Con} \times A$ (the *entailment* relation), which satisfy

$$\begin{aligned}
&U \subseteq V \in \mathrm{Con} \to U \in \mathrm{Con},\\
&\{a\} \in \mathrm{Con},\\
&U \vdash a \to U \cup \{a\} \in \mathrm{Con},\\
&a \in U \in \mathrm{Con} \to U \vdash a,\\
&U \in \mathrm{Con} \to \forall_{a \in V}(U \vdash a) \to V \vdash b \to U \vdash b.
\end{aligned}$$

The elements of Con are called *formal neighborhoods*. We use U, V, W to denote *finite* sets, and write

$$U \vdash V \quad \text{for} \quad U \in \mathrm{Con} \wedge \forall_{a \in V}(U \vdash a),$$

[1] `http://minlog-system.de`.

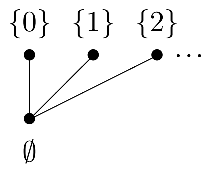
$$\begin{aligned} &a \uparrow b && \text{for} \quad \{a,b\} \in \mathrm{Con} && (a, b \text{ are } \textit{consistent}),\\ &U \uparrow V && \text{for} \quad \forall_{a \in U, b \in V}(a \uparrow b). \end{aligned}$$

Definition (Objects)**.** The *objects* (or *ideals*) of an information system $\boldsymbol{A} = (A, \mathrm{Con}, \vdash)$ are defined to be those subsets x of A which satisfy

$$\begin{aligned} &U \subseteq x \to U \in \mathrm{Con} \quad (x \text{ is } \textit{consistent}),\\ &U \vdash a \to U \subseteq x \to a \in x \quad (x \text{ is } \textit{deductively closed}). \end{aligned}$$

For example the *deductive closure* $\overline{U} := \{\, a \in A \mid U \vdash a \,\}$ of $U \in \mathrm{Con}$ is an object. The set of all objects of $\boldsymbol{A}$ is denoted by $|\boldsymbol{A}|$.

Example. Every countable set A can be turned into a "flat" information system by letting the set of tokens be A, $\mathrm{Con} := \{\emptyset\} \cup \{\, \{a\} \mid a \in A \,\}$ and $U \vdash a$ mean $a \in U$. In this case the objects are just the elements of Con. For $A = \mathbb{N}$ we have the following picture of the Con-sets.



The main feature of information systems is that they admit the construction of function spaces.

Definition. Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Define $\boldsymbol{A} \to \boldsymbol{B} = (C, \mathrm{Con}, \vdash)$ by

$$C := \mathrm{Con}_A \times B,$$

$$\{\, (U_i, b_i) \mid i \in I \,\} \in \mathrm{Con} := \forall_{J \subseteq I}\Bigl(\bigcup_{j \in J} U_j \in \mathrm{Con}_A \to \{\, b_j \mid j \in J \,\} \in \mathrm{Con}_B\Bigr).$$

For the definition of the entailment relation $\vdash$ it is helpful to first define the notion of an *application* of $W := \{\, (U_i, b_i) \mid i \in I \,\} \in \mathrm{Con}$ to $U \in \mathrm{Con}_A$:

$$\{\, (U_i, b_i) \mid i \in I \,\}U := \{\, b_i \mid U \vdash_A U_i \,\}.$$

From the definition of Con we know that this set is in Con_B. Now define $W \vdash (U, b)$ by $WU \vdash_B b$.

Remark. Clearly application is *monotone in the second argument*, in the sense that $U \vdash_A U'$ implies ($WU' \subseteq WU$, hence also) $WU \vdash_B WU'$. In fact, application is also *monotone in the first argument*, i.e.,

$$W \vdash W' \quad \text{implies} \quad WU \vdash_B W'U.$$

To see this let $W = \{\, (U_i, b_i) \mid i \in I \,\}$ and $W' = \{\, (U'_j, b'_j) \mid j \in J \,\}$. By definition $W'U = \{\, b'_j \mid U \vdash_A U'_j \,\}$. Now fix j such that $U \vdash_A U'_j$; we must show $WU \vdash_B b'_j$. By assumption $W \vdash (U'_j, b'_j)$, hence $WU'_j \vdash_B b'_j$. Because of $WU \supseteq WU'_j$ the claim follows.

Lemma 1.1. *If $\boldsymbol{A}$ and $\boldsymbol{B}$ are information systems, then so is $\boldsymbol{A} \to \boldsymbol{B}$.*

Lemma 1.2. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be information systems and $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ monotone (i.e., $x \subseteq y \to f(x) \subseteq f(y)$). Then the following are equivalent.*

(a) *f satisfies the "principle of finite support" PFS: If $b \in f(x)$, then $b \in f(\overline{U})$ for some $U \subseteq x$.*
(b) *f commutes with directed unions: for every directed $D \subseteq |\boldsymbol{A}|$ (i.e., for any $x, y \in D$ there is a $z \in D$ such that $x, y \subseteq z$)*
$$f\Big(\bigcup_{x\in D} x\Big) = \bigcup_{x\in D} f(x).$$

Note that in (b) the set $\{ f(x) \mid x \in D \}$ is directed by monotonicity of f; hence its union is indeed an object in $|\boldsymbol{B}|$. Also from PFS and monotonicity of f it follows that if $V \subseteq f(x)$, then $V \subseteq f(\overline{U})$ for some $U \subseteq x$.

We call a function $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ continuous if it satifies the conditions in Lemma 1.2. Hence continuous maps $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract objects $x \in |\boldsymbol{A}|$ and $f(x) \in |\boldsymbol{B}|$: whenever we are given a finite approximation V to the value $f(x)$, then there is a finite approximation U to the argument x such that already $f(\overline{U})$ contains the information in V; note that by monotonicity $f(\overline{U}) \subseteq f(x)$.

Clearly the identity and constant functions are continuous, and also the *composition* $g \circ f$ of continuous functions $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ and $g\colon |\boldsymbol{B}| \to |\boldsymbol{C}|$.

Theorem 1.3. *Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Then the objects of $\boldsymbol{A} \to \boldsymbol{B}$ are in a natural bijective correspondence with the continuous functions from $|\boldsymbol{A}|$ to $|\boldsymbol{B}|$, as follows.*
(a) *With any object x of $\boldsymbol{A} \to \boldsymbol{B}$ we can associate a continuous function $|x|\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ by*
$$|x|(z) := \{ b \in B \mid (U, b) \in x \text{ for some } U \subseteq z \}.$$
We call $|x|(z)$ the application *of x to z.*
(b) *Conversely, with any continuous function $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ we can associate an object $\hat{f}$ of $\boldsymbol{A} \to \boldsymbol{B}$ by*
$$\hat{f} := \{ (U, b) \mid b \in f(\overline{U}) \}.$$
These assignments are inverse to each other, i.e., $f = |\hat{f}|$ and $x = \widehat{|x|}$.

1.2. **Algebras and types.** We now consider concrete information systems, our basis for continuous functionals. Types will be built from base types by the formation of function types, $\tau \to \sigma$. As domains for the base types we choose non-flat free algebras, given by their constructors. The reason for taking non-flat base domains is that we want the constructors to be injective and with disjoint ranges. This generally is not the case for flat domains.

Definition (Constructor types and algebra forms)**.** *Constructor types κ* have the form
$$\vec{\alpha} \to (\xi)_{i<n} \to \xi$$
with all type variables α_i distinct from each other and from ξ. Iterated arrows are understood as associated to the right. An argument type of a constructor type is called a *parameter* argument type if it is different from ξ, and a *recursive* argument type otherwise. A constructor type κ is *nullary* if it has no recursive argument types. We call
$$\iota := \mu_\xi \vec{\kappa}$$

with $\vec{\kappa}$ not empty an *algebra form*. An algebra form is *explicit* if it does not have recursive argument types.

Examples. We list some parameter-free algebra forms, with standard names for the constructors added to each constructor type.

$\mathbb{U} := \mu_\xi(\mathrm{Dummy}\colon \xi)$	(unit),
$\mathbb{B} := \mu_\xi(\mathtt{tt}\colon \xi, \mathtt{ff}\colon \xi)$	(booleans),
$\mathbb{N} := \mu_\xi(0\colon \xi, \mathcal{S}\colon \xi \to \xi)$	(natural numbers, unary),
$\mathbb{P} := \mu_\xi(1\colon \xi, \mathcal{S}_0\colon \xi \to \xi, \mathcal{S}_1\colon \xi \to \xi)$	(positive numbers, binary),
$\mathbb{Y} := \mu_\xi(-\colon \xi, \mathrm{Branch}\colon \xi \to \xi \to \xi)$	(binary trees).

Algebra forms with type parameters are

$\mathbb{I}(\alpha)$	$:= \mu_\xi(\mathrm{Id}\colon \alpha \to \xi)$	(identity),
$\mathbb{L}(\alpha)$	$:= \mu_\xi(\mathrm{Nil}\colon \xi, \mathrm{Cons}\colon \alpha \to \xi \to \xi)$	(lists),
$\mathbb{S}(\alpha)$	$:= \mu_\xi(\mathrm{SCons}\colon \alpha \to \xi \to \xi)$	(streams),
$\alpha \times \beta$	$:= \mu_\xi(\mathrm{Pair}\colon \alpha \to \beta \to \xi)$	(product),
$\alpha + \beta$	$:= \mu_\xi(\mathrm{InL}\colon \alpha \to \xi, \mathrm{InR}\colon \beta \to \xi)$	(sum),
$\mathrm{uysum}(\alpha)$	$:= \mu_\xi(\mathrm{DummyL}\colon \xi, \mathrm{Inr}\colon \alpha \to \xi)$	(for $\mathbb{U} + \alpha$),
$\mathrm{ysumu}(\alpha)$	$:= \mu_\xi(\mathrm{Inl}\colon \alpha \to \xi, \mathrm{DummyR}\colon \xi)$	(for $\alpha + \mathbb{U}$).

The default name for the i-th constructor of an algebra form is $\mathcal{C}_i$.

Definition (Type)**.**

$$\rho, \sigma, \tau ::= \alpha \mid \iota(\vec{\rho}) \mid \tau \to \sigma,$$

where ι is an algebra form with $\vec{\alpha}$ its parameter type variables, and $\iota(\vec{\rho})$ the result of substituting the (already generated) types $\vec{\rho}$ for $\vec{\alpha}$. Types of the form $\iota(\vec{\rho})$ are called *algebras*. An algebra is *closed* if it has no type variables. The *level* of a type is defined by

$$\mathrm{lv}(\alpha) := 0, \quad \mathrm{lv}(\iota(\vec{\rho})) := \max(\mathrm{lv}(\vec{\rho})), \quad \mathrm{lv}(\tau \to \sigma) := \max(\mathrm{lv}(\sigma), 1 + \mathrm{lv}(\tau)).$$

Base types are types of level 0, and a *higher* type has level at least 1.

1.3. **The model $(\mathfrak{C}_\tau)_\tau$.** For every closed type τ we define an information system $\boldsymbol{C}_\tau = (C_\tau, \mathrm{Con}_\tau, \vdash_\tau)$. The definition is by induction on τ, and in case of an algebra $\iota(\vec{\rho})$ by a side inductive definition.

Definition (Information system of type τ)**.** *Case* $\iota(\tau)$. For simplicity assume that there is only one parameter type τ.

(a) *Tokens* $a \in C_{\iota(\tau)}$ are the type correct constructor expressions $\mathcal{C}V a_1^* \dots a_n^*$ where a_i^* is an *extended token*, i.e., a token or the special symbol $*$ which carries no information, and V is a consistent set of tokens in C_τ.

(b) A finite set U of tokens in $C_{\iota(\tau)}$ is *consistent* (i.e., $\in \mathrm{Con}_{\iota(\tau)}$) if all its elements start with the same constructor $\mathcal{C}$, say of arity $\tau \to \iota(\tau) \dots \to \iota(\tau) \to \iota(\tau)$. Let $U = \{\mathcal{C}V_1 a_{11}^* \dots a_{1n}^*, \dots, \mathcal{C}V_m a_{m1}^* \dots a_{mn}^*\}$. Then we require that (i) $V_1 \cup \dots \cup V_m$ is consistent (i.e., $\in \mathrm{Con}_\tau$) and (ii) the sets U_i consisting of all (proper) tokens at the i-th argument position of some token in U are consistent (i.e., $\in \mathrm{Con}_{\iota(\tau)}$).

(c) $\{\mathcal{C}V_1a^*_{11}\dots a^*_{1n},\dots,\mathcal{C}V_ma^*_{m1}\dots a^*_{mn}\} \vdash_{\iota(\tau)} \mathcal{C}Va^*_1\dots a^*_n$ if and only if (i) $V_1\cup\dots\cup V_m \vdash_\tau V$ and (ii) for each set U_i as in (b) above we have $U_i \vdash_{\iota(\tau)} a^*_i$ (where $U_i \vdash *$ is taken to be true).

Case $\tau\to\sigma$. Tokens, consistency and entailment for $\boldsymbol{C}_{\tau\to\sigma} := \boldsymbol{C}_\tau \to \boldsymbol{C}_\sigma$ are defined as done in Section 1.1 for arbitrary information systems.

Lemma 1.4. $\boldsymbol{C}_\tau := (C_\tau, \mathrm{Con}_\tau, \vdash_\tau)$ *is an information system.*

Definition. The objects $x\in|\boldsymbol{C}_\tau|$ are called *partial continuous functionals* of type τ. Since $\boldsymbol{C}_{\tau\to\sigma} = \boldsymbol{C}_\tau\to\boldsymbol{C}_\sigma$, the partial continuous functionals of type $\tau\to\sigma$ correspond to the continuous functions from $|\boldsymbol{C}_\tau|$ to $|\boldsymbol{C}_\sigma|$. A partial continuous functional $x\in|\boldsymbol{C}_\tau|$ is *computable* if it is recursively enumerable when viewed as a set of tokens. The *Scott-Ershov model* $\mathfrak{C}$ of partial continuous functionals is defined to be $(|\boldsymbol{C}_\tau|)_\tau$.

