Logic II

Helmut Schwichtenberg

Mathematisches Institut der LMU, Sommersemester 2021

Contents

Preface 1

Chapter 1. Logic 3
1.1. Natural deduction 3
1.2. Embedding intuitionistic and classical logic 9
1.3. The Curry-Howard correspondence 16

Chapter 2. Computability 19
2.1. Abstract computability via information systems 20
2.2. A term language for computable functionals 36
2.3. Denotational semantics 42

Chapter 3. A theory TCF of computable functionals 45
3.1. Formulas and their computational content 45
3.2. Axioms of TCF 51
3.3. Equality and extensionality w.r.t. cotypes 58

Chapter 4. Computational content of proofs 65
4.1. Realizers 65
4.2. Extracted terms, soundness 70
4.3. Extensionality of extracted terms 77

Chapter 5. Applications 83
5.1. Exact real arithmetic 83
5.2. Algorithms on stream-represented real numbers 90

Bibliography 99

Index 101

Preface

This is a script for the course "Logic II" at the Mathematics Institute of LMU, Sommersemester 2021. It is mainly based on the textbook "Proofs and Computations" in the references. I thank Nils Köpp and Philippe Vollmuth for their help with running the course and for many discussions concerning its content.

München, 13. July 2021
Helmut Schwichtenberg

CHAPTER 1

Logic

The main subject of Mathematical Logic is mathematical proof. In this chapter we deal with the basics of formalizing such proofs and analysing their structure. The system we pick for the representation of proofs is natural deduction as in Gentzen (1935). Our reasons for this choice are twofold. First, as the name says this is a *natural* notion of formal proof, which means that the way proofs are represented corresponds very much to the way a careful mathematician writing out all details of an argument would go anyway. Second, formal proofs in natural deduction are closely related (via the Curry-Howard correspondence) to terms in typed lambda calculus. This provides us not only with a compact notation for logical derivations (which otherwise tend to become somewhat unmanageable tree-like structures), but also opens up a route to applying the computational techniques which underpin lambda calculus.

An essential point for Mathematical Logic is to fix the formal language to be used. We take implication $\to$ and the universal quantifier $\forall$ as basic. Then the logic rules correspond precisely to lambda calculus. The additional connectives (i.e., the existential quantifier $\exists$, disjunction $\vee$ and conjunction $\wedge$) will be added via axioms. Later we will see that these axioms are determined by particular inductive definitions. In addition to the use of inductive definitions as a unifying concept, another reason for that change of emphasis will be that it fits more readily with the more computational viewpoint adopted there.

This chapter does not simply introduce basic proof theory, but in addition there is an underlying theme: to bring out the constructive content of logic, particularly in regard to the relationship between minimal and classical logic. It seems that the latter is most appropriately viewed as a subsystem of the former.

1.1. Natural deduction

Rules come in pairs: we have an introduction and an elimination rule for each of the logical connectives. The resulting system is called *minimal logic*; it was introduced by Kolmogorov (1932), Gentzen (1935) and Johansson

(1937). First we only consider implication $\to$ and universal quantification $\forall$. Note that no negation is yet present. If we go on and require *ex-falso-quodlibet* for a distinguished propositional variable $\bot$ ("falsum") we can embed *intuitionistic logic* with negation $\neg A$ defined as $A \to \bot$. To embed classical logic, we need to go further and add as an axiom schema the principle of *indirect proof*, also called *stability* ($\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x}$) for relation symbols R), but then it is appropriate to restrict to the language based on $\to$, $\forall$, $\bot$ and $\wedge$. The reason for this restriction is that we can neither prove $\neg\neg\exists_x A \to \exists_x A$ nor $\neg\neg(A \vee B) \to A \vee B$ (the former one for decidable A is Markov's scheme). However, we can prove them for the classical existential quantifier and disjunction defined by $\neg\forall_x\neg A$ and $\neg A \to \neg B \to \bot$. Thus we need to make a distinction between two kinds of "exists" and two kinds of "or": the classical ones are "weak" and the non-classical ones "strong" since they have constructive content. We mark the distinction by writing a tilde above the weak disjunction and existence symbols thus $\tilde{\vee}, \tilde{\exists}$.

1.1.1. Examples of derivations. To motivate the rules for natural deduction, let us start with informal proofs of some simple logical facts.

$$(A \to B \to C) \to (A \to B) \to A \to C.$$

Informal proof. Assume $A \to B \to C$. To show: $(A \to B) \to A \to C$. So assume $A \to B$. To show: $A \to C$. So finally assume A. To show: C. Using the third assumption twice we have $B \to C$ by the first assumption, and B by the second assumption. From $B \to C$ and B we then obtain C. Then $A \to C$, cancelling the assumption on A; $(A \to B) \to A \to C$ cancelling the second assumption; and the result follows by cancelling the first assumption. □

$$\forall_x(A \to B) \to A \to \forall_x B, \quad \text{if } x \notin \mathrm{FV}(A).$$

Informal proof. Assume $\forall_x(A \to B)$. To show: $A \to \forall_x B$. So assume A. To show: $\forall_x B$. Let x be arbitrary; note that we have not made any assumptions on x. To show: B. We have $A \to B$ by the first assumption. Hence also B by the second assumption. Hence $\forall_x B$. Hence $A \to \forall_x B$, cancelling the second assumption. Hence the result, cancelling the first assumption. □

A characteristic feature of these proofs is that assumptions are introduced and eliminated again. At any point in time during the proof the free or "open" assumptions are known, but as the proof progresses, free assumptions may become cancelled or "closed" because of the implies-introduction rule.