Definition (Cototal and total objects of closed base type)**.** Let $\iota(\vec{\tau})$ be a closed base type. Its tokens can be seen as constructor trees with some recursive argument positions occupied by $*$. An object x is *cototal* if for each of its tokens $P(*)$ with a distinguished occurrence of $*$ there is another token of the form $P(\mathcal{C}\vec{\emptyset}\vec{*})$ in x. Finite cototal objects are called *total.*

1.4. **Cototality and bisimilarity.** For closed ground types equality of cototal objects can be characterized by *bisimilarity*. As an example we consider the algebra $\mathbb{Y}$ of binary trees. We define bisimilarity $\approx_\mathbb{Y}$ as the largest relation on $\mathcal{C}_\mathbb{Y}$ satisfying the *closure* axiom $\approx^-_\mathbb{Y}$:

$$\forall_{x,x'}(x\approx x' \to (x\equiv - \wedge x'\equiv -)\vee$$
$$\exists_{x_1,x_2,x'_1,x'_2}(x_1\approx x'_1 \wedge x_2\approx x'_2 \wedge x\equiv \mathcal{C}x_1x_2 \wedge x'\equiv \mathcal{C}x'_1x'_2))$$

with $\mathcal{C}$ for the Branch constructor. Being the "largest" relation means that any other relation ("competitor") X satisfying the same closure property is below $\approx_\mathbb{Y}$, i.e., we require the *greatest-fixed-point* property $\approx^+_\mathbb{Y}$:

$$\forall_{x,x'}(Xxx' \to (x\equiv - \wedge x'\equiv -)\vee$$
$$\exists_{x_1,x_2,x'_1,x'_2}((x_1\approx x'_1 \vee Xx_1x'_1)\wedge(x_2\approx x'_2\vee Xx_2x'_2)\wedge$$
$$x\equiv\mathcal{C}x_1x_2\wedge x'\equiv\mathcal{C}x'_1x'_2)))\to$$
$$X\subseteq\approx.$$

Lemma 1.5 (Bisimilarity)**.** $x\approx_\mathbb{Y} x'$ *implies* $x\equiv x'$*, for* $x,x'\in\mathcal{C}_\mathbb{Y}$*.*

Proof. Let a range over tokens for $\mathbb{Y}$, and define the *height* $|a^*|$ of an extended token a^* by $|*| := 0$, $|-| := 1$, $|\mathcal{C}a^*_1a^*_2| := 1+\max(|a^*_1|,|a^*_2|)$. By induction on the height $|a^*|$ of extended tokens a^* we prove that for all objects x,x' and extended tokens $a^*\in x$ we have $a^*\in x'$. It suffices to consider the case $\mathcal{C}a^*_1a^*_2$. From $x\approx_\mathbb{Y} x'$ we obtain by the closure axiom x_1,x_2,x'_1,x'_2 with

$$x_1\approx x'_1\wedge x_2\approx x'_2\wedge x\equiv\mathcal{C}x_1x_2\wedge x'\equiv\mathcal{C}x'_1x'_2.$$

Then $a^*_i\in x_i$ (for $i=1,2$), and by IH $a^*_i\in x'_i$. Thus $\mathcal{C}a^*_1a^*_2\in x'$. □

From Lemma 1.5 we obtain the following characterization of $\approx_\mathbb{Y}$ on $\mathcal{C}_\mathbb{Y}$. We define ${}^{\mathrm{co}}T_\mathbb{Y}$ as the largest subset of $\mathcal{C}_\mathbb{Y}$ satisfying the *closure* axiom ${}^{\mathrm{co}}T^-_\mathbb{Y}$:

$$\forall_x(x\in{}^{\mathrm{co}}T\to x\equiv -\vee\exists_{x_1,x_2}(x_1\in{}^{\mathrm{co}}T\wedge x_2\in{}^{\mathrm{co}}T\wedge x\equiv\mathcal{C}x_1x_2)).$$

Again we require the *greatest-fixed-point* property ${}^{\mathrm{co}}T^{+}_{\mathbb{Y}}$:

$$\begin{aligned}&\forall_x(x \in X \to (x \equiv -) \lor \\ &\qquad\qquad \exists_{x_1,x_2}(x_1 \in {}^{\mathrm{co}}T \cup X \land x_2 \in {}^{\mathrm{co}}T \cup X \land x \equiv \mathcal{C}x_1x_2)) \to \\ &X \subseteq {}^{\mathrm{co}}T.\end{aligned}$$

For objects $x, x' \in \mathcal{C}_{\mathbb{Y}}$ we show

Lemma 1.6 (Characterization of $\approx_{\mathbb{Y}}$)**.**

$$x \approx_{\mathbb{Y}} x' \leftrightarrow x, x' \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x \equiv x', \textit{ for } x, x' \in \mathcal{C}_{\mathbb{Y}}.$$

Proof. "$\to$". By Lemma 1.5 it remains to prove $x \approx_{\mathbb{Y}} x' \to x \in {}^{\mathrm{co}}T_{\mathbb{Y}}$. To this end we apply ${}^{\mathrm{co}}T^{+}_{\mathbb{Y}}$ with competitor $X := \{\, x \mid \exists_{x'}(x \approx_{\mathbb{Y}} x') \,\}$. It suffices to prove the premise. Fix x, x' with $x \approx_{\mathbb{Y}} x'$. The goal is

$$\begin{aligned}&(x \equiv -) \lor \\ &\exists_{x_1,x_2}((x_1 \in {}^{\mathrm{co}}T \lor \exists_{x'_1}(x_1 \approx x'_1)) \land (x_2 \in {}^{\mathrm{co}}T \lor \exists_{x'_2}(x_2 \approx x'_2)) \land x \equiv \mathcal{C}x_1x_2).\end{aligned}$$

By the closure property $\approx^{-}_{\mathbb{Y}}$ we have

$$(x \equiv - \land x' \equiv -) \lor \exists_{x_1,x_2,x'_1,x'_2}(x_1 \approx x'_1 \land x_2 \approx x'_2 \land x \equiv \mathcal{C}x_1x_2 \land x' \equiv \mathcal{C}x'_1x'_2)).$$

In the first case we have $x \equiv -$ and are done. In the second case we have x_1, x_2, x'_1, x'_2 with $x_1 \approx x'_1$, $x_2 \approx x'_2$ and $x \equiv \mathcal{C}x_1x_2$, and are done as well.

"$\leftarrow$". We prove $x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \to x \equiv x' \to x \approx_{\mathbb{Y}} x'$ by the greatest-fixed-point property $\approx^{+}_{\mathbb{Y}}$ with competitor $X := \{\, x, x' \mid x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x \equiv x' \,\}$. It suffices to prove the premise. Fix x, x' with $x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x \equiv x'$. The goal is

$$\begin{aligned}(x \equiv - \land x' \equiv -) \lor \exists_{x_1,x_2,x'_1,x'_2}((&x_1 \approx x'_1 \lor (x_1 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x_1 \equiv x'_1)) \land \\ &(x_2 \approx x'_2 \lor (x_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x_2 \equiv x'_2)) \land \\ &x \equiv \mathcal{C}x_1x_2 \land x' \equiv \mathcal{C}x'_1x'_2)).\end{aligned}$$

By the closure property ${}^{\mathrm{co}}T^{-}_{\mathbb{Y}}$ applied to $x \in {}^{\mathrm{co}}T_{\mathbb{Y}}$ we have

$$(x \equiv -) \lor \exists_{x_1,x_2}(x_1 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land x_2 \in {}^{\mathrm{co}}T \land x \equiv \mathcal{C}x_1x_2).$$

In the first case we have $x \equiv -$ and are done, since $x \equiv x'$. In the second case we have $x_1, x_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}}$ with $x \equiv \mathcal{C}x_1x_2$. Then we are done as well with $x'_1 := x_1$ and $x'_2 := x_2$, since again $x \equiv x'$. □

1.5. **Constructors as continuous functions.** Let ι be an algebra. Every constructor $\mathcal{C}$ generates the following object in the function space determined by the type of the constructor:

$$r_{\mathcal{C}} := \{\, (\vec{U}, \mathcal{C}\vec{a^*}\,) \mid \vec{U} \vdash \vec{a^*} \,\}.$$

Here $(\vec{U}, a)$ abbreviates $(U_1, (U_2, \dots (U_n, a) \dots))$.

According to the general definition of a continuous function associated to an object in a function space the continuous map $|r_{\mathcal{C}}|$ satisfies

$$|r_{\mathcal{C}}|(\vec{x}\,) = \{\, \mathcal{C}\vec{a^*} \mid \exists_{\vec{U} \subseteq \vec{x}}(\vec{U} \vdash \vec{a^*}) \,\}.$$

(For $\mathbb{N}$ we have $|r_{\mathcal{S}}|(\{0\}) = \{\mathcal{S}0, \mathcal{S}*\}$ and $|r_{\mathcal{S}}|(\{\mathcal{S}0, \mathcal{S}*\}) = \{\mathcal{S}\mathcal{S}0, \mathcal{S}\mathcal{S}*, \mathcal{S}*\}$.) An immediate consequence is that the (continuous maps corresponding to) constructors are injective and their ranges are disjoint, which is what we wanted to achieve by associating non-flat rather than flat information systems with algebras.

Lemma 1.7 (Constructors are injective and have disjoint ranges)**.** *Let ι be an algebra and $\mathcal{C}$ be a constructor of ι. Then*

$$|r_{\mathcal{C}}|(\vec{x}\,) \subseteq |r_{\mathcal{C}}|(\vec{y}\,) \leftrightarrow \vec{x} \subseteq \vec{y}.$$

If $\mathcal{C}_1, \mathcal{C}_2$ are distinct constructors of ι, then $|r_{\mathcal{C}_1}|(\vec{x}\,) \neq |r_{\mathcal{C}_2}|(\vec{y}\,)$, since the two objects are non-empty and disjoint.

Proof. Immediate from the definitions. □

Remark. Notice that neither property holds for flat information systems, since for them, by monotonicity, constructors need to be *strict* (i.e., if one argument is the empty object, then the value is as well). But then we have

$$|r_{\mathcal{C}}|(\overline{\emptyset}, y) = \overline{\emptyset} = |r_{\mathcal{C}}|(x, \overline{\emptyset}), \qquad |r_{\mathcal{C}_1}|(\overline{\emptyset}) = \overline{\emptyset} = |r_{\mathcal{C}_2}|(\overline{\emptyset})$$

where in the first case we have one binary and, in the second, two unary constructors.

2. A term language for computable functionals

We set up a system T^+ of typed terms, as an extension of Gödel's T (1958). Every closed term of type τ denotes an object of this type in the model $\mathfrak{C}$, i.e., a partial continuous functional. This is in contrast to Martin-Löf style type theories like Coq's calculus of inductive constructions, where terms must be total. Dropping this restriction has the advantage that non-terminating operators like corecursion can directly be represented as constants. We will define constants by equations, in a pattern-matching style.

2.1. Constants, terms and computation rules.

For every algebra $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi)_{i<k}$ we have constants

$\mathcal{C}_{\iota,i}$	$(\rho_{i\nu}(\iota))_{\nu<n_i} \to \iota$	i-th constructor
$\mathcal{R}_\iota^\alpha$	$\iota \to ((\rho_{i\nu}(\iota \times \alpha))_{\nu<n_i} \to \alpha)_{i<k} \to \alpha$	recursion
$\mathcal{D}_\iota$	${}^{\mathrm{co}}\iota \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}({}^{\mathrm{co}}\iota)$	destructor
${}^{\mathrm{co}}\mathcal{R}_\iota^\alpha$	$\alpha \to (\alpha \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}({}^{\mathrm{co}}\iota + \alpha)) \to {}^{\mathrm{co}}\iota$	corecursion.

It is convenient to write the type of the *recursion* operator $\mathcal{R}_{\mathbb{N}}^\tau$ in the form $\mathbb{N} \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \tau$. The first argument is the recursion argument, the second one gives the base value, and the third gives the step function, mapping the recursion argument and the previous value to the next value. The *destructor* $\mathcal{D}_\iota$ disassembles a constructor-built object into its parts. The *corecursion* operator ${}^{\mathrm{co}}\mathcal{R}_\iota^\tau$ is used to construct a map from τ to ${}^{\mathrm{co}}\iota$.

From the constants above, typed variables and possibly other typed constants D^τ we define *terms* by abstraction and application:

$$M, N ::= x^\tau \mid \mathcal{C}_{\iota,i} \mid \mathcal{R}_\iota \mid \mathcal{D}_\iota \mid {}^{\mathrm{co}}\mathcal{R}_\iota \mid D^\tau \mid (\lambda_{x^\tau} M^\sigma)^{\tau\to\sigma} \mid (M^{\tau\to\sigma} N^\tau)^\sigma.$$

For each term we want to define its denotation in the model $\mathfrak{C}$. To this end we use defining equations. Each constant C comes with a system of *computation rules* consisting of finitely many equations

$$C\vec{P}_i(\vec{y}_i) = M_i \qquad (i = 1, \dots, n \text{ where } n \geq 0) \tag{1}$$

with free variables of $\vec{P}_i(\vec{y}_i)$ and M_i among $\vec{y}_i$, where the arguments on the left hand side must be "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables. To ensure consistency of the defining equations, we require that for $i \neq j$ $\vec{P}_i$ and $\vec{P}_j$ have disjoint free variables, and either $\vec{P}_i$ and $\vec{P}_j$ are non-unifiable (i.e., there is no substitution which identifies them), or else for the "most general unifier" ϑ of $\vec{P}_i$ and $\vec{P}_j$ we have $M_i\vartheta = M_j\vartheta$. Notice that the substitution ϑ assigns to the variables $\vec{y}_i$ in M_i constructor patterns $\vec{R}_k(\vec{z})$ $(k = i, j)$. A further requirement on a system of computation rules $C\vec{P}_i(\vec{y}_i) = M_i$ is that the lengths of all $\vec{P}_i(\vec{y}_i)$ are the same; this number is called the *arity* of C, denoted by $\mathrm{ar}(C)$. A substitution instance of a left hand side of (1) is called a *C-redex*.

The computation rules for the constants $\mathcal{C}_{\iota,i}$, $\mathcal{R}_\iota$, $\mathcal{D}_\iota$ and ${}^{\mathrm{co}}\mathcal{R}_\iota$ are fixed, as follows. For the constructors no computation rules are necessary, since the model $\mathfrak{C}$ is built from them. For the recursion operator let

$$\alpha_0 \to \ldots \to \alpha_{m-1} \to (\xi)_{i<n} \to \xi$$

be the type of the i-th constructor $\mathcal{C}_i$ of ι and consider a term $\mathcal{C}_i\vec{x}$ of type ι. We write $\vec{x}^P = x_0^P, \ldots, x_{m-1}^P$ for the *parameter arguments* $x_0^{\alpha_0}, \ldots, x_{m-1}^{\alpha_{m-1}}$ and $\vec{x}^R = x_0^R, \ldots, x_{n-1}^R$ for the *recursive arguments* $x_m^\iota, \ldots, x_{m+n-1}^\iota$. Writing $\mathcal{R}$ for $\mathcal{R}_\iota^\tau$ we take as its computation rules

$$\mathcal{R}(\mathcal{C}_i\vec{x})\vec{f} = f_i\vec{x}(\mathcal{R}x_0^R\vec{f}\,)\ldots(\mathcal{R}x_{n-1}^R\vec{f}\,).$$

In particular $\mathcal{R}_\mathbb{N}^\tau$ is defined by the computation rules

$$\mathcal{R}_\mathbb{N}^\tau 0af = a, \qquad \mathcal{R}_\mathbb{N}^\tau(\mathcal{S}n)af = fn(\mathcal{R}_\mathbb{N}^\tau naf).$$

For example, $\mathcal{R}_\mathbb{N}^\mathbb{N} nm\lambda_{n,l}(\mathcal{S}l)$ defines addition $m+n$ by recursion on n. The computation rules for the *destructor* $\mathcal{D}_\iota$ are

$$\mathcal{D}_\iota(\mathcal{C}_i\vec{x}\,) = \langle\vec{x}\,\rangle.$$

To deal with *corecursion* we introduce some notation. For $f\colon \rho \to \tau$ and $g\colon \sigma \to \tau$ we denote $\lambda_x(\mathcal{R}_{\rho+\sigma}^\tau xfg)$ of type $\rho + \sigma \to \tau$ by $[f, g]$, and similary for ternary sumtypes etc. The identity functions id below is of type $\iota \to \iota$ with ι the respective algebra. The (single) computation rule for ${}^{\mathrm{co}}\mathcal{R}_\iota^\tau$ is

$${}^{\mathrm{co}}\mathcal{R}_\iota^\tau xf = [g_0, \ldots, g_{k-1}](fx)$$

where g_i of type $\prod_{\nu<n_i} \rho_{i\nu}(\iota + \tau) \to \iota$ is defined as

$$g_i := \lambda_{\vec{x}}(\mathcal{C}_i(N_\nu)_{\nu<n_i}) \qquad \text{with } x_\nu\colon \rho_{i\nu}(\iota+\tau),$$

$$N_\nu := \begin{cases} x_\nu & \text{if } \rho_{i\nu}(\xi) \text{ is a parameter arg. type,} \\ [\mathrm{id}^{\iota\to\iota}, \lambda_x({}^{\mathrm{co}}\mathcal{R}_\iota^\tau xf)]x_\nu^{\iota+\tau} & \text{otherwise.} \end{cases}$$

Remark. It can be difficult to read the computation rules for corecursion operators. However, it helps if we know some properties of the "step" function f. For instance we have

$${}^{\mathrm{co}}\mathcal{R}_\mathbb{N}^\tau xf = \begin{cases} 0 & \text{if } fx = \mathrm{DummyL}^{\mathbb{U}+(\mathbb{N}+\tau)} \\ \mathcal{S}n & \text{if } fx = \mathrm{Inr}(\mathrm{InL}^{\mathbb{N}\to\mathbb{N}+\tau}n) \\ \mathcal{S}({}^{\mathrm{co}}\mathcal{R}_\mathbb{N}^\tau x'f) & \text{if } fx = \mathrm{Inr}(\mathrm{InR}^{\tau\to\mathbb{N}+\tau}x') \end{cases}$$

$$ {}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}(\rho)}xf = \begin{cases} a :: u & \text{if } fx = \langle a, \mathrm{InL}^{\mathbb{S}(\rho)\to\mathbb{S}(\rho)+\tau}u\rangle \\ a :: {}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}(\rho)}x'f & \text{if } fx = \langle a, \mathrm{InR}^{\tau\to\mathbb{S}(\rho)+\tau}x'\rangle. \end{cases} $$

2.2. **Denotational semantics.** We set up a connection between the term system T^+ and the model $\mathfrak{C}$. The main point is to clarify how the computation rules define an object z in a function space. The idea is to inductively define the set of tokens (U, a) that make up z. It is convenient to define the value $[\![\lambda_{\vec{x}}M]\!]$, where M is a term with free variables among $\vec{x}$. Since this value is a token set, we can define inductively the relation $(\vec{U}, a) \in [\![\lambda_{\vec{x}}M]\!]$.

For a constructor pattern $\vec{P}(\vec{x}\,)$ and a list $\vec{V}$ of the same length and types as $\vec{x}$ we define a list $\vec{P}(\vec{V})$ of formal neighborhoods of the same length and types as $\vec{P}(\vec{x}\,)$, by induction on $\vec{P}(\vec{x}\,)$. $x(V)$ is the singleton list V, and for $\langle\rangle$ take the empty list. $(\vec{P}, Q)(\vec{V}, \vec{w}\,)$ is covered by induction, and

$$ (\mathcal{C}\vec{P}\,)(\vec{V}) := \{\, \mathcal{C}\vec{a} \mid a_i \in P_i(\vec{V}_i) \text{ if } P_i(\vec{V}_i) \neq \emptyset, \text{ and } a_i = * \text{ otherwise} \,\}. $$

We use the following notation. $(\vec{U}, a)$ means $(U_1, (U_2, \dots (U_n, a)) \dots)$, and $(\vec{U}, V) \subseteq [\![\lambda_{\vec{x}}M]\!]$ means $(\vec{U}, a) \in [\![\lambda_{\vec{x}}M]\!]$ for all (finitely many) $a \in V$.

Definition (Inductive, of $(\vec{U}, a) \in [\![\lambda_{\vec{x}}M]\!]$)**.**

$$ \frac{U_i \vdash a}{(\vec{U}, a) \in [\![\lambda_{\vec{x}}x_i]\!]}(V), \qquad \frac{(\vec{U}, V, a) \in [\![\lambda_{\vec{x}}M]\!] \qquad (\vec{U}, V) \subseteq [\![\lambda_{\vec{x}}N]\!]}{(\vec{U}, a) \in [\![\lambda_{\vec{x}}(MN)]\!]}(A). $$

For every constructor $\mathcal{C}$ and defined constant D we have

$$ \frac{\vec{V} \vdash \vec{a}}{(\vec{U}, \vec{V}, \mathcal{C}\vec{a}\,) \in [\![\lambda_{\vec{x}}\mathcal{C}]\!]}(\mathcal{C}), \qquad \frac{(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}M]\!] \qquad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, a) \in [\![\lambda_{\vec{x}}D]\!]}(D) $$

with one such rule (D) for every computation rule $D\vec{P}(\vec{y}\,) = M$.

This "denotational semantics" has good properties (see Schwichtenberg and Wainer (2012, pp.279–287)): $[\![\lambda_{\vec{x}}M]\!]$ is an object in the model, and the definition above of the denotation of a term is reasonable in the sense that it is not changed by an application of the standard (β- and η-) conversions or a computation rule.

3. A theory of computable functionals

After getting clear about the objects we intend to reason about, we now set up a theory to prove their properties. The main concepts are those of inductively and coinductively defined predicates. They can be declared to be either computationally relevant (c.r.) or else non-computational (n.c.).

3.1. **Formulas and their computational content.** Assume an infinite supply of predicate variables, each of its own *arity* (a list of types). We distinguish two sorts of predicate variables, "computationally relevant" ones X^c and "non-computational" ones X^{nc}, and use X for both.

Definition (Clauses and predicate forms)**.** *Clauses* K have the form

$$ \forall_{\vec{x}}(\tilde{Y}^c \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}^{\mathrm{nc}}_i \to \bar{X}_i))_{i<n} \to \bar{X}) $$

with all predicate variables Y^c_i, Z^{nc}_i, W^{nc}_i occurring exactly once and distinct from each other and from X. By $\bar{X}$ we denote the result of applying the predicate variable X to a list of terms of fitting types, and by $\tilde{X}$ lists of those. Iterated implications are understood as associated to the right. A premise of a clause is called a *parameter* premise if X does not occur in it, and a *recursive* premise otherwise. A clause K is *nullary* if it has no recursive premises. We call $I^c := \mu_{X^c}\vec{K}$ and $I^{\mathrm{nc}} := \mu_{X^{\mathrm{nc}}}\vec{K}$ with $\vec{K}$ not empty *predicate forms* (and use I for both), and similarly with ${}^{\mathrm{co}}I$ for I and ν for μ.

Definition (Algebra form of a predicate form)**.** From every clause K we obtain a constructor type by (i) omitting quantifiers, (ii) dropping all n.c. predicates and from the c.r. predicates their arguments, and (iii) replacing the remaining predicate variables by type variables. That is, from the clause

$$\forall_{\vec{x}}(\tilde{Y}^c \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}_i^{\mathrm{nc}} \to \bar{X}_i))_{i<n} \to \bar{X})$$

we obtain the constructor type $\vec{\alpha} \to (\xi)_{i<n} \to \xi$. With every predicate form $I^c := (\mu/\nu)_{X^c}\vec{K}$ we canonically associate the algebra form $\iota_{I^c} := \mu_\xi\vec{\kappa}$.

Definition (Predicates and formulas)**.**

$$\begin{aligned} &P, Q ::= X \mid \{\, \vec{x} \mid A \,\} \mid I(\vec{\rho}, \vec{P}\,) \mid {}^{\mathrm{co}}I(\vec{\rho}, \vec{P}\,) \qquad &&\text{(predicates)},\\ &A, B ::= P\vec{t} \mid A \to B \mid \forall_x A &&\text{(formulas)} \end{aligned}$$

with $I/{}^{\mathrm{co}}I$ a predicate form. $(I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)$ is the result of substituting the types $\vec{\rho}$ and the (already generated) predicates $\vec{P}$ for its type and predicate variables. To take care of the difference between X^c and X^{nc} we define the final predicate of a predicate or formula by

$$\begin{aligned} \mathrm{fp}(X) &:= X, & \mathrm{fp}(P\vec{t}\,) &:= \mathrm{fp}(P),\\ \mathrm{fp}(\{\, \vec{x} \mid A \,\}) &:= \mathrm{fp}(A), & \mathrm{fp}(A \to B) &:= \mathrm{fp}(B),\\ \mathrm{fp}((I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)) &:= I/{}^{\mathrm{co}}I, & \mathrm{fp}(\forall_x A) &:= \mathrm{fp}(A). \end{aligned}$$

We call a predicate or formula C *non-computational* (n.c., or *Harrop*) if its final predicate $\mathrm{fp}(C)$ is of the form X^{nc} or I^{nc}, else *computationally relevant* (c.r.). All predicate substitutions involved in $(I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)$ must substitute c.r. predicates for c.r. predicate variables and n.c. predicates for n.c. predicate variables. Such predicate substitutions are called *sharp*.

Predicates of the form $I(\vec{\rho}, \vec{P}\,)$ are called *inductive*, and predicates of the form ${}^{\mathrm{co}}I(\vec{\rho}, \vec{P}\,)$ *coinductive*.

The terms $\vec{t}$ are those introduced in Section 2.1, i.e., typed terms built from typed variables and constants by abstraction and application, and (importantly) those with a common reduct are identified.

A predicate of the form $\{\, \vec{x} \mid C \,\}$ is called a *comprehension term*. We identify $\{\, \vec{x} \mid C(\vec{x}\,) \,\}\vec{t}$ with $C(\vec{t}\,)$. For a predicate C of arity $(\rho, \vec{\sigma}\,)$ we write Ct for $\{\, \vec{y} \mid Ct\vec{y} \,\}$.

Definition (Type $\tau(C)$ and cotype $\varphi(C)$ of a c.r. predicate or formula C)**.** Assume a global injective assignment of type variables ξ to c.r. predicate variables X.

$$\begin{aligned} \tau(X) &:= \xi, \\ \tau(\{\, \vec{x} \mid A \,\}) &:= \tau(A), \\ \tau((I/{}^{\mathrm{co}}I)(\vec{\tau}, \vec{P}\,)) &:= \iota_I(\tau(\vec{P}^c)), \end{aligned} \qquad \begin{aligned} \tau(P\vec{t}\,) &:= \tau(P), \\ \tau(A \to B) &:= \begin{cases} \tau(A) \to \tau(B) & (A \text{ c.r.}) \\ \tau(B) & (A \text{ n.c.}) \end{cases} \\ \tau(\forall_x A) &:= \tau(A) \end{aligned}$$

where $\vec{P}^c$ are the c.r. predicates among $\vec{P}$ and ι_I is the algebra associated with the predicate $I/{}^{\mathrm{co}}I$. *Cotypes* are like types, but with algebra occurrences ι_I marked as ${}^{\mathrm{co}}\iota_I$ if they arise from a coinductive predicate: $\varphi({}^{\mathrm{co}}I(\vec{\tau}, \vec{P}\,)) := {}^{\mathrm{co}}\iota_I(\varphi(\vec{P}^c))$.

Examples. 1. The *even numbers* are inductively defined by

$$\mathrm{Even} := \mu_{X^c}(0 \in X^c, \forall_n(n \in X^c \to \mathcal{S}(\mathcal{S}n) \in X^c)).$$

The constructor types of $\tau(\mathrm{Even})$ are ξ and $\xi \to \xi$, hence $\tau(\mathrm{Even}) = \mathbb{N}$.

2. *Leibniz equality* $\equiv$ is inductively defined by $\mathrm{EqD} := \mu_{X^{\mathrm{nc}}}(\forall_x X^{\mathrm{nc}} xx)$.

3. The missing *logical connectives* $\vee$, $\wedge$, $\exists$ are nullary inductive predicates with parameters. For instance, disjunction is a special case of union

$$\mathrm{Cup}_{Y,Z} := \mu_{X^c}(\forall_{\vec{x}}(Y\vec{x} \to X^c\vec{x}\,),\ \forall_{\vec{x}}(Z\vec{x} \to X^c\vec{x}\,)).$$

Since Y, Z can be chosen as either c.r. or n.c. we obtain the variants

$$\begin{aligned} \mathrm{CupD}_{Y^c,Z^c} &:= \mu_{X^c}(\forall_{\vec{x}}(Y^c\vec{x} \to X^c\vec{x}\,),\ \forall_{\vec{x}}(Z^c\vec{x} \to X^c\vec{x}\,)), \\ \mathrm{CupL}_{Y^c,Z^{\mathrm{nc}}} &:= \mu_{X^c}(\forall_{\vec{x}}(Y^c\vec{x} \to X^c\vec{x}\,),\ \forall_{\vec{x}}(Z^{\mathrm{nc}}\vec{x} \to X^c\vec{x}\,)), \\ \mathrm{CupR}_{Y^{\mathrm{nc}},Z^c} &:= \mu_{X^c}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to X^c\vec{x}\,),\ \forall_{\vec{x}}(Z^c\vec{x} \to X^c\vec{x}\,)), \\ \mathrm{CupU}_{Y^{\mathrm{nc}},Z^{\mathrm{nc}}} &:= \mu_{X^c}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to X^c\vec{x}\,),\ \forall_{\vec{x}}(Z^{\mathrm{nc}}\vec{x} \to X^c\vec{x}\,)), \\ \mathrm{CupNc}_{Y,Z} &:= \mu_{X^{\mathrm{nc}}}(\forall_{\vec{x}}(Y\vec{x} \to X^{\mathrm{nc}}\vec{x}\,),\ \forall_{\vec{x}}(Z\vec{x} \to X^{\mathrm{nc}}\vec{x}\,)). \end{aligned}$$

(D, L, R, U for "double", "left", "right" and "uniform"). Then by definition

$$\begin{aligned} \tau(\mathrm{CupD}) &= \mu_\xi(\beta_0 \to \xi, \beta_1 \to \xi) &&= \beta_0 + \beta_1, \\ \tau(\mathrm{CupL}) &= \mu_\xi(\beta \to \xi, \xi) &&= \beta + \mathbb{U}, \\ \tau(\mathrm{CupR}) &= \mu_\xi(\xi, \beta \to \xi) &&= \mathbb{U} + \beta, \\ \tau(\mathrm{CupU}) &= \mu_\xi(\xi, \xi) &&= \mathbb{B}. \end{aligned}$$

In case of nullary predicates we write $A \vee^{\mathrm{d}} B$ for $\mathrm{CupD}_{\{|A\},\{|B\}}$, and similarly for $\vee^{\mathrm{l}}$, $\vee^{\mathrm{r}}$, $\vee^{\mathrm{u}}$, $\vee^{\mathrm{nc}}$. Since the "decoration" is determined by the c.r./n.c. status of the two parameter predicates we can leave it out in $\vee^{\mathrm{d}}, \vee^{\mathrm{l}}, \vee^{\mathrm{r}}, \vee^{\mathrm{u}}$ and write $\vee$. However in the final nc-variant we suppress even the information which clause has been used, and hence must keep the notation $\vee^{\mathrm{nc}}$.