We reserve the word *proof* for the informal level; a formal representation of a proof will be called a *derivation*.

An intuitive way to communicate derivations is to view them as labelled trees each node of which denotes a rule application. The labels of the inner nodes are the formulas derived as conclusions at those points, and the labels of the leaves are formulas or terms. The labels of the nodes immediately above a node k are the *premises* of the rule application. At the root of the tree we have the conclusion (or end formula) of the whole derivation. In natural deduction systems one works with *assumptions* at leaves of the tree; they can be either *open* or *closed* (cancelled). Any of these assumptions carries a *marker*. As markers we use *assumption variables* denoted $u, v, w, u_0, u_1, \ldots$. The variables of the language previously introduced will now often be called *object variables*, to distinguish them from assumption variables. If at a node below an assumption the dependency on this assumption is removed (it becomes closed) we record this by writing down the assumption variable. Since the same assumption may be used more than once (this was the case in the first example above), the assumption marked with u (written $u\colon A$) may appear many times. Of course we insist that distinct assumption formulas must have distinct markers. An inner node of the tree is understood as the result of passing from premises to the conclusion of a given rule. The label of the node then contains, in addition to the conclusion, also the name of the rule. In some cases the rule binds or closes or cancels an assumption variable u (and hence removes the dependency of all assumptions $u\colon A$ thus marked). An application of the $\forall$-introduction rule similarly binds an object variable x (and hence removes the dependency on x). In both cases the bound assumption or object variable is added to the label of the node.

Definition. A formula A is called *derivable* (in *minimal logic*), written $\vdash A$, if there is a derivation of A (without free assumptions) using the natural deduction rules. A formula B is called derivable from assumptions $A_1, \ldots, A_n$, if there is a derivation of B with free assumptions among $A_1, \ldots, A_n$. Let Γ be a (finite or infinite) set of formulas. We write $\Gamma \vdash B$ if the formula B is derivable from finitely many assumptions $A_1, \ldots, A_n \in \Gamma$.

We now formulate the rules of natural deduction.

1.1.2. Introduction and elimination rules for $\to$ and $\forall$. First we have an assumption rule, allowing to write down an arbitrary formula A together with a marker u:

$$u\colon A \qquad \text{assumption.}$$

The other rules of natural deduction split into introduction rules (I-rules for short) and elimination rules (E-rules) for the logical connectives which, for the time being, are just $\to$ and $\forall$. For implication $\to$ there is an introduction

rule $\to^+$ and an elimination rule $\to^-$ also called *modus ponens*. The left premise $A \to B$ in $\to^-$ is called the *major* (or *main*) premise, and the right premise A the *minor* (or *side*) premise. Note that with an application of the $\to^+$-rule *all* assumptions above it marked with $u\colon A$ are cancelled (which is denoted by putting square brackets around these assumptions), and the u then gets written alongside. There may of course be other uncancelled assumptions $v\colon A$ of the same formula A, which may get cancelled at a later stage.

$$\frac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B}\to^+ u \qquad \frac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B}\to^-$$

For the universal quantifier $\forall$ there is an introduction rule $\forall^+$ (again marked, but now with the bound variable x) and an elimination rule $\forall^-$ whose right premise is the term t to be substituted. The rule $\forall^+ x$ with conclusion $\forall_x A$ is subject to the following *(eigen-)variable condition*: the derivation M of the premise A should not contain any open assumption having x as a free variable.

$$\frac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A}\forall^+ x \text{ (var.cond.)} \qquad \frac{\begin{array}{cc}\mid M & \\ \forall_x A(x) & t\end{array}}{A(t)}\forall^-$$

We now give derivations of the two example formulas treated informally above. Since in many cases the rule used is determined by the conclusion, we suppress in such cases the name of the rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon A \to B \to C \qquad w\colon A}{B \to C} \qquad \dfrac{v\colon A \to B \qquad w\colon A}{B}}{C}\to^+ w}{A \to C}\to^+ v}{(A \to B) \to A \to C}\to^+ u}{(A \to B \to C) \to (A \to B) \to A \to C}$$

For the second example we obtain

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon \forall_x(A \to Px) \qquad x}{A \to Px} \qquad v\colon A}{Px}}{\forall_x Px}\forall^+ x}{A \to \forall_x Px}\to^+ v}{\forall_x(A \to Px) \to A \to \forall_x Px}\to^+ u$$

Note that the variable condition is satisfied: x is not free in A (and also not free in $\forall_x(A \to Px)$).

1.1.3. Negation, disjunction, conjunction and existence. Recall that negation is defined by $\neg A := (A \to \bot)$. The following can easily be derived.

$$A \to \neg\neg A,$$
$$\neg\neg\neg A \to \neg A.$$

However, $\neg\neg A \to A$ is in general *not* derivable (without stability – we will come back to this later on). The derivation of $\neg\neg\neg A \to \neg A$ is

$$\dfrac{\dfrac{u\colon ((A \to \bot) \to \bot) \to \bot \qquad \dfrac{\dfrac{w\colon A \to \bot \qquad v\colon A}{\bot}}{(A \to \bot) \to \bot} \to^+ w}{\dfrac{\bot}{A \to \bot} \to^+ v}}{((((A \to \bot) \to \bot) \to \bot) \to A \to \bot} \to^+ u$$

Derivations for the following formulas are left as exercises.

$$(A \to B) \to \neg B \to \neg A,$$
$$\neg(A \to B) \to \neg B,$$

$$\neg\neg(A \to B) \to \neg\neg A \to \neg\neg B,$$
$$(\bot \to B) \to (\neg\neg A \to \neg\neg B) \to \neg\neg(A \to B),$$

$$\neg\neg\forall_x A \to \forall_x \neg\neg A.$$

For disjunction the introduction and elimination axioms are

$$\vee_0^+\colon A \to A \vee B,$$
$$\vee_1^+\colon B \to A \vee B,$$
$$\vee^-\colon A \vee B \to (A \to C) \to (B \to C) \to C.$$

For conjunction we have

$$\wedge^+\colon A \to B \to A \wedge B, \qquad \wedge^-\colon A \wedge B \to (A \to B \to C) \to C$$

and for the existential quantifier

$$\exists^+\colon A \to \exists_x A, \qquad \exists^-\colon \exists_x A \to \forall_x (A \to B) \to B \quad (x \notin \mathrm{FV}(B)).$$

Remark. All these axioms can be seen as special cases of a general schema, that of an *inductively defined predicate*, which is defined by some introduction rules and one elimination rule. Later we will study this kind of definition in full generality.

It is easy to see that for each of the connectives $\vee$, $\wedge$, $\exists$ the axioms and the following rules are equivalent over minimal logic; this is left as an exercise. For disjunction the introduction and elimination rules are

$$\frac{\begin{array}{c}\mid M\\ A\end{array}}{A \vee B}\vee_0^+ \qquad \frac{\begin{array}{c}\mid M\\ B\end{array}}{A \vee B}\vee_1^+ \qquad \frac{\begin{array}{ccc} & [u\colon A] & [v\colon B]\\ \mid M & \mid N & \mid K\\ A \vee B & C & C\end{array}}{C}\vee^- u, v$$

For conjunction we have

$$\frac{\begin{array}{cc}\mid M & \mid N\\ A & B\end{array}}{A \wedge B}\wedge^+ \qquad \frac{\begin{array}{ccc} & [u\colon A] & [v\colon B]\\ \mid M & & \mid N\\ A \wedge B & & C\end{array}}{C}\wedge^- u, v$$

and for the existential quantifier

$$\frac{\begin{array}{cc} & \mid M\\ t & A(t)\end{array}}{\exists_x A(x)}\exists^+ \qquad \frac{\begin{array}{cc} & [u\colon A]\\ \mid M & \mid N\\ \exists_x A & B\end{array}}{B}\exists^- x, u \text{ (var.cond.)}$$

Similar to $\forall^+ x$ the rule $\exists^- x, u$ is subject to an *(eigen-)variable condition*: in the derivation N the variable x (i) should not occur free in the formula of any open assumption other than $u\colon A$, and (ii) should not occur free in B.

We collect some easy facts about derivability; $B \leftarrow A$ means $A \to B$.

LEMMA 1.1.1. *The following are derivable.*

$$\begin{aligned}
&(A \wedge B \to C) \leftrightarrow (A \to B \to C),\\
&(A \to B \wedge C) \leftrightarrow (A \to B) \wedge (A \to C),\\
&(A \vee B \to C) \leftrightarrow (A \to C) \wedge (B \to C),\\
&(A \to B \vee C) \leftarrow (A \to B) \vee (A \to C),\\
&(\forall_x A \to B) \leftarrow \exists_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),\\
&(A \to \forall_x B) \leftrightarrow \forall_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(A),\\
&(\exists_x A \to B) \leftrightarrow \forall_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),\\
&(A \to \exists_x B) \leftarrow \exists_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(A).
\end{aligned}$$

PROOF. A derivation of the final formula is

$$\dfrac{\dfrac{\dfrac{\dfrac{}{u\colon \exists_x(A\to B)}\qquad \dfrac{\dfrac{\dfrac{}{w\colon A\to B}\qquad \dfrac{}{v\colon A}}{B}}{\exists_x B}\,x}{\exists_x B}\,\exists^-x,w}{A\to\exists_x B}\to^+ v}{\exists_x(A\to B)\to A\to\exists_x B}\to^+ u$$

The variable condition for $\exists^-$ is satisfied since the variable x (i) is not free in the formula A of the open assumption $v\colon A$, and (ii) is not free in $\exists_x B$. The rest of the proof is left as an exercise. □