3.2. **Axioms of** TCF. The essential axioms of TCF are introduction and elimination axioms for (co)inductively defined predicates. To grasp the general form of these axioms it is convenient to write a clause

$$\forall_{\vec{x}}(\tilde{Y}^c \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}_i^{\mathrm{nc}} \to \bar{X}_i))_{i<n} \to \bar{X}) \quad \text{as} \quad \forall_{\vec{x}}((A_\nu(X))_{\nu<n} \to X\vec{t}\,).$$

Definition. For an inductive predicate $\mu_X(\forall_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i} \to X\vec{t}_i))_{i<k} =: I$ we have k introduction axioms I_i^+ $(i < k)$ and one elimination axiom I^-:

$$I_i^+ : \forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to I\vec{t}_i), \tag{2}$$

$$I^- \colon (\forall_{\vec{x}_i}((A_{i\nu}(I \cap X))_{\nu<n_i} \to X\vec{t_i}))_{i<k} \to I \subseteq X \tag{3}$$

(3) expresses that every competitor X satisfying the same clauses contains I. We take all substitution instances of I_i^+, I^- (w.r.t. substitutions for type and predicate variables) as axioms.

In (3) a "strengthened" form of the "step formula" has been used, namely $\forall_{\vec{x}_i}(A_{i\nu}(I \cap X))_{\nu<n_i} \to X\vec{t_i}$ rather than $\forall_{\vec{x}_i}(A_{i\nu}(X))_{\nu<n_i} \to X\vec{t_i}$. In applications this simplifies the proof of the "step", since we have an additional I-hypothesis available.

To understand the axioms for coinductive predicates note that the conjunction of the k clauses (2) of an inductive predicate I is equivalent to

$$\forall_{\vec{x}}(\bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I) \land \vec{x} \equiv \vec{t_i}) \to I\vec{x}\,).$$

Definition. For an inductive predicate $\mu_X(\forall_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i} \to X\vec{t_i}))_{i<k} =: I$ we define its *closure axiom* ${}^{\mathrm{co}}I^-$ and its *greatest-fixed-point axiom* ${}^{\mathrm{co}}I^+$:

$${}^{\mathrm{co}}I^- \colon \forall_{\vec{x}}({}^{\mathrm{co}}I\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I) \land \vec{x} \equiv \vec{t_i})) \tag{4}$$

$${}^{\mathrm{co}}I^+ \colon \forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I \cup X) \land \vec{x} \equiv \vec{t_i})) \to X \subseteq {}^{\mathrm{co}}I. \tag{5}$$

(5) expresses that every competitor X satisfying the closure axiom is contained in ${}^{\mathrm{co}}I$. We take all substitution instances of ${}^{\mathrm{co}}I^-$, ${}^{\mathrm{co}}I^+$ (w.r.t. type and predicate variables) as axioms.

Here again we have used a strengthened form of the step formula, with $A_{i\nu}({}^{\mathrm{co}}I \cup X)$ rather than $A_{i\nu}(X)$. In applications of (5) this simplifies the proof of the step, since its conclusion is weaker.

Example. The conjunction of the two clauses of Even is equivalent to

$$\forall_n(n \equiv 0 \lor \exists_{n'}(n' \in \mathrm{Even} \land n \equiv \mathcal{S}(\mathcal{S}n')) \to n \in \mathrm{Even}).$$

Hence the closure and greatest-fixed-point axioms for its dual ${}^{\mathrm{co}}\mathrm{Even}$ are

$$\forall_n(n \in {}^{\mathrm{co}}\mathrm{Even} \to n \equiv 0 \lor \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \land n \equiv \mathcal{S}(\mathcal{S}n'))),$$
$$\forall_n(Xn \to n \equiv 0 \lor \exists_{n'}(n' \in ({}^{\mathrm{co}}\mathrm{Even} \cup X) \land n \equiv \mathcal{S}(\mathcal{S}n'))) \to X \subseteq {}^{\mathrm{co}}\mathrm{Even}.$$

For n.c. inductive or coinductive predicates the axioms are formed as in the c.r. case, using $\lor^{\mathrm{nc}}$ for the closure axiom of ${}^{\mathrm{co}}I^{\mathrm{nc}}$. But there is an important restriction: for I^{nc} with more than one clause the elimination axiom $(I^{\mathrm{nc}})^-$ can only be used with a *non-computational* competitor predicate. This is needed in the proof of the soundness theorem below. However, this restriction does not apply to I^{nc} defined by one clause only. Important examples of such *one-clause-nc* inductive predicates are Leibniz equality and the non-computational variants of the existential quantifier and of conjunction.

Lemma 3.1. *$I \subseteq {}^{\mathrm{co}}I$, $I^{\mathrm{nc}} \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$ and also $I \subseteq I^{\mathrm{nc}}$, ${}^{\mathrm{co}}I \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$.*

From the definition of Leibniz equality we can deduce the property Leibniz used as a definition.

Lemma 3.2 (Compatibility of EqD). *$\forall_{x,y}(x \equiv y \to A(x) \to A(y))$.*

Proof. By the elimination axiom with $X := \{\, x, y \mid A(x) \to A(y) \,\}$. □

Using compatibility of $\equiv$ one easily proves symmetry and transitivity. Define *falsity* by $\mathbf{F} := (\mathtt{ff} \equiv \mathtt{tt})$. Then we can prove "Ex-falso-quodlibet":

Theorem 3.3. *For every formula A we can derive $\mathbf{F} \to A$ from assumptions $\mathrm{Ef}_Y\colon \forall_{\vec{x}}(\mathbf{F} \to Y\vec{x}\,)$ for predicate variables Y strictly positive in A, and $\mathrm{Ef}_I\colon \forall_{\vec{x}}(\mathbf{F} \to I\vec{x}\,)$ for inductive predicates I without a nullary clause.*

Proof. We first show $\mathrm{Ef}_{\mathrm{EqD}}\colon \mathbf{F} \to x^\rho \equiv y^\rho$. By the introduction axiom we have $\mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy \equiv \mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy$. Then from $\mathtt{ff} \equiv \mathtt{tt}$ we get $\mathcal{R}^\rho_{\mathbb{B}}\mathtt{tt}xy \equiv \mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy$ by compatibility. Now $\mathcal{R}^\rho_{\mathbb{B}}\mathtt{tt}xy$ converts to x and $\mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy$ converts to y. Hence $x^\rho \equiv y^\rho$, since we identify terms with a common reduct.

The claim can now be proved by induction on A. *Case $I\vec{s}$.* If I has no nullary clause take Ef_I. Otherwise 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 $s_i \equiv t_i$, by the remark above. Hence $I\vec{s}$ by compatibility. *Case ${}^{\mathrm{co}}I\vec{s}$.* Use Lemma 3.1. The cases $Y\vec{s}$, $A \to B$ and $\forall_x A$ are obvious. $\square$

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

3.3. **Equality and extensionality.**

We first consider closed base types and take the algebra $\mathbb{Y}$ of binary trees as an example. Totality $T_{\mathbb{Y}}$ is inductively defined by the axioms

$$(T_{\mathbb{Y}})_0^+\colon - \in T_{\mathbb{Y}}, \qquad (T_{\mathbb{Y}})_1^+\colon \forall_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \to \mathcal{C}t_1t_2 \in T_{\mathbb{Y}}),$$
$$T_{\mathbb{Y}}^-\colon - \in X \to \forall_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \cap X \to \mathcal{C}t_1t_2 \in X) \to T_{\mathbb{Y}} \subseteq X.$$

and cototality ${}^{\mathrm{co}}T_{\mathbb{Y}}$ coinductively by

$${}^{\mathrm{co}}T_{\mathbb{Y}}^-\colon \forall_t(t \in {}^{\mathrm{co}}T_{\mathbb{Y}} \to (t{\equiv}-) \lor \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \land t{\equiv}\mathcal{C}t_1t_2))$$
$${}^{\mathrm{co}}T_{\mathbb{Y}}^+\colon \forall_t(t \in X \to (t{\equiv}-) \lor \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \cup X \land t{\equiv}\mathcal{C}t_1t_2)) \to X \subseteq {}^{\mathrm{co}}T_{\mathbb{Y}}.$$

As candidates for equality we define binary versions of $T_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$, called similarity $\sim_{\mathbb{Y}}$ and bisimilarity $\approx_{\mathbb{Y}}$, for instance by

$$- \sim_{\mathbb{Y}} -, \qquad \forall_{t_1,t_1'}(t_1 \sim_{\mathbb{Y}} t_1' \to \forall_{t_2,t_2'}(t_2 \sim_{\mathbb{Y}} t_2' \to \mathcal{C}t_1t_2 \sim_{\mathbb{Y}} \mathcal{C}t_1't_2')).$$

We aim at using $\sim_{\mathbb{Y}}$ and $\approx_{\mathbb{Y}}$ for a characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$. This is useful because it gives us a tool (induction, coinduction) to prove equalities $t \equiv t'$, which otherwise would be difficult. We will need another axiom, the Bisimilarity Axiom, which is justified by the fact that it holds in our intended model (cf. Lemma 1.5).

Axiom (Bisimilarity)**.** $\forall_{t,t'}(t \approx_{\mathbb{Y}} t' \to t \equiv t')$.

Lemma 3.4 (Characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$)**.**

(a) $\forall_{t,t'}(t \sim_{\mathbb{Y}} t' \leftrightarrow t, t' \in T_{\mathbb{Y}} \wedge t \equiv t')$.
(b) $\forall_{t,t'}(t \approx_{\mathbb{Y}} t' \leftrightarrow t, t' \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge t \equiv t')$.

Proof. (b). The proof of Lemma 1.6 has been given in enough detail to make its formalization immediate. We need ${}^{\mathrm{co}}T^{\pm}_{\mathbb{Y}}$ and $\approx^{\pm}_{\mathbb{Y}}$.

(a). Similar to (b), using $T^{\pm}_{\mathbb{Y}}$, $\sim^{\pm}_{\mathbb{Y}}$ instead. For the proof of $t \sim_{\mathbb{Y}} t' \to t \equiv t'$ use (b) and $\sim_{\mathbb{Y}} \subseteq \approx_{\mathbb{Y}}$. $\square$

Hence $\sim_{\mathbb{Y}}$ is a partial equivalence relation on $\mathfrak{C}_{\mathbb{Y}}$ with domain $T_{\mathbb{Y}}$, and similar for $\approx_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$.

At higher types we use *pointwise equality* of Gandy (1953, 1956) and Takeuti (1953). This notion is somewhat delicate in our setting, since we allow infinite base type objects.

Definition. For every algebra form ι with type parameters $\vec{\alpha}$ we define two predicate forms $\sim_\iota$, $\approx_\iota$ (called *relative similarity* and *relative bisimilarity*) with type parameters $\vec{\alpha}$ and predicate parameters $\vec{Y}$ (where Y_i has arity (α_i, α_i)) as follows. Let $\vec{\alpha} \to (\xi)_{i<n} \to \xi$ be a constructor type. Take $(\mu/\nu)_Z(\vec{K})$, where the clause for the constructor type above is

$$Y_1 u_1 u_1' \to \cdots \to Y_n u_n u_n' \to Z v_1 v_1' \to \cdots \to Z v_m v_m' \to Z(\mathcal{C}\vec{u}\vec{v}, \mathcal{C}\vec{u}'\vec{v}\,)$$

with $\mathcal{C}$ the corresponding constructor of ι. (Absolute) similarity / bisimilarity predicates arise from the relative ones by substituting a similarity / bisimilarity predicate for Y.

Definition (Cotype of a c.r. predicate or formula C)**.** *Cotypes* $\varphi(C)$ are like types $\tau(C)$, but with algebra occurrences ι_I marked as ${}^{\mathrm{co}}\iota_I$ if they arise from a coinductive predicate: $\varphi({}^{\mathrm{co}}I(\vec{\tau}, \vec{P}\,)) := {}^{\mathrm{co}}\iota_I(\varphi(\vec{P}^c))$.

Definition (Pointwise equality $\doteq_\varphi$ w.r.t. a cotype φ)**.**

$$\begin{aligned}
(x \doteq_\alpha y) &:= Yxy \quad \text{with } Y \text{ uniquely assigned to } \alpha,\\
(x \doteq_{\iota(\vec{\varphi})} y) &:= (x \sim y) \quad \text{with } {\sim} := {\sim_\iota}(\doteq_{\vec{\varphi}}),\\
(x \doteq_{{}^{\mathrm{co}}\iota(\vec{\varphi})} y) &:= (x \approx y) \quad \text{with } {\approx} := {\approx_\iota}(\doteq_{\vec{\varphi}}),\\
(f \doteq_{\varphi \to \psi} g) &:= \forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi gy).
\end{aligned}$$

Extensionality Ext_φ w.r.t. a cotype φ arises as a special case

$$(x \in \mathrm{Ext}_\varphi) := (x \doteq_\varphi x).$$

Of course extensionality is a desirable property, but in our model it does not hold generally. Here is an example of a functional F which is non-extensional w.r.t. $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$. Define f, g of type $\mathbb{N} \to \mathbb{N}$ by the computation rules $fn = 0$ and $g0 = 0$, $g(\mathcal{S}n) = gn$. Then $f\bot_{\mathbb{N}} = 0$ because of the computation rules for f. For $g\bot_{\mathbb{N}}$ no computation rule fits, but by the inductive definition of $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$ (see Section 2) $[\![g\bot_{\mathbb{N}}]\!]$ is the empty object $[\![\bot_{\mathbb{N}}]\!]$. Hence $f \doteq g$, i.e., $\forall_{n,m}(n \doteq_{\mathbb{N}} m \to fn \doteq_{\mathbb{N}} gm)$, since $n \doteq_{\mathbb{N}} m$ implies $n \in T_{\mathbb{N}}$ and $n \equiv m$. Therefore F defined by $Fh = h\bot_{\mathbb{N}}$ maps the pointwise equal f, g to different values.

By Lemma 3.4 we know the equivalence of $\mathrm{Ext}_{\mathbb{Y}}$ and $T_{\mathbb{Y}}$ (and of $\mathrm{Ext}_{({}^{\mathrm{co}}\mathbb{Y})}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$); this also holds for arbitrary closed base cotypes. This equivalence can be extended to closed cotypes of level 1:

Lemma 3.5. *The predicates* Ext_φ *and* T_φ *are equivalent for closed cotypes of level* ≤ 1.