1.2. Embedding intuitionistic and classical logic

As already mentioned, we distinguish two kinds of "or" and "exists": the "weak" or classical ones and the "strong" or constructive ones. In the present context both kinds occur together and hence we must mark the distinction; we shall do this by writing a tilde above the weak disjunction and existence symbols thus

$$A\mathbin{\tilde\vee} B := \neg A\to\neg B\to\bot,\qquad \tilde\exists_x A := \neg\forall_x\neg A.$$

These weak variants of disjunction and the existential quantifier are no stronger than the proper ones (in fact, they are weaker):

$$A\vee B\to A\mathbin{\tilde\vee} B,\qquad \exists_x A\to\tilde\exists_x A.$$

This can be seen easily by putting $C := \bot$ in $\vee^-$ and $B := \bot$ in $\exists^-$.

REMARK. Since $\tilde\exists_x\tilde\exists_y A$ unfolds into a rather awkward formula we extend the $\tilde\exists$-terminology to lists of variables:

$$\tilde\exists_{x_1,\dots,x_n} A := \forall_{x_1,\dots,x_n}(A\to\bot)\to\bot.$$

Nothing is lost here, since the omitted double negations could be eliminated. Moreover let

$$\tilde\exists_{x_1,\dots,x_n}(A_1\mathbin{\tilde\wedge}\dots\mathbin{\tilde\wedge}A_m) := \forall_{x_1,\dots,x_n}(A_1\to\dots\to A_m\to\bot)\to\bot.$$

This allows to stay in the $\to,\forall$ part of the language. Notice that $\tilde\wedge$ only makes sense in this context, i.e., in connection with $\tilde\exists$.

1.2.1. Intuitionistic and classical derivability. In the definition of derivability falsity $\bot$ plays no role. We may change this and require *ex-falso-quodlibet* axioms, of the form

$$\forall_{\vec x}(\bot\to R\vec x\,)$$

with R a relation symbol distinct from $\bot$. Let Efq denote the set of all such axioms. A formula A is called *intuitionistically derivable*, written $\vdash_i A$, if $\mathrm{Efq} \vdash A$. We write $\Gamma \vdash_i B$ for $\Gamma \cup \mathrm{Efq} \vdash B$.

We may even go further and require *stability* axioms, of the form

$$\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x}\,)$$

with R again a relation symbol distinct from $\bot$. Let Stab denote the set of all these axioms. A formula A is called *classically derivable*, written $\vdash_c A$, if $\mathrm{Stab} \vdash A$. We write $\Gamma \vdash_c B$ for $\Gamma \cup \mathrm{Stab} \vdash B$.

It is easy to see that intuitionistically (i.e., from Efq) we can derive $\bot \to A$ for an *arbitrary* formula A, using the introduction rules for the connectives. A similar generalization of the stability axioms is only possible for formulas in the language not involving $\vee, \exists$. However, it is still possible to use the substitutes $\tilde{\vee}$ and $\tilde{\exists}$.

THEOREM 1.2.1 (Stability, or principle of indirect proof).

(a) $\vdash (\neg\neg A \to A) \to (\neg\neg B \to B) \to \neg\neg(A \wedge B) \to A \wedge B$.
(b) $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$.
(c) $\vdash (\neg\neg A \to A) \to \neg\neg\forall_x A \to A$.
(d) $\vdash_c \neg\neg A \to A$ *for every formula* A *without* $\vee, \exists$.

PROOF. (a) is left as an exercise.

(b) For simplicity, in the derivation to be constructed we leave out applications of $\to^+$ at the end.

$$\dfrac{u\colon \neg\neg B \to B \qquad \dfrac{\dfrac{v\colon \neg\neg(A \to B) \qquad \dfrac{\dfrac{u_1\colon \neg B \qquad \dfrac{u_2\colon A \to B \qquad w\colon A}{B}}{\bot}\to^+ u_2}{\neg(A \to B)}}{\bot}\to^+ u_1}{\neg\neg B}}{B}$$

(c)

$$\dfrac{u\colon \neg\neg A \to A \qquad \dfrac{\dfrac{v\colon \neg\neg\forall_x A \qquad \dfrac{\dfrac{u_1\colon \neg A \qquad \dfrac{u_2\colon \forall_x A \qquad x}{A}}{\bot}\to^+ u_2}{\neg\forall_x A}}{\bot}\to^+ u_1}{\neg\neg A}}{A}$$

(d) Induction on A. The case $R\vec{t}$ with R distinct from $\bot$ is given by Stab. In the case $\bot$ the desired derivation is

$$\dfrac{u\colon (\bot \to \bot) \to \bot \qquad \dfrac{v\colon \bot}{\bot \to \bot}\to^+ v}{\bot}$$

In the cases $A \wedge B$, $A \to B$ and $\forall_x A$ use (a), (b) and (c), respectively. $\square$

Using stability we can prove some well-known facts about the interaction of weak disjunction and the weak existential quantifier with implication. We first prove a more refined claim, stating to what extent we need to go beyond minimal logic.