Proof. For closed base cotypes this has been proved in Lemma 3.4 (for the special case of the algebra $\mathbb{Y}$). In case of level 1 we use induction on the height of the cotype. Let $\varphi \to \psi$ be a closed cotype of level 1. The following are equivalent.

$$\begin{aligned}
&f \in \mathrm{Ext}_{\varphi\to\psi}\\
&f \doteq_{\varphi\to\psi} f\\
&\forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi fy)\\
&\forall_{x\in T_\varphi}(fx \doteq_\psi fx) \qquad\qquad \text{by Lemma 3.4, since } \mathrm{lv}(\varphi)=0\\
&\forall_{x\in T_\varphi}(fx \in \mathrm{Ext}_\psi).
\end{aligned}$$

By induction hypothesis the final formula is equivalent to $f \in T_{\varphi\to\psi}$. □

Lemma 3.6. *For every closed cotype* φ *the relation* $\doteq_\varphi$ *is a partial equivalence relation with domain* Ext_φ.

Proof. By induction on the height $|\varphi|$ of φ. *Case* $\iota(\vec{\varphi})/{}^{\mathrm{co}}\iota(\vec{\varphi})$. For $\sim_{\mathbb{Y}}$ and $\approx_{\mathbb{Y}}$ this was proved in Lemma 3.4. In the general case use the induction hypothesis and the inductive / coinductive definition of $\sim_\iota$ / $\approx_\iota$.

Case $\varphi \to \psi$. We first prove symmetry of $\doteq_{\varphi\to\psi}$. Let $f \doteq_{\varphi\to\psi} g$. The goal is $g \doteq_{\varphi\to\psi} f$. Assume $x \doteq_\varphi y$. The goal now is $gx \doteq_\psi fy$. From $x \doteq_\varphi y$ we obtain $y \doteq_\varphi x$ by symmetry of $\doteq_\varphi$, hence $fy \doteq_\psi gx$ from $f \doteq_{\varphi\to\psi} g$, hence $gx \doteq_\psi fy$ by symmetry of $\doteq_\psi$.

We finally prove transitivity of $\doteq_{\varphi\to\psi}$. Let $f \doteq_{\varphi\to\psi} g$ and $g \doteq_{\varphi\to\psi} h$. The goal is $f \doteq_{\varphi\to\psi} h$. Assume $x \doteq_\varphi y$. The goal now is $fx \doteq_\psi hy$. From $x \doteq_\varphi y$ we obtain $y \doteq_\varphi x$ by symmetry of $\doteq_\varphi$, hence $x \doteq_\varphi x$ by transitivity of $\doteq_\varphi$. Then $fx \doteq_\psi gx$ follows from $f \doteq_{\varphi\to\psi} g$. We also have $gx \doteq_\psi hy$ from $g \doteq_{\varphi\to\psi} h$. Using transitivity of $\doteq_\psi$ we obtain $fx \doteq_\psi hy$. □

4. Computational content of proofs

We define what it means for a term t to "realize" a c.r. formula A. From a proof M of A we extract a term $\mathrm{et}(M)$ and (formally) prove that it is a realizer of A. In this proof we need "invariance axioms" stating that every c.r. formula not involving realizability is invariant under realizability.

4.1. **Realizability.**

Assume that we have a global assignment giving for every c.r. predicate variable X of arity $\vec{\rho}$ an n.c. predicate variable $X^{\mathbf{r}}$ of arity $(\vec{\rho}, \xi)$ where ξ is the type variable associated with X. We will also introduce $I^{\mathbf{r}}/{}^{\mathrm{co}}I^{\mathbf{r}}$ for (co)inductive predicates $I/{}^{\mathrm{co}}I$. A formula or predicate C is called $\mathbf{r}$-free if it does not contain any of these $X^{\mathbf{r}}$, $I^{\mathbf{r}}$ or ${}^{\mathrm{co}}I^{\mathbf{r}}$. A derivation M is called $\mathbf{r}$-free if it contains $\mathbf{r}$-free formulas only.

Definition ($C^{\mathbf{r}}$ for $\mathbf{r}$-free predicates and formulas C)**.** For every $\mathbf{r}$-free predicate or formula C we define a predicate or formula $C^{\mathbf{r}}$. For n.c. C let $C^{\mathbf{r}} := C$. In case C is c.r. $C^{\mathbf{r}}$ is an n.c. predicate of arity $(\vec{\sigma}, \tau(C))$ with $\vec{\sigma}$ the arity of C. We often write $z\ \mathbf{r}\ C$ for $C^{\mathbf{r}}z$ in case C is a c.r. formula. For c.r. *predicates* X let $X^{\mathbf{r}}$ be the n.c. predicate variable provided, and

$$\{\,\vec{x} \mid A\,\}^{\mathbf{r}} := \{\,\vec{x}, z \mid z\ \mathbf{r}\ A\,\}.$$

Now consider a c.r. (co)inductive predicate

$$I/{}^{\mathrm{co}}I := (\mu/\nu)_X((K_i(X))_{i<k}$$

with associated algebra form $\iota_I = \mu_\xi(\kappa_i(\xi))_{i<k}$ where $\kappa_i(\xi) := \tau(K_i(X))$. The i-th constructor of ι_I is $\mathcal{C}_i \colon \kappa_i(\iota_I)$. Let s be a variable of type $\tau(I)$ and ϑ the substitution $\xi \mapsto \tau(I)$, $X^{\mathbf{r}} \mapsto \{\, \vec{x}, s \mid Y\vec{x}s \,\}$. We define n.c. predicates $I^{\mathbf{r}}$ and ${}^{\mathrm{co}}I^{\mathbf{r}}$ by

$$I^{\mathbf{r}}/{}^{\mathrm{co}}I^{\mathbf{r}} := (\mu/\nu)_Y((C_i \;\mathbf{r}\; K_i(X))\vartheta)_{i<k}.$$

The substitution ϑ is necessary since the arity of Y (and hence of $I^{\mathbf{r}}/{}^{\mathrm{co}}I^{\mathbf{r}}$) must be $(\vec{\rho}, \tau(I))$ and not $(\vec{\rho}, \xi)$. For c.r. *formulas* let

$$z \;\mathbf{r}\; P\vec{t} := P^{\mathbf{r}}\vec{t}z,$$

$$z \;\mathbf{r}\; (A \to B) := \begin{cases} \forall_w(w \;\mathbf{r}\; A \to zw \;\mathbf{r}\; B) & \text{if } A \text{ is c.r.} \\ A \to z \;\mathbf{r}\; B & \text{if } A \text{ is n.c.} \end{cases}$$

$$z \;\mathbf{r}\; \forall_x A := \forall_x(z \;\mathbf{r}\; A).$$

As an example for the construction of $I^{\mathbf{r}}$ consider the predicate Even, defined by $\mu_X(K_0(X), K_1(X))$ with $K_0(X) := (0 \in X)$ and $K_1(X) := \forall_n(n \in X \to \mathcal{S}(\mathcal{S}n) \in X)$. The associated algebra form is $\mu_\xi(\kappa_0(\xi), \kappa_1(\xi))$ with $\kappa_0(\xi) := \xi$ and $\kappa_1(\xi) := \xi \to \xi$, i.e., the algebra $\mathbb{N}$ with constructors $\mathcal{C}_0 := 0$ and $\mathcal{C}_1 := \mathcal{S}$. Let ϑ be the substitution $\xi \mapsto \mathbb{N}$, $X^{\mathbf{r}} \mapsto \{\, n, m \mid Ynm \,\}$. Since $\mathcal{S} \;\mathbf{r}\; K_1(X)$ is $\forall_{n,m}(X^{\mathbf{r}}nm \to X^{\mathbf{r}}(\mathcal{S}(\mathcal{S}n), \mathcal{S}m))$ we obtain

$$I^{\mathbf{r}} := \mu_Y(Y00, \forall_{n,m}(Ynm \to Y(\mathcal{S}(\mathcal{S}n), \mathcal{S}m)).$$

Lemma 4.1. *For closed base types ι the following are equivalent.*

(a) $T^{\mathbf{r}}_\iota xy$,
(b) $x \sim^{\mathrm{nc}}_\iota y$,
(c) $x \in T^{\mathrm{nc}}_\iota \wedge x \equiv y$.

Proof. (a) $\leftrightarrow$ (b). Both $T^{\mathbf{r}}_\iota xy$ and $x \sim^{\mathrm{nc}}_\iota y$ satisfy the same clauses. Use the respective elimination axiom in each of the two directions.

(b) $\leftrightarrow$ (c). Use Lemma 3.4. □

Lemma 4.2. *For closed base types ι the following are equivalent.*

(a) ${}^{\mathrm{co}}T^{\mathbf{r}}_\iota xy$,
(b) $x \approx^{\mathrm{nc}}_\iota y$,
(c) $x \in {}^{\mathrm{co}}T^{\mathrm{nc}}_\iota \wedge x \equiv y$.

Proof. As an example we give the proof for $\mathbb{N}$. Since we have n.c. goals only, decorations are omitted. For (a) $\to$ (b) apply $\approx^-_{\mathbb{N}}$ with ${}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}$ for X.

$$\begin{aligned} \approx^-_{\mathbb{N}} \colon \forall_{n,m}(Xnm \to (n \equiv 0 \wedge m \equiv 0) \vee \\ \exists_{n',m'}((n' \approx_{\mathbb{N}} m' \vee Xn'm') \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m')) \to X \subseteq \approx_{\mathbb{N}}. \end{aligned}$$

It suffices to prove the premise. Assume ${}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}nm$; the goal is

$$C := {}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}00 \vee \exists_{n',m'}((n' \approx_{\mathbb{N}} m' \vee {}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}n'm') \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m').$$

By the closure axiom $({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}})^-$ we have

$$(n \equiv 0 \wedge m \equiv 0) \vee \exists_{n',m'}({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}n'm' \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m').$$

We argue by cases (i.e., use $\vee^-$).

Case 1. $n \equiv 0 \wedge m \equiv 0$. Go for the l.h.s. of the disjunction C and show ${}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}00$. But this follows from the greatest-fixed-point axiom for ${}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}$ with competitor predicate $\{\, n, m \mid n \equiv 0 \wedge m \equiv 0 \,\}$.

Case 2. $\exists_{n',m'}({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}n'm' \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m')$. Go for the r.h.s. of C.

(b) $\to$ (a). Recall ${}^{\mathrm{co}}T_{\mathbb{N}} := \nu_X(0 \in X,\ \forall_{n \in X}(\mathcal{S}n \in X))$, hence by definition

$$ {}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}} := \nu_{X^{\mathbf{r}}}(X^{\mathbf{r}}00,\ \forall_{n,m}(X^{\mathbf{r}}nm \to X^{\mathbf{r}}(\mathcal{S}n)(\mathcal{S}m))). $$

To show $m \approx_{\mathbb{N}} n \to {}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}mn$, apply $({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}})^+$ with $\approx_{\mathbb{N}}$ for X; recall $({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}})^+$:

$$ \forall_{n,m}(Xnm \to X00 \vee \exists_{n',m'}(n', m' \in ({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}} \cup X) \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m')) \to $$
$$ X \subseteq {}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}}. $$

It suffices to prove the premise. Assume $n \approx_{\mathbb{N}} m$; the goal is

$$ C := (0 \approx_{\mathbb{N}} 0) \vee \exists_{n',m'}((n', m' \in ({}^{\mathrm{co}}T^{\mathbf{r}}_{\mathbb{N}} \cup \approx_{\mathbb{N}}) \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m')). $$

By the closure axiom $(\approx_{\mathbb{N}})^-$ we have

$$ n \approx_{\mathbb{N}} m \to (n \equiv 0 \wedge m \equiv 0) \vee \exists_{n',m'}(n' \approx_{\mathbb{N}} m' \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m'). $$

We argue by cases (i.e., use $\vee^-$).

Case 1. $n \equiv 0 \wedge m \equiv 0$. Go for the l.h.s. of the disjunction C and show $0 \approx_{\mathbb{N}} 0$. But this follows from $n \approx_{\mathbb{N}} m$.

Case 2. $\exists_{n',m'}(n' \approx_{\mathbb{N}} m' \wedge n \equiv \mathcal{S}n' \wedge m \equiv \mathcal{S}m')$. Go for the r.h.s. of C.

(b) $\leftrightarrow$ (c). Use the Bisimilarity Axiom and Lemma 3.4. □

Lemma 4.3 (Realizers for $\exists$). *$z \mathbf{r} \exists_x A \leftrightarrow \exists_x(z \mathbf{r} A)$ for A c.r.*

Proof. Recall $\mathrm{Ex}_Y := \mu_X(\forall_x(x \in Y \to X))$. Then

$$ \mathrm{Ex}^{\mathbf{r}}_{Y^{\mathbf{r}}} := \mu_{X^{\mathbf{r}}}(\forall_{x,z}(Y^{\mathbf{r}}xz \to X^{\mathbf{r}}z)). $$

Now substituting $Y^{\mathbf{r}}$ by $\{\, x, z \mid z \mathbf{r} A \,\}$ in the introduction axiom gives

$$ (\mathrm{Ex}^{\mathbf{r}}_{\{x,z|z\mathbf{r}A\}})^+_0 \colon \forall_{x,z}(z \mathbf{r} A \to z \mathbf{r} \exists_x A) $$

Conversely, the elimination axiom $(\mathrm{Ex}^{\mathbf{r}}_{Y^{\mathbf{r}}})^-$ is

$$ \forall_z(z \in \mathrm{Ex}^{\mathbf{r}}_{Y^{\mathbf{r}}} \to \forall_{x,z}(Y^{\mathbf{r}}xz \to z \in X) \to z \in X). $$

which is equivalent to

$$ \forall_z(z \in \mathrm{Ex}^{\mathbf{r}}_{Y^{\mathbf{r}}} \to \forall_z(\exists_x Y^{\mathbf{r}}xz \to z \in X) \to z \in X). $$

Substituting X by $\{\, z \mid \exists_x(Y^{\mathbf{r}}xz) \,\}$ makes the middle part provable. Thus with $\{\, x, z \mid z \mathbf{r} A \,\}$ for $Y^{\mathbf{r}}$ we obtain $\forall_z(z \mathbf{r} \exists_x A \to \exists_x(z \mathbf{r} A))$ from $(\mathrm{Ex}^{\mathbf{r}}_{\{x,z|z\mathbf{r}A\}})^-$. □

Lemma 4.4 (Realizers for $\wedge$). *$z \mathbf{r} (A \wedge B)$ is equivalent to*

$$ \begin{array}{ll} z \equiv \langle \mathrm{lft}(z), \mathrm{rht}(z) \rangle \wedge (\mathrm{lft}(z) \mathbf{r} A) \wedge (\mathrm{rht}(z) \mathbf{r} B) & \textit{for } A \textit{ c.r. and } B \textit{ c.r.} \\ (z \mathbf{r} A) \wedge B & \textit{for } A \textit{ c.r. and } B \textit{ n.c.} \\ A \wedge (z \mathbf{r} B) & \textit{for } A \textit{ n.c. and } B \textit{ c.r.} \end{array} $$

Proof. Case A, B c.r. Recall $\mathrm{AndD}_{X^c,Y^c} := \mu_{Z^c}(X^c \to Y^c \to Z^c)$. Then

$$ \mathrm{AndD}^{\mathbf{r}}_{X^{\mathbf{r}},Y^{\mathbf{r}}} := \mu_{Z^{\mathbf{r}}}(\forall_x(x \in X^{\mathbf{r}} \to \forall_y(y \in Y^{\mathbf{r}} \to \langle x, y \rangle \in Z^{\mathbf{r}}))). $$

Substituting $X^{\mathbf{r}}$ by $\{\, x \mid x \mathbf{r} A \,\}$ and $Y^{\mathbf{r}}$ by $\{\, y \mid y \mathbf{r} B \,\}$ gives

$$ (\mathrm{AndD}^{\mathbf{r}}_{\{x|x\mathbf{r}A\},\{y|y\mathbf{r}B\}})^+_0 \colon \forall_x((x \mathbf{r} A) \to \forall_y(y \mathbf{r} B \to \langle x, y \rangle \mathbf{r} (A \wedge B))). $$

This suffices for "$\leftarrow$". Conversely, the elimination axiom $(\mathrm{AndD}^{\mathbf{r}}_{X^{\mathbf{r}},Y^{\mathbf{r}}})^-$ is

$$\forall_x(x \in X^{\mathbf{r}} \to \forall_y(y \in Y^{\mathbf{r}} \to \langle x,y\rangle \in Z)) \to \mathrm{AndD}^{\mathbf{r}}_{X^{\mathbf{r}},Y^{\mathbf{r}}} \subseteq Z.$$

Substitute Z by $\{\, z \mid z \equiv \langle \mathrm{lft}(z), \mathrm{rht}(z)\rangle \wedge (\mathrm{lft}(z)\ \mathbf{r}\ A) \wedge (\mathrm{rht}(z)\ \mathbf{r}\ B)\,\}$. With $\{\, x \mid x\ \mathbf{r}\ A\,\}$ for $X^{\mathbf{r}}$ and $\{\, y \mid y\ \mathbf{r}\ B\,\}$ for $Y^{\mathbf{r}}$ the premise is provable. Hence

$$\forall_z(z\ \mathbf{r}\ (A \wedge B) \to z \equiv \langle \mathrm{lft}(z), \mathrm{rht}(z)\rangle \wedge (\mathrm{lft}(z)\ \mathbf{r}\ A) \wedge (\mathrm{rht}(z)\ \mathbf{r}\ B)).$$

Case A c.r., B n.c. Recall $\mathrm{AndL}_{X^c,Y^{\mathrm{nc}}} := \mu_{Z^c}(X^c \to Y^{\mathrm{nc}} \to Z^c)$. Then

$$\mathrm{AndL}^{\mathbf{r}}_{X^{\mathbf{r}},Y^{\mathrm{nc}}} := \mu_{Z^{\mathbf{r}}}(\forall_z(z\ \mathbf{r}\ X \to Y^{\mathrm{nc}} \to z \in Z^{\mathbf{r}})).$$

Substituting $X^{\mathbf{r}}$ by $\{\, z \mid z\ \mathbf{r}\ A\,\}$ and Y^{nc} by B gives

$$(\mathrm{AndL}^{\mathbf{r}}_{\{z|z\mathbf{r}A\},B})^+_0 \colon \forall_z((z\ \mathbf{r}\ A) \to B \to z\ \mathbf{r}\ (A \wedge B)).$$

This suffices for "$\leftarrow$". Conversely, the elimination axiom $(\mathrm{AndL}^{\mathbf{r}}_{X,Y^{\mathrm{nc}}})^-$ is

$$\forall_z(z\ \mathbf{r}\ X \to Y^{\mathrm{nc}} \to z \in Z) \to \mathrm{AndL}^{\mathbf{r}}_{X,Y^{\mathrm{nc}}} \subseteq Z.$$

Substitute Z by $\{\, z \mid (z\ \mathbf{r}\ A) \wedge B\,\}$. Then with $\{\, z \mid z\ \mathbf{r}\ A\,\}$ for X and B for Y^{nc} the premise is provable and we obtain

$$\forall_z(z\ \mathbf{r}\ (A \wedge B) \to (z\ \mathbf{r}\ A) \wedge B). \qquad \square$$

Recall that for the sum type $\rho+\sigma$ we had the constructors $(\mathrm{InL}_{\rho\sigma})^{\rho\to\rho+\sigma}$ and $(\mathrm{InR}_{\rho\sigma})^{\sigma\to\rho+\sigma}$. In the special situation that one of the two parameter types is the unit type $\mathbb{U}$ it is common to view the sum type $\mathbb{U}+\sigma$ as a unary algebra form, with constructors DummyL of type $\mathbb{U}+\sigma$ and Inr of type $\sigma \to \mathbb{U}+\sigma$. Similarly $\rho+\mathbb{U}$ is viewed as a unary algebra, with constructors Inl of type $\rho \to \rho+\mathbb{U}$ and DummyR of type $\rho+\mathbb{U}$.

Lemma 4.5 (Realizers for $\vee$). *$z\ \mathbf{r}\ (A \vee B)$ is equivalent to*

$\exists_x(x\ \mathbf{r}\ A \wedge z \equiv \mathrm{InL}(x)) \vee^{\mathrm{nc}} \exists_y(y\ \mathbf{r}\ B \wedge z \equiv \mathrm{InR}(y))$	*for A, B c.r.*
$\exists_x(x\ \mathbf{r}\ A \wedge z \equiv \mathrm{Inl}(x)) \vee^{\mathrm{nc}} (B \wedge z \equiv \mathrm{DummyR})$	*for A c.r. and B n.c.*
$(A \wedge z \equiv \mathrm{DummyL}) \vee^{\mathrm{nc}} \exists_y(y\ \mathbf{r}\ B \wedge z \equiv \mathrm{Inr}(y))$	*for A n.c. and B c.r.*
$(A \wedge z \equiv \mathtt{tt}) \vee^{\mathrm{nc}} (B \wedge z \equiv \mathtt{ff})$	*for A, B n.c.*

Proof. As an example consider the case A n.c., B c.r. Recall $\mathrm{OrR}_{X^{\mathrm{nc}},Y^c} := \mu_Z(X^{\mathrm{nc}} \to Z, Y^c \to Z)$. Then

$$\mathrm{OrR}^{\mathbf{r}}_{X^{\mathrm{nc}},Y^{\mathbf{r}}} := \mu_{Z^{\mathbf{r}}}(X^{\mathrm{nc}} \to \mathrm{DummyL} \in Z^{\mathbf{r}}, \forall_y(y\ \mathbf{r}\ Y \to \mathrm{Inr}(y) \in Z^{\mathbf{r}})).$$

Substituting X^{nc} by A and $Y^{\mathbf{r}}$ by $\{\, y \mid y\ \mathbf{r}\ B\,\}$ gives

$$(\mathrm{OrR}^{\mathbf{r}}_{A,\{y|y\mathbf{r}B\}})^+_0 \colon A \to \mathrm{DummyL}\ \mathbf{r}\ (A \vee B),$$
$$(\mathrm{OrR}^{\mathbf{r}}_{A,\{y|y\mathbf{r}B\}})^+_1 \colon \forall_y(y\ \mathbf{r}\ B \to \mathrm{Inr}(y)\ \mathbf{r}\ (A \vee B)).$$

This suffices for "$\leftarrow$": if $A \wedge z \equiv \mathrm{DummyL}$, then from $(\mathrm{OrR}^{\mathbf{r}}_{A,\{y|y\mathbf{r}B\}})^+_0$ we obtain $z\ \mathbf{r}\ (A \vee B)$, and if we have y with $y\ \mathbf{r}\ B$ and $z \equiv \mathrm{Inr}(y)$, then from $(\mathrm{OrR}^{\mathbf{r}}_{A,\{y|y\mathbf{r}B\}})^+_1$ we again obtain $z\ \mathbf{r}\ (A \vee B)$.

Conversely, the elimination axiom $(\mathrm{OrR}^{\mathbf{r}}_{X^{\mathrm{nc}},Y^{\mathbf{r}}})^-$ is

$$(X^{\mathrm{nc}} \to \mathrm{DummyL} \in Z) \to \forall_y(y\ \mathbf{r}\ Y \to \mathrm{Inr}(y) \in Z) \to \mathrm{OrR}^{\mathbf{r}}_{X^{\mathrm{nc}},Y^{\mathbf{r}}} \subseteq Z.$$

Substitute Z by $\{\, z \mid (A \wedge z \equiv \mathrm{DummyL}) \vee^{\mathrm{nc}} \exists_y (y \mathbf{\ r\ } B \wedge z \equiv \mathrm{Inr}(y)) \,\}$. Then with A for X^{nc} and $\{\, y \mid y \mathbf{\ r\ } B \,\}$ for $Y^{\mathbf{r}}$ the premises are provable. Hence

$$\forall_z (z \mathbf{\ r\ } (A \vee B) \to (A \wedge z \equiv \mathrm{DummyL}) \vee^{\mathrm{nc}} \exists_y (y \mathbf{\ r\ } B \wedge z \equiv \mathrm{Inr}(y))). \qquad \square$$

4.2. **Extracted terms.** Let M be a proof in TCF of a c.r. formula A. Assume M is an $\mathbf{r}$-free proof, i.e., M contains no realizability predicates $I^{\mathbf{r}}$ or ${}^{\mathrm{co}}I^{\mathbf{r}}$. We define its *extracted term* $\mathrm{et}(M)$, of type $\tau(A)$, with the aim to express M's computational content.

Let M be a proof in TCF of a c.r. formula A. Assume M is an $\mathbf{r}$-free proof, i.e., M contains no realizability predicates $I^{\mathbf{r}}$ or ${}^{\mathrm{co}}I^{\mathbf{r}}$. We define its *extracted term* $\mathrm{et}(M)$, of type $\tau(A)$, with the aim to express M's computational content. It will be a term built up from variables, constructors, recursion operators, destructors and corecursion operators by λ-abstraction and application.

Definition (Extracted term)**.** For an $\mathbf{r}$-free proof M of a c.r. formula A we define its extracted term $\mathrm{et}(M)$ by

$$\begin{aligned}
&\mathrm{et}(u^A) && := z_u^{\tau(A)} \quad (z_u^{\tau(A)} \text{ uniquely associated to } u^A),\\
&\mathrm{et}((\lambda_{u^A} M^B)^{A \to B}) && := \begin{cases} \lambda_{z_u} \mathrm{et}(M) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}\\
&\mathrm{et}((M^{A \to B} N^A)^B) && := \begin{cases} \mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}\\
&\mathrm{et}((\lambda_x M^A)^{\forall_x A}) && := \mathrm{et}(M),\\
&\mathrm{et}((M^{\forall_x A(x)} t)^{A(t)}) && := \mathrm{et}(M).
\end{aligned}$$

It remains to define extracted terms for the axioms. Consider a (c.r.) inductively defined predicate I. For its introduction and elimination axioms define $\mathrm{et}(I_i^+) := \mathcal{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where both the constructor $\mathcal{C}_i$ and the recursion operator $\mathcal{R}$ refer to the algebra ι_I associated with I. For the closure and greatest-fixed-point axioms of ${}^{\mathrm{co}}I$ define $\mathrm{et}({}^{\mathrm{co}}I^-) := \mathcal{D}$ and $\mathrm{et}({}^{\mathrm{co}}I_i^+) := {}^{\mathrm{co}}\mathcal{R}$, where both the destructor $\mathcal{D}$ and the corecursion operator ${}^{\mathrm{co}}\mathcal{R}$ refer to the cotype ${}^{\mathrm{co}}\iota_I$ where ι_I is the algebra associated with I. For the elimination axiom $(I^{\mathrm{nc}})^-$ of a one-clause-nc inductive predicate with a c.r. competitor predicate the extracted term is the identity.

One can see easily that the identity realizes the elimination axiom $(I^{\mathrm{nc}})^-$ of a one-clause-nc inductive predicate with a c.r. competitor predicate. More work is needed to show that the extracted term of $I^{\pm}$, ${}^{\mathrm{co}}I^{\pm}$ realizes the respective axiom. We prove this for a special case only, the algebras of lists and streams of "signed digits". Such objects are of interest for the representation of (dyadic) rational numbers and of real numbers.

Let $\sim_{\mathbb{D}}$ be the similarity relation for the three-element algebra $\mathbb{D}$ of signed digits 1, 0, -1 (written $\bar{1}$), defined by the three clauses $s \sim_{\mathbb{D}} s$ for s a signed digit. We will work with lists $\mathbb{L}(\mathbb{D})$ of signed digits and streams $\mathbb{S}(\mathbb{D})$ of signed digits, abbreviated $\mathbb{L}$ and $\mathbb{S}$. The similarity relation $\sim_{\mathbb{L}}$ has clauses

$$[] \sim_{\mathbb{L}} [], \qquad \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2)$$

and the elimination axiom $\sim_{\mathbb{L}}^{-}$:

$X[][] \to \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to X(s_1 :: \ell_1, s_2 :: \ell_2)) \to$
$\sim_{\mathbb{L}} \subseteq X$.

For the first two claims we only consider the inductive predicate $\sim_{\mathbb{L}}$.

Lemma 4.6. *The constructors of* $\mathbb{L}$ *realize the clauses of* $\sim_{\mathbb{L}}$.

Proof. We only consider the second constructor ::. We must show that :: realizes the following formula C equivalent to $(\sim_{\mathbb{L}})_1^+$:

$$\forall_{s_1,s_2}(s_1 \sim_{\mathbb{D}} s_2 \to \forall_{\ell_1,\ell_2}(\ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2))$$

i.e., $:: \mathbf{r}\, C$. Pick s_1, s_2. The goal then is

$$:: \mathbf{r}\ (s_1 \sim_{\mathbb{D}} s_2 \to \forall_{\ell_1,\ell_2}(\ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2)).$$

Pick s with $\sim_{\mathbb{D}}^{\mathbf{r}}(s_1, s_2, s)$. The goal then is

$$:: s\ \mathbf{r}\ \forall_{\ell_1,\ell_2}(\ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2).$$

Pick ℓ_1, ℓ_2, ℓ with $\sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell)$. The goal then is

$$(s :: \ell)\ \mathbf{r}\ (s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2), \quad \text{i.e.,} \quad \sim_{\mathbb{L}}^{\mathbf{r}}\ (s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)).$$

But this follows from what we have by the second clause of $\sim_{\mathbb{L}}^{\mathbf{r}}$:

$$\forall_{s_1,s_2,s,\ell_1,\ell_2,\ell}(\sim_{\mathbb{D}}^{\mathbf{r}}(s_1, s_2, s) \to \sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \to \sim_{\mathbb{L}}^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)). \quad \square$$

Lemma 4.7. *The recursion operator* $\mathcal{R}_{\mathbb{L}}^{\alpha}$ *realizes* $\sim_{\mathbb{L}}^{-}$.