LEMMA 1.2.2. *The following are derivable.*

(1) $(\tilde{\exists}_x A \to B) \to \forall_x(A \to B)$ *if* $x \notin \mathrm{FV}(B)$,

(2) $(\neg\neg B \to B) \to \forall_x(A \to B) \to \tilde{\exists}_x A \to B$ *if* $x \notin \mathrm{FV}(B)$,

(3) $(\bot \to B[x{:=}c]) \to (A \to \tilde{\exists}_x B) \to \tilde{\exists}_x(A \to B)$ *if* $x \notin \mathrm{FV}(A)$,

(4) $\tilde{\exists}_x(A \to B) \to A \to \tilde{\exists}_x B$ *if* $x \notin \mathrm{FV}(A)$.

The last two items can also be seen as simplifying a weakly existentially quantified implication whose premise does not contain the quantified variable. In case the conclusion does not contain the quantified variable we have

(5) $(\neg\neg B \to B) \to \tilde{\exists}_x(A \to B) \to \forall_x A \to B$ *if* $x \notin \mathrm{FV}(B)$,

(6) $\forall_x(\neg\neg A \to A) \to (\forall_x A \to B) \to \tilde{\exists}_x(A \to B)$ *if* $x \notin \mathrm{FV}(B)$.

PROOF. (1)

$$\dfrac{\tilde{\exists}_x A \to B \qquad \dfrac{\dfrac{u_1\colon \forall_x \neg A \qquad x}{\neg A} \qquad A}{\bot}\to^+ u_1}{\neg\forall_x \neg A}}{B}$$

(2)

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall_x(A \to B) \qquad x}{A \to B} \qquad \dfrac{u_1\colon A}{\qquad}}{B}}{\bot}\to^+ u_1}{\neg A}}{\forall_x \neg A}}{\neg\forall_x \neg A} \qquad \dfrac{\dfrac{\dfrac{u_2\colon \neg B}{\qquad}}{\bot}\to^+ u_2}{\neg\neg B}}{B}$$

(3) Writing B_0 for $B[x{:=}c]$ we have

$$\dfrac{\dfrac{\forall_x\neg(A\to B)\quad c}{\neg(A\to B_0)}\qquad \dfrac{\dfrac{\dfrac{A\to\tilde\exists_x B\quad u_2\colon A}{\tilde\exists_x B}\qquad \dfrac{\dfrac{\dfrac{\forall_x\neg(A\to B)\quad x}{\neg(A\to B)}\qquad \dfrac{u_1\colon B}{A\to B}}{\bot}}{\neg B}\to^+u_1}{\forall_x\neg B}}{\bot}}{\dfrac{\bot\to B_0}{\dfrac{B_0}{A\to B_0}\to^+u_2}}}{\bot}$$

(4)

$$\dfrac{\tilde\exists_x(A\to B)\qquad \dfrac{\dfrac{\dfrac{\forall_x\neg B\quad x}{\neg B}\qquad \dfrac{u_1\colon A\to B\quad A}{B}}{\bot}}{\neg(A\to B)}\to^+u_1}{\forall_x\neg(A\to B)}}{\bot}$$

(5)

$$\dfrac{\neg\neg B\to B\qquad \dfrac{\dfrac{\tilde\exists_x(A\to B)\qquad \dfrac{\dfrac{u_2\colon\neg B\qquad \dfrac{u_1\colon A\to B\qquad \dfrac{\forall_x A\quad x}{A}}{B}}{\bot}}{\neg(A\to B)}\to^+u_1}{\forall_x\neg(A\to B)}}{\bot}}{\neg\neg B}\to^+u_2}{B}$$

(6) We derive $\forall_x(\bot\to A)\to(\forall_x A\to B)\to\forall_x\neg(A\to B)\to\neg\neg A$. Writing Ax, Ay for $A(x), A(y)$ we have

$$\dfrac{\dfrac{\forall_x\neg(Ax\to B)\quad x}{\neg(Ax\to B)}\qquad \dfrac{\dfrac{\forall_x Ax\to B\qquad \dfrac{\dfrac{\forall_y(\bot\to Ay)\quad y}{\bot\to Ay}\qquad \dfrac{u_1\colon\neg Ax\quad u_2\colon Ax}{\bot}}{Ay}}{\forall_y Ay}}{B}\to^+u_2}{Ax\to B}}{\bot}\to^+u_1}{\neg\neg Ax}$$

Using this derivation M we obtain

$$\dfrac{\dfrac{\forall_x\neg(Ax\to B)\quad x}{\neg(Ax\to B)}\qquad \dfrac{\forall_x Ax\to B\qquad \dfrac{\dfrac{\dfrac{\forall_x(\neg\neg Ax\to Ax)\quad x}{\neg\neg Ax\to Ax}\qquad \neg\neg Ax}{Ax}\;\mid M}{\forall_x Ax}}{B}}{Ax\to B}}{\bot}$$

Since clearly $\vdash(\neg\neg A\to A)\to\bot\to A$ the claim follows. □

Remark. An immediate consequence of (6) is the classical derivability of the "drinker formula" $\tilde\exists_x(Px\to\forall_x Px)$, to be read "in every non-empty bar there is a person such that, if this person drinks, then everybody drinks". To see this let $A:=Px$ and $B:=\forall_x Px$ in (6).