Proof. We equivalently rewrite $\sim_{\mathbb{L}}^{-}$ as $C :=$

$$\forall_{\ell_1,\ell_2}(\ell_1 \sim_{\mathbb{L}} \ell_2 \to X[][] \to \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to$$
$$X(s_1 :: \ell_1, s_2 :: \ell_2)) \to X\ell_1\ell_2)$$

to make its type the same as the one for $\mathcal{R}_{\mathbb{L}}^{\alpha}$:

$$\mathbb{L} \to \alpha \to (\mathbb{D} \to \mathbb{L} \to \alpha \to \alpha) \to \alpha.$$

We must show $\mathcal{R}_{\mathbb{L}}^{\alpha}\ \mathbf{r}\ C$. Pick ℓ_1, ℓ_2, ℓ, x with $\sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell)$ and $X^{\mathbf{r}}[][]x$. The goal then is

$\mathcal{R}_{\mathbb{L}}^{\alpha}\ell x\ \mathbf{r}\ (\forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to X(s_1 :: \ell_1, s_2 :: \ell_2)) \to$
$X\ell_1\ell_2)$.

Assume $f\ \mathbf{r}\ \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to X(s_1 :: \ell_1, s_2 :: \ell_2))$, which implies $\forall_{s_1,s_2,s,\ell_1,\ell_2,\ell,y}(\sim_{\mathbb{D}}(s_1, s_2, s) \to \sim_{\mathbb{L}}(\ell_1, \ell_2, \ell) \to X^{\mathbf{r}}\ell_1\ell_2 y \to X^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, fs\ell y))$. Our goal is

$$X^{\mathbf{r}}(\ell_1, \ell_2, \mathcal{R}_{\mathbb{L}}^{\alpha}\ell x f) =: Q\ell_1\ell_2\ell.$$

To this end we use the elimination axiom for $\sim_{\mathbb{L}}^{\mathbf{r}}$:

$$\forall_{\ell_1,\ell_2,\ell}(\sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \to Q[][][] \to \forall_{s_1,s_2,s,\ell_1,\ell_2,\ell}(\sim_{\mathbb{D}}^{\mathbf{r}}(s_1, s_2, s) \to \sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \to$$
$$Q\ell_1\ell_2\ell \to Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)) \to Q\ell_1\ell_2\ell).$$

It suffices to prove the premises $Q[][][]$ and $\forall_{s_1,s_2,s,\ell_1,\ell_2,\ell}(\sim_{\mathbb{D}}^{\mathbf{r}}(s_1, s_2, s) \to \sim_{\mathbb{L}}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \to Q\ell_1\ell_2\ell \to Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell))$. By a computation rule for $\mathcal{R}_{\mathbb{L}}^{\alpha}$ the former is $X^{\mathbf{r}}[][]x$, which we have. For the latter assume s_1, s_2, s, ℓ_1, ℓ_2, ℓ and its premises. We show $Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)$, i.e.,

$$X^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, \mathcal{R}_{\mathbb{L}}^{\alpha}(s :: \ell)xf).$$

By the computation rules for $\mathcal{R}^\alpha_{\mathbb{L}}$ this is the same as

$$X^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, fs\ell(\mathcal{R}^\alpha_{\mathbb{L}}\ell x f)).$$

But with $y := \mathcal{R}^\alpha_{\mathbb{L}}\ell x f$ this follows from what we have. □

The bisimilarity relation $\approx_{\mathbb{S}}$ is defined by the closure axiom

$$\begin{aligned}\approx_{\mathbb{S}}^- : \forall_{u_1,u_2}(u_1 \approx_{\mathbb{S}} u_2 \to &\\ &\exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge u_1' \approx_{\mathbb{S}} u_2' \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2'))\end{aligned}$$

and the greatest-fixed-point axiom $\approx_{\mathbb{S}}^+$:

$$\begin{aligned}&\forall_{u_1,u_2}(Xu_1u_2 \to \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2') \wedge \\ &\qquad\qquad u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2')) \to \\ &X \subseteq \approx_{\mathbb{S}}.\end{aligned}$$

For the final two claims we only consider the coinductive predicate $\approx_{\mathbb{S}}$.

Lemma 4.8. *The destructor $\mathcal{D}_{\mathbb{S}}$ realizes the closure axiom $\approx_{\mathbb{S}}^-$.*

Proof. Recall $\approx_{\mathbb{S}}^-$: $\forall_{u_1,u_2}(u_1 \approx_{\mathbb{S}} u_2 \to \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge u_1' \approx_{\mathbb{S}} u_2' \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2'))$ with cotype ${}^{\mathrm{co}}\mathbb{S} \to \mathbb{D} \times {}^{\mathrm{co}}\mathbb{S}$. The goal is $\mathcal{D}_{\mathbb{S}}\ \mathbf{r} \approx_{\mathbb{S}}^-$:

$$\begin{aligned}&\forall_{u_1,u_2,u}(\approx^{\mathbf{r}}_{\mathbb{S}}(u_1,u_2,u) \to \mathcal{D}_{\mathbb{S}}u\ \mathbf{r}\ \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge u_1' \approx_{\mathbb{S}} u_2' \wedge \\ &\qquad\qquad u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2')).\end{aligned}$$

Assume $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1,u_2,u)$. We need to prove

$$\exists_{s_1,s_2,u_1',u_2'}(\mathcal{D}_{\mathbb{S}}u\ \mathbf{r}\ (s_1 \sim_{\mathbb{D}} s_2 \wedge u_1' \approx_{\mathbb{S}} u_2') \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2').$$

By $(\approx^{\mathbf{r}}_{\mathbb{S}})^-$ from $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1,u_2,u)$ we obtain s_1, s_2, s, u_1', u_2', u' such that

$$\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \wedge \approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2') \wedge u \equiv s :: u'.$$

Take s_1, s_2, u_1', u_2'. It remains to show $\mathcal{D}_{\mathbb{S}}u\ \mathbf{r}\ (s_1 \sim_{\mathbb{D}} s_2 \wedge u_1' \approx_{\mathbb{S}} u_2')$. By the computation rule of $\mathcal{D}_{\mathbb{S}}$ we know $\mathcal{D}_{\mathbb{S}}u \equiv \mathcal{D}_{\mathbb{S}}(s :: u') \equiv \langle s, u'\rangle$. Hence we must prove $\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s)$ and $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1,u_2,u)$, which we both have. □

Lemma 4.9. *The corecursion operator ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$ realizes $\approx_{\mathbb{S}}^+$.*

Proof. Equivalently rewrite $\approx_{\mathbb{S}}^+$ as $C := \forall_{u_1,u_2}(Xu_1u_2 \to \forall_{u_1,u_2}(Xu_1u_2 \to \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2') \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2')) \to u_1 \approx_{\mathbb{S}} u_2)$ to make its cotype the same as the one for ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$:

$$\alpha \to (\alpha \to \mathbb{D} \times ({}^{\mathrm{co}}\mathbb{S} + \alpha)) \to {}^{\mathrm{co}}\mathbb{S}.$$

We show that ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$ realizes C, i.e., ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}\ \mathbf{r}\ C$. The goal then is

$$\begin{aligned}&{}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}\ \mathbf{r}\ (Xu_1u_2 \to \\ &\qquad \forall_{u_1,u_2}(Xu_1u_2 \to \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2') \wedge \\ &\qquad\qquad u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2')) \to \\ &\qquad u_1 \approx_{\mathbb{S}} u_2).\end{aligned}$$

Pick u with $X^{\mathbf{r}}u_1u_2u$ and f such that

$$\begin{aligned}&\forall_{u_1,u_2,u}(X^{\mathbf{r}}u_1u_2u \to fu\ \mathbf{r}\ \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2') \wedge \\ &\qquad\qquad u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2')).\end{aligned}$$

Our goal is $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1, u_2, {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} uf)$. To this end we use $(\approx^{\mathbf{r}}_{\mathbb{S}})^+$ in the form

$$\begin{aligned}
&\forall_{u_1,u_2,u}(Qu_1u_2u \to \\
&\qquad \forall_{u_1,u_2,u}(Qu_1u_2u \to \\
&\qquad\qquad \exists_{s_1,s_2,s,u_1',u_2',u'}(\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \wedge (\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \vee Qu_1u_2u') \wedge \\
&\qquad\qquad\qquad u_1 \equiv s_1{::}u_1' \wedge u_2 \equiv s_2{::}u_2') \wedge u \equiv s{::}u')) \to \\
&\qquad \approx^{\mathbf{r}}_{\mathbb{S}}(u_1,u_2,u))
\end{aligned}$$

with

$$\exists_{z'}(X^{\mathbf{r}}u_1u_2z' \wedge u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} z'f) =: Qu_1u_2u.$$

It suffices to prove the closure property of Q. Let u_1, u_2, u and also u' be given such that $X^{\mathbf{r}}u_1u_2u' \wedge u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u'f$. We need to show

$$\begin{aligned}
&\exists_{s_1,s_2,s,u_1',u_2',u'}(\\
(6)\qquad &\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \wedge (\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \vee \exists_{u'}(X^{\mathbf{r}}u_1u_2u' \wedge u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u'f)) \wedge \\
&u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2' \wedge u \equiv s :: u').
\end{aligned}$$

Since $\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s)$ is equivalent to $s_1 \equiv s_2 \equiv s$ and $X^{\mathbf{r}}u_1u_2u'$ we know

$$fu' \;\mathbf{r}\; \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2') \wedge u_1 \equiv s_1 :: u_1' \wedge u_2 \equiv s_2 :: u_2').$$

Then $fu' \equiv \langle s, w\rangle$ with $\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s)$ and $w \;\mathbf{r}\; (u_1' \approx_{\mathbb{S}} u_2' \vee Xu_1'u_2')$, for some s_1, s_2, u_1', u_2' such that $u_1 \equiv s_1 :: u_1'$ and $u_2 \equiv s_2 :: u_2'$. Hence

$$\exists_{u'}(\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \wedge w \equiv \mathrm{InL}(u')) \vee \exists_{u''}(X^{\mathbf{r}}u_1'u_2'u'' \wedge w \equiv \mathrm{InR}(u'')).$$

We distinguish cases on this disjunction. Recall

$${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} uf \equiv \begin{cases} s :: u & \text{if } fu \equiv \langle s, \mathrm{InL}(u)\rangle, \\ s :: {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u'f & \text{if } fu \equiv \langle s, \mathrm{InR}(u')\rangle. \end{cases}$$

Case L. $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \wedge w \equiv \mathrm{InL}(u')$ for some u'. Then (6) holds, since $u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u'f \equiv s :: u'$.

Case R. $X^{\mathbf{r}}u_1'u_2'u'' \wedge w \equiv \mathrm{InR}(u'')$ for some u''. Then again (6) holds with $u' := {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u''f$, since $u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u'f \equiv s :: {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} u''f \equiv s :: u'$. □

4.3. Soundness.

Constructively to state A means the same as to say that A has a realizer. This statement $A \leftrightarrow \exists_x(x \;\mathbf{r}\; A)$ was called "to assert is to realize" in Feferman (1979). Here we call it invariance axiom, since it expresses invariance of A under the realizability interpretation. Using the invariance axioms we will prove the soundness theorem.

Axiom (Invariance). *For* $\mathbf{r}$*-free c.r. formulas* A *we require*

$$\text{(7)}\qquad \mathrm{InvAll}_A\colon \forall_z(z \;\mathbf{r}\; A \to A).$$

$$\text{(8)}\qquad \mathrm{InvEx}_A\colon A \to \exists_z(z \;\mathbf{r}\; A).$$

Theorem 4.10 (Soundness). *Let* M *be an* $\mathbf{r}$*-free derivation of a formula* A *from assumptions* $u_i\colon C_i$ $(i < n)$*. Then we can derive*

$$\begin{cases} \mathrm{et}(M) \;\mathbf{r}\; A & \textit{if } A \textit{ is c.r.} \\ A & \textit{if } A \textit{ is n.c.} \end{cases}$$

from assumptions

$$\begin{cases} z_{u_i} \mathbf{r}\, C_i & \textit{if } C_i \textit{ is c.r.} \\ C_i & \textit{if } C_i \textit{ is n.c.} \end{cases}$$

Proof. By induction on M. The axiom cases have been done before, and from the remaining cases we only treat the ones using invariance axioms.

Case $(\lambda_{u^A} M^B)^{A \to B}$ with B n.c. and A c.r. We need a derivation of $A \to B$. By induction hypothesis we have a derivation of B from z $\mathbf{r}$ A. Using the invariance axiom $A \to \exists_z (z \; \mathbf{r} \; A)$ we obtain the required derivation of B from A as follows.

$$\dfrac{\dfrac{A \to \exists_z (z \; \mathbf{r} \; A) \qquad A}{\exists_z (z \; \mathbf{r} \; A)} \qquad \begin{matrix}[z \; \mathbf{r} \; A] \\ \mid \text{IH} \\ B\end{matrix}}{B}\ \exists^-$$

Case $(M^{A \to B} N^A)^B$ with B n.c. and A c.r. We need a derivation of B. By induction hypothesis we have derivations of $A \to B$ and of $\mathrm{et}(N)$ $\mathbf{r}$ A. Using the invariance axiom $\forall_z (z \; \mathbf{r} \; A \to A)$ we obtain the required derivation from

$$\dfrac{\dfrac{\forall_z (z \; \mathbf{r} \; A \to A) \qquad \mathrm{et}(N)}{\mathrm{et}(N) \; \mathbf{r} \; A \to A} \qquad \begin{matrix} \mid \text{IH} \\ \mathrm{et}(N) \; \mathbf{r} \; A\end{matrix}}{A}$$

and the derivation of $A \to B$. □

4.4. **Extensionality of extracted terms.** Let I be an inductive predicate and ι_I its associated algebra. One can show that

- every constructor of ι_I is extensional w.r.t. its clause I_i^+,
- $\mathcal{R}_{\iota_I}^{\alpha}$ is extensional w.r.t. the least-fixed-point axiom I^-,
- the destructor of ι_I is extensional w.r.t. the closure axiom ${}^{\mathrm{co}}I^-$, and
- ${}^{\mathrm{co}}\mathcal{R}_{\iota_I}^{\alpha}$ is extensional w.r.t. the greatest-fixed-point axiom ${}^{\mathrm{co}}I^+$.

We prove these claims for special cases only. For the first two claims we consider the inductive predicate $\sim_{\mathbb{L}}$.

Lemma 4.11. *The constructors of* $\mathbb{L}$ *are extensional w.r.t.* $\sim_{\mathbb{L}}$*'s clauses.*

Proof. We only consider the second constructor $\mathcal{C}$. The goal is to show that $\mathcal{C}$ is extensional w.r.t. the cotype $\mathbb{D} \to \mathbb{L} \to \mathbb{L}$ of $\sim_{\mathbb{L}}$'s second clause, which by definition of $\doteq$ means

$$\forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2).$$

But this is the second clause of $\sim_{\mathbb{L}}$. □

Lemma 4.12. $\mathcal{R}_{\mathbb{L}}^{\alpha}$ *is extensional w.r.t. the least-fixed-point axiom* $\sim_{\mathbb{L}}^-$.