Corollary 1.2.3.

$\vdash_c(\tilde\exists_x A\to B)\leftrightarrow\forall_x(A\to B)$ *if* $x\notin\mathrm{FV}(B)$ *and* B *without* $\vee,\exists$,

$\vdash_i(A\to\tilde\exists_x B)\leftrightarrow\tilde\exists_x(A\to B)$ *if* $x\notin\mathrm{FV}(A)$,

$\vdash_c\tilde\exists_x(A\to B)\leftrightarrow(\forall_x A\to B)$ *if* $x\notin\mathrm{FV}(B)$ *and* A,B *without* $\vee,\exists$.

There is a similar lemma on weak disjunction:

Lemma 1.2.4. *The following are derivable.*

$$\begin{aligned}
&(A\,\tilde\vee\,B\to C)\to(A\to C)\wedge(B\to C),\\
(\neg\neg C\to C)\to\;&(A\to C)\to(B\to C)\to A\,\tilde\vee\,B\to C,\\
(\bot\to B)\to\;&(A\to B\,\tilde\vee\,C)\to(A\to B)\,\tilde\vee\,(A\to C),\\
&(A\to B)\,\tilde\vee\,(A\to C)\to A\to B\,\tilde\vee\,C,\\
(\neg\neg C\to C)\to\;&(A\to C)\,\tilde\vee\,(B\to C)\to A\to B\to C,\\
(\bot\to C)\to\;&(A\to B\to C)\to(A\to C)\,\tilde\vee\,(B\to C).
\end{aligned}$$

Proof. The derivation of the final formula is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A\to B\to C\quad u_1\colon A}{B\to C}\qquad u_2\colon B}{C}}{A\to C}\to^+u_1}{\neg(A\to C)}\qquad \dfrac{\bot\to C\qquad \bot}{C}}{B\to C}\to^+u_2}{\neg(B\to C)}}{\bot}$$

The other derivations are similar to the ones above, if one views $\tilde\exists$ as an infinitary version of $\tilde\vee$. □

COROLLARY 1.2.5.

$\vdash_c (A \mathbin{\tilde\vee} B \to C) \leftrightarrow (A \to C) \wedge (B \to C)$ *for C without $\vee, \exists$,*

$\vdash_i (A \to B \mathbin{\tilde\vee} C) \leftrightarrow (A \to B) \mathbin{\tilde\vee} (A \to C)$,

$\vdash_c (A \to C) \mathbin{\tilde\vee} (B \to C) \leftrightarrow (A \to B \to C)$ *for C without $\vee, \exists$.*

It is easy to see that weak disjunction and the weak existential quantifier satisfy the same axioms as the strong variants, if one restricts the conclusion of the elimination axioms to formulas without $\vee, \exists$. In fact, we have

LEMMA 1.2.6.

$\vdash A \to \tilde\exists_x A$,

$\vdash_c \tilde\exists_x A \to \forall_x (A \to B) \to B$ *($x \notin \mathrm{FV}(B)$, B without $\vee, \exists$),*

$\vdash A \to A \mathbin{\tilde\vee} B, \quad \vdash B \to A \mathbin{\tilde\vee} B$,

$\vdash_c A \mathbin{\tilde\vee} B \to (A \to C) \to (B \to C) \to C$ *(C without $\vee, \exists$).*

PROOF. The derivations of the second and the fourth formula are

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\dfrac{\neg\forall_x \neg A \qquad \dfrac{\dfrac{\dfrac{\forall_x(A \to B) \quad x}{A \to B} \qquad u_2 \colon A}{B}}{\bot}}{\neg\neg B}\to^+ u_1}{\forall_x \neg A}}{\bot}\to^+ u_2}{B}$$

and

$$\dfrac{\neg\neg C \to C \qquad \dfrac{\dfrac{\neg A \to \neg B \to \bot \qquad \dfrac{\dfrac{A \to C \quad u_2 \colon A}{C}}{\bot}}{\neg A}\to^+ u_2}{\neg B \to \bot} \qquad \dfrac{\dfrac{B \to C \quad u_3 \colon B}{C}}{\bot}\to^+ u_3}{\neg\neg C}\to^+ u_1}{C}$$

□

1.2.2. Gentzen translation. Classical derivability $\Gamma \vdash_c B$ was defined in Section 1.2.1 by $\Gamma \cup \mathrm{Stab} \vdash B$. This embedding of classical logic into minimal logic can be expressed in a somewhat different and very explicit form, namely as a syntactic translation $A \mapsto A^g$ of formulas such that A is derivable in classical logic if and only if its translation A^g is derivable in minimal logic.

DEFINITION (Gentzen translation A^g).

$$\begin{aligned}
(R\vec{t})^g &:= \neg\neg R\vec{t} \quad \text{for } R \text{ distinct from } \bot,\\
\bot^g &:= \bot,\\
(A \vee B)^g &:= A^g \mathbin{\tilde{\vee}} B^g,\\
(\exists_x A)^g &:= \tilde{\exists}_x A^g,\\
(A \circ B)^g &:= A^g \circ B^g \quad \text{for } \circ = \to, \wedge,\\
(\forall_x A)^g &:= \forall_x A^g.
\end{aligned}$$

LEMMA 1.2.7. $\vdash \neg\neg A^g \to A^g$.

PROOF. Induction on A.

Case $R\vec{t}$ with R distinct from $\bot$. We must show $\neg\neg\neg\neg R\vec{t} \to \neg\neg R\vec{t}$, which is a special case of $\vdash \neg\neg\neg B \to \neg B$.

Case $\bot$. Use $\vdash \neg\neg\bot \to \bot$.

Case $A \vee B$. We must show $\vdash \neg\neg(A^g \mathbin{\tilde{\vee}} B^g) \to A^g \mathbin{\tilde{\vee}} B^g$, which is a special case of $\vdash \neg\neg(\neg C \to \neg D \to \bot) \to \neg C \to \neg D \to \bot$:

$$\dfrac{\dfrac{\neg\neg(\neg C \to \neg D \to \bot)}{\bot}\quad \dfrac{\dfrac{\dfrac{\dfrac{u_1 : \neg C \to \neg D \to \bot \qquad \neg C}{\neg D \to \bot}\qquad \neg D}{\bot}}{\neg(\neg C \to \neg D \to \bot)}\to^+ u_1}{\bot}$$

Case $\exists_x A$. In this case we must show $\vdash \neg\neg\tilde{\exists}_x A^g \to \tilde{\exists}_x A^g$, but this is a special case of $\vdash \neg\neg\neg B \to \neg B$, because $\tilde{\exists}_x A^g$ is the negation $\neg\forall_x \neg A^g$.

Case $A \wedge B$. We must show $\vdash \neg\neg(A^g \wedge B^g) \to A^g \wedge B^g$. By induction hypothesis $\vdash \neg\neg A^g \to A^g$ and $\vdash \neg\neg B^g \to B^g$. Now use part (a) of Theorem 1.2.1 (on stability).

The cases $A \to B$ and $\forall_x A$ are similar, using parts (b) and (c) of Theorem 1.2.1 instead. □

THEOREM 1.2.8. (a) $\Gamma \vdash_c A$ *implies* $\Gamma^g \vdash A^g$.
(b) $\Gamma^g \vdash A^g$ *implies* $\Gamma \vdash_c A$ *for* Γ, A *without* $\vee, \exists$.

PROOF. (a) Use induction on $\Gamma \vdash_c A$. For a stability axiom $\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x})$ we must derive $\forall_{\vec{x}}(\neg\neg\neg\neg R\vec{x} \to \neg\neg R\vec{x})$, which is easy (as above). For the rules $\to^+$, $\to^-$, $\forall^+$, $\forall^-$, $\wedge^+$ and $\wedge^-$ the claim follows immediately from the induction hypothesis, using the same rule again. This works because the Gentzen translation acts as a homomorphism for these connectives. For the rules $\vee_i^+$, $\vee^-$, $\exists^+$ and $\exists^-$ the claim follows from the induction hypothesis

and Lemma 1.2.6. For example, in case $\exists^-$ the induction hypothesis gives

$$\begin{array}{c} \\ | \, M \\ \tilde{\exists}_x A^g \end{array} \quad \text{and} \quad \begin{array}{c} u\colon A^g \\ | \, N \\ B^g \end{array}$$

with $x \notin \mathrm{FV}(B^g)$. Now use $\vdash (\neg\neg B^g \to B^g) \to \tilde{\exists}_x A^g \to \forall_x (A^g \to B^g) \to B^g$. Its premise $\neg\neg B^g \to B^g$ is derivable by Lemma 1.2.7.

(b) First note that $\vdash_c (B \leftrightarrow B^g)$ for B without $\vee, \exists$ (induction on B). From $\Gamma^g \vdash A^g$ we obtain $\Gamma \vdash_c A$ as follows. We argue informally. Assume Γ. Then Γ^g by the note, hence A^g because of $\Gamma^g \vdash A^g$, hence A again by the note. □

1.3. The Curry-Howard correspondence

Clearly the tree structure of logical derivations of any complexity at all can be quite cumbersome, and the availability of some alternative representation therefore becomes increasingly important, especially when we wish to operate on derivations. The Curry-Howard correspondence provides a neat, computationally inspired alternative. The underlying idea is that if we have a derivation $M(x)$ of $A(x)$ then any means of (universally) binding the x should then represent a derivation of $\forall_x A(x)$. The notation chosen for binding the x is $\lambda_x M(x)$, denoting the function $x \mapsto M(x)$. On the side of $\to$ a derivation M of B from some assumptions A, each of which must now in addition have a label u, is then represented as $\lambda_u M(u)$, denoting the function $u \mapsto M(u)$. This requires the labelling of assumptions so that all assumptions discharged by an application of $\to^+$ must have the same label.