Proof. We equivalently rewrite $\sim_{\mathbb{L}}^-$ as $C :=$

$$\forall_{\ell_1, \ell_2} (\ell_1 \sim_{\mathbb{L}} \ell_2 \to X[][] \to \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X \ell_1 \ell_2 \to \\ X(s_1 :: \ell_1, s_2 :: \ell_2)) \to \ell_1 \ell_2)$$

to make its cotype the same as the one for $\mathcal{R}_{\mathbb{L}}^{\alpha}$:

$$\mathbb{L} \to \alpha \to (\mathbb{D} \to \mathbb{L} \to \alpha \to \alpha) \to \alpha.$$

We must show $\mathcal{R}^\alpha_{\mathbb{L}} \doteq_C \mathcal{R}^\alpha_{\mathbb{L}}$ with $\alpha := \varphi(X)$. By definition of $\doteq_C$ this is equivalent to

$$\forall_{x_1,x_2,f_1,f_2,\ell_1,\ell_2}(x_1 \doteq_\alpha x_2 \to f_1 \doteq_{\mathbb{D}\to\mathbb{L}\to\alpha\to\alpha} f_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to \mathcal{R}^\alpha_{\mathbb{L}}\ell_1 x_1 f_1 \doteq_\alpha \mathcal{R}^\alpha_{\mathbb{L}}\ell_2 x_2 f_2).$$

Assume $x_1 \doteq_\alpha x_2$ and $f_1 \doteq_{\mathbb{D}\to\mathbb{L}\to\alpha\to\alpha} f_2$. Use the least-fixed-point axiom $\sim^-_{\mathbb{L}}$ (in its original form) with competitor predicate

$$X := \{\, \ell_1, \ell_2 \mid \mathcal{R}^\alpha_{\mathbb{L}}\ell_1 x_1 f_1 \doteq_\alpha \mathcal{R}^\alpha_{\mathbb{L}}\ell_2 x_2 f_2 \,\}.$$

Case []. By the computation rules for $\mathcal{R}^\alpha_{\mathbb{L}}$ the claim $X[\,][\,]$ follows from $x_1 \doteq_\alpha x_2$. *Case* ::. Assume $s_1 \sim_{\mathbb{D}} s_2$ and $f_1 \doteq_{\mathbb{D}\to\mathbb{L}\to\alpha\to\alpha} f_2$. Let $y_1 := \mathcal{R}^\alpha_{\mathbb{L}}\ell_1 x_1 f_1$ and $y_2 := \mathcal{R}^\alpha_{\mathbb{L}}\ell_2 x_2 f_2$. Then $y_1 \doteq_\alpha y_2$ by assumption. The goal $f_1 s_1 \ell_1 y_1 \doteq_\alpha f_2 \ell_2 x_2 y_2$ follows from $f_1 \doteq f_2$, $s_1 \sim s_2$, $\ell_1 \sim \ell_2$ and $y_1 \doteq_\alpha y_2$. □

For the final two claims we only consider the coinductive predicate $\approx_{\mathbb{S}}$.

Lemma 4.13. *The destructor $\mathcal{D}_{\mathbb{S}}$ is extensional w.r.t. $\approx^-_{\mathbb{S}}$.*

Proof. The closure axiom $\approx^-_{\mathbb{S}}$ has cotype ${}^{\mathrm{co}}\mathbb{S} \to \mathbb{D} \times {}^{\mathrm{co}}\mathbb{S}$. The goal is $\mathcal{D}_{\mathbb{S}} \doteq_{({}^{\mathrm{co}}\mathbb{S}\to\mathbb{D}\times{}^{\mathrm{co}}\mathbb{S})} \mathcal{D}_{\mathbb{S}}$, which unfolds into

$$\forall_{u_1,u_2}(u_1 \approx_{\mathbb{S}} u_2 \to \mathcal{D}_{\mathbb{S}}u_1 \sim_{\mathbb{D}\times{}^{\mathrm{co}}\mathbb{S}} \mathcal{D}_{\mathbb{S}}u_2).$$

Assume $u_1 \approx_{\mathbb{S}} u_2$. By $\approx^-_{\mathbb{S}}$ we obtain s_1, s_2, u'_1, u'_2 with

$$s_1 \sim_{\mathbb{D}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2 \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2.$$

By the computation rule for $\mathcal{D}_{\mathbb{S}}$ we have $\mathcal{D}_{\mathbb{S}}u_i \equiv \langle s_i, u'_i\rangle$. By the clause for $\sim_{\mathbb{D}\times{}^{\mathrm{co}}\mathbb{S}}$ this implies the claim $\mathcal{D}_{\mathbb{S}}u_1 \sim_{\mathbb{D}\times{}^{\mathrm{co}}\mathbb{S}} \mathcal{D}_{\mathbb{S}}u_2$. □

Lemma 4.14. *${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$ is extensional w.r.t. the greatest-fixed-point axiom $\approx^+_{\mathbb{S}}$.*

Proof. Equivalently rewrite $\approx^+_{\mathbb{S}}$ as $C := \forall_{u_1,u_2}(Xu_1u_2 \to \forall_{u_1,u_2}(Xu_1u_2 \to \exists_{s_1,s_2,u'_1,u'_2}(s_1 \sim_{\mathbb{D}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \to u_1 \approx_{\mathbb{S}} u_2)$ to make its cotype the same as the one for ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$:

$$\alpha \to (\alpha \to \mathbb{D} \times ({}^{\mathrm{co}}\mathbb{S} + \alpha)) \to {}^{\mathrm{co}}\mathbb{S}.$$

Call this cotype ψ. The goal is ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}} \doteq_\psi {}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}$, which unfolds into

$$\forall_{x_1,x_2}(x_1 \doteq_\alpha x_2 \to \forall_{f_1,f_2}(f_1 \doteq_{\alpha\to\mathbb{D}\times({}^{\mathrm{co}}\mathbb{S}+\alpha)} f_2 \to {}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}x_1f_1 \approx_{\mathbb{S}} {}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}x_2f_2)).$$

Assume $x_1 \doteq_\alpha x_2$ and $f_1 \doteq_{\alpha\to\mathbb{D}\times({}^{\mathrm{co}}\mathbb{S}+\alpha)} f_2$. Let $u_1 := {}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}x_1f_1$ and $u_2 := {}^{\mathrm{co}}\mathcal{R}^\alpha_{\mathbb{S}}x_2f_2$. To prove the goal $u_1 \approx_{\mathbb{S}} u_2$ we use coinduction, or more precisely $\approx^+_{\mathbb{S}}$ with competitor predicate

$$X := \{\, u_1, u_2 \mid \exists_{y_1,y_2}(u_1 \equiv {}^{\mathrm{co}}\mathcal{R}y_1f_1 \wedge u_2 \equiv {}^{\mathrm{co}}\mathcal{R}y_2f_2 \wedge y_1 \doteq_\alpha y_2) \,\}.$$

This means that we have to show

$$\exists_{s_1,s_2,u'_1,u'_2}($$
$$s_1 \sim_{\mathbb{D}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee \exists_{y_1,y_2}(u'_1 \equiv {}^{\mathrm{co}}\mathcal{R}y_1f_1 \wedge u'_2 \equiv {}^{\mathrm{co}}\mathcal{R}y_2f_2 \wedge y_1 \doteq_\alpha y_2)) \wedge$$
$$u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2).$$

From $x_1 \doteq_\alpha x_2$ and $f_1 \doteq_{\alpha\to\mathbb{D}\times({}^{\mathrm{co}}\mathbb{S}+\alpha)} f_2$ we obtain $f_1x_1 \sim_{\mathbb{D}\times({}^{\mathrm{co}}\mathbb{S}+\alpha)} f_2x_2$. By definition of $\sim_\times$ this implies the existence of s_1, s_2, a_1, a_2 with

$$f_1x_1 \equiv \langle s_1, a_1\rangle \wedge f_2x_2 \equiv \langle s_2, a_2\rangle \wedge s_1 \sim_{\mathbb{D}} s_2 \wedge a_1 \sim_{({}^{\mathrm{co}}\mathbb{S}+\alpha)} a_2,$$

and by definition of $\sim_+$ from $a_1 \sim_{({}^{\mathrm{co}}\mathbb{S}+\alpha)} a_2$ we obtain the disjunction

$$(a_1 \equiv \mathrm{InL}(u_1') \wedge a_2 \equiv \mathrm{InL}(u_2') \wedge u_1' \approx_{\mathbb{S}} u_2') \vee$$
$$(a_1 \equiv \mathrm{InR}(x_1') \wedge a_2 \equiv \mathrm{InR}(x_2') \wedge x_1' \doteq_\alpha x_2').$$

We argue by cases on this disjunction. Recall

$${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} xf \equiv \begin{cases} s :: u & \text{if } fx \equiv \langle s, \mathrm{InL}(u)\rangle, \\ s :: {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} x' f & \text{if } fx \equiv \langle s, \mathrm{InR}(x')\rangle. \end{cases}$$

Case L. Then we have s_1, s_2, u_1', u_2' with $s_1 \sim_{\mathbb{D}} s_2$ and $u_1' \approx_{\mathbb{S}} u_2'$ such that $f_i x_i \equiv \langle s_i, \mathrm{InL}(u_i')\rangle$. Hence $u_i := {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} x_i f_i \equiv s_i :: u_i'$, and the claim follows.

Case R. Then we have s_1, s_2, x_1', x_2' with $s_1 \sim_{\mathbb{D}} s_2$ and $x_1' \doteq_\alpha x_2'$ such that $f_i x_i \equiv \langle s_i, \mathrm{InR}(x_i')\rangle$. Hence $u_i := {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} x_i f_i \equiv s_i :: u_i'$ with $u_i' := {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}} x_i' f_i$, and again the claim follows. □

We now prove compatibility of extracted terms with pointwise equality w.r.t. the cotype of the formula proved. For a convenient formulation we assume two more fixed assignments $u \mapsto z_u', z_u''$ of object variables to assumption variables.

Theorem 4.15 (Compatibility of extracted terms). *Let $M\colon A$ be a proof of a c.r. formula A and $u_i\colon C_i$ $(i = 1, \dots, n)$ all free c.r. assumptions whose associated object variable z_{u_i} is free in* $\mathrm{et}(M)$*. Then we can find a proof of*

$$\mathrm{et}(M)(z_{u_1}', \dots z_{u_n}') \doteq_A \mathrm{et}(M)(z_{u_1}'', \dots z_{u_n}'')$$

from assumptions $z_{u_i}' \doteq_{C_i} z_{u_i}''$ for $i = 1, \dots, n$.

Proof. By induction on M. *Case* $u\colon C$. Immediate. *Case* $c\colon A$ an axiom. This is clear in case the extracted term is the identity. For the axioms $I^{\pm}$ and ${}^{\mathrm{co}}I^{\pm}$ it was proved in Lemmas 4.11 - 4.14.

Case $(\lambda_{u^A} M^B)^{A \to B}$ with A c.r. For simplicity assume that u is the only assumption variable whose z_u is free in $\mathrm{et}(M)$. By IH we have a proof of $\mathrm{et}(M)(z_u') \doteq_A \mathrm{et}(M)(z_u'')$ from $z_u' \doteq_A z_u''$. We want a proof of $\mathrm{et}(\lambda_u M) \doteq_{A\to B} \mathrm{et}(\lambda_u M)$, i.e., $\lambda_{z_u}\mathrm{et}(M)(z_u) \doteq_{A\to B} \lambda_{z_u}\mathrm{et}(M)(z_u)$, which is

$$\forall_{z_u', z_u''}(z_u' \doteq_A z_u'' \to \mathrm{et}(M)(z_u') \doteq_B \mathrm{et}(M)(z_u'')).$$

Apply $\to^+$ and twice $\forall^+$ to the proof given by IH. In case A n.c. the extracted term $\mathrm{et}(\lambda_u M)$ is $\mathrm{et}(M)$ and the claim is immediate.

Case $M^{A\to B} N^A$ with A c.r. For simplicity assume that there no assumption variables whose associated object variable is free in $\mathrm{et}(MN)$. By IH_M we have a proof of $\mathrm{et}(M) \doteq_{A\to B} \mathrm{et}(M)$, i.e.,

$$\forall_{z_u', z_u''}(z_u' \doteq_A z_u'' \to \mathrm{et}(M)z_u' \doteq_B \mathrm{et}(M)z_u'').$$

By IH_N we have a proof of $\mathrm{et}(N) \doteq_A \mathrm{et}(N)$. Applying an instance of the first proof to the second gives $\mathrm{et}(M)\mathrm{et}(N) \doteq_B \mathrm{et}(M)\mathrm{et}(N)$, as required. In case A n.c. the extracted term $\mathrm{et}(MN)$ is $\mathrm{et}(M)$ and the claim is immediate.

Cases $\lambda_x M$, Mt. Obvious, since the extracted term does not change. □

Corollary 4.16 (Extensionality of extracted terms). *Let $M\colon A$ be a proof of a c.r. formula A and $u_i\colon C_i$ $(i = 1, \dots, n)$ all free c.r. assumptions whose associated object variable z_{u_i} is free in* $\mathrm{et}(M)$*. Then we can find a proof of* $\mathrm{et}(M) \doteq_A \mathrm{et}(M)$ *from assumptions $z_{u_i} \doteq_{C_i} z_{u_i}$ for $i = 1, \dots, n$.*

Proof. In the constructed proof substitute z'_{u_i}, z''_{u_i} by z_{u_i}. □

5. Applications

Space restrictions do not permit to go into applications, which are mainly in constructive analysis[2]. We can only refer to e.g. Berger et al. (2016); Schwichtenberg and Wiesnet (2021).

References

Ulrich Berger, Kenji Miyamoto, Helmut Schwichtenberg, and Hideki Tsuiki. Logic for Gray-code computation. In D. Probst and P. Schuster, editors, *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 69–110. De Gruyter, 2016.

Errett Bishop. Mathematics as a numerical language. In J. Myhill A. Kino and R.E. Vesley, editors, *Intuitionism and Proof Theory, Proceedings of the summer conference at Buffalo N.Y. 1968*, Studies in logic and the foundations of mathematics, pages 53–71. North-Holland, Amsterdam, 1970.

Solomon Feferman. Constructive theories of functions and classes. In K. McAloon M. Boffa, D. van Dalen, editor, *Logic Colloquium 78*, volume 97 of *Studies in Logic and the Foundations of Mathematics*, pages 159–224. North-Holland, Amsterdam, 1979.

Robin Gandy. *On axiomatic systems in mathematics and theories in physics.* PhD thesis, University of Cambridge, 1953.

Robin Gandy. On the axiom of extensionality – part I. *The Journal of Symbolic Logic*, 21(1):36–48, 1956.

Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.

Kim G. Larsen and Glynn Winskel. Using information systems to solve recursive domain equations. *Information and Computation*, 91:232–258, 1991.

Helmut Schwichtenberg and Stanley S. Wainer. *Proofs and Computations.* Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.

Helmut Schwichtenberg and Franziskus Wiesnet. Logic for exact real arithmetic. *Logical Methods in Computer Science*, 17(2), 2021. arxiv.org/abs/1904.12763.

Dana Scott. Domains for denotational semantics. In E. Nielsen and E.M. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *LNCS*, pages 577–613. Springer Verlag, Berlin, Heidelberg, New York, 1982.

Gaisi Takeuti. On a generalized logic calculus. *Japanese Journal of Mathematics*, 23:39–96, 1953.

Anne S. Troelstra. *Handbook of Proof Theory (ed. S. Buss)*, chapter Realizability, pages 408–473. Elsevier, 1998.

[2]see `minlog/examples/analysis`