More precisely, we represent natural deduction derivations as typed "derivation terms", where the derived formula is the "type" of the term (and displayed as a superscript). This representation goes under the name of *Curry-Howard correspondence*. It dates back to Curry (1930) and somewhat later Howard, published only in (1980), who noted that the types of the combinators used in combinatory logic are exactly the Hilbert style axioms for minimal propositional logic. Subsequently Martin-Löf (1984) transferred these ideas to a natural deduction setting where natural deduction proofs of formulas A now correspond exactly to lambda terms with type A. This representation of natural deduction proofs will henceforth be used consistently.

We give an inductive definition of such derivation terms for the $\to, \forall$-rules in Table 1 where for clarity we have written the corresponding derivations to the left. One can also define derivation terms covering the rules for $\vee$, $\wedge$ and $\exists$, but we shall not do so here.

To see the usefulness of derivation terms consider the problem of eliminating "detours" in logical derivations. Such a detour occurs if the main premise of an elimination rule (for $\to$ or $\forall$) is derived by an introduction

Derivation	Term
$u\colon A$	u^A
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^+ u$	$(\lambda_{u^A} M^B)^{A\to B}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B} \to^-$	$(M^{A\to B} N^A)^B$
$\dfrac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A} \forall^+ x$ (with var.cond.)	$(\lambda_x M^A)^{\forall_x A}$ (with var.cond.)
$\dfrac{\begin{array}{cc}\mid M & \\ \forall_x A(x) & t\end{array}}{A(t)} \forall^-$	$(M^{\forall_x A(x)} t)^{A(t)}$

TABLE 1. Derivation terms for $\to$ and $\forall$

rule. One can then eliminate this detour be a "conversion". We write them in tree notation and also as derivation terms.

$\to$*-conversion.*

$$\dfrac{\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^+ u \qquad \begin{array}{c}\mid N\\ A\end{array}}{B} \to^- \quad \mapsto_\beta \quad \begin{array}{c}\mid N\\ A\\ \mid M\\ B\end{array}$$

or written as derivation terms

$$(\lambda_u M(u^A)^B)^{A\to B} N^A \mapsto_\beta M(N^A)^B.$$

The reader familiar with λ-calculus should note that this is nothing other than β-conversion.

$\forall$-*conversion.*

$$\dfrac{\dfrac{\begin{array}{c}\\ A(x)\end{array}\;|\,M(x)}{\forall_x A(x)}\,\forall^+x \qquad t}{A(t)}\,\forall^- \quad \mapsto_\beta \quad \begin{array}{c}|\,M(t)\\ A(t)\end{array}$$

or written as derivation terms

$$(\lambda_x M(x)^{A(x)})^{\forall_x A(x)} t \mapsto_\beta M(t)^{A(t)}.$$

Every derivation term carries a formula as its type. However, we shall usually leave these formulas implicit and write derivation terms without them. The two β-conversions above then appear as

$$(\lambda_u M(u))N \mapsto_\beta M(N),$$
$$(\lambda_x M(x))t \;\mapsto_\beta M(t).$$

Remark (Normalization). One can show that every reduction sequence given by internal β-conversions terminates after finitely many steps, and that the resulting "normal form" is uniquely determined. For time reasons we refer to the literature for proofs of these facts.

CHAPTER 2

Computability

At this point we leave the general setting of logic and aim to get closer to mathematics. We introduce free algebras (for example, the natural numbers) as basic data structures and consider function spaces based on them. The functional objects are viewed as limits of their finite approximations. We call a functional computable if it is the limit of a recursively enumerable set of finite approximations. To work with such objects in a formal theory, we need to have a language to denote them. Again lambda calculus is the appropriate tool, this time extended by constants for particular functionals defined by equations.

It is a fundamental property of computation that evaluation must be finite. So in any evaluation of $\Phi(\varphi)$ the argument φ can be called upon only finitely many times, and hence the value – if defined – must be determined by some finite subfunction of φ. This is the principle of finite support.

Let us carry this discussion somewhat further and look at the situation one type higher up. Let $\mathcal{H}$ be a partial functional of type-3, mapping type-2 functionals Φ to natural numbers. Suppose Φ is given and $\mathcal{H}(\Phi)$ evaluates to a defined value. Again, evaluation must be finite. Hence the argument Φ can only be called on finitely many functions φ. Furthermore each such φ must be presented to Φ in a finite form (explicitly say, as a set of ordered pairs). In other words, $\mathcal{H}$ and also any type-2 argument Φ supplied to it must satisfy the finite support principle, and this must continue to apply as we move up through the types.

To describe this principle more precisely, we need to introduce the notion of a "finite approximation" Φ_0 of a functional Φ. By this we mean a finite set X of pairs (φ_0, n) such that (i) φ_0 is a finite function, (ii) $\Phi(\varphi_0)$ is defined with value n, and (iii) if (φ_0, n) and (φ_0', n') belong to X where φ_0 and φ_0' are "consistent", then $n = n'$. The essential idea here is that Φ should be viewed as the union of all its finite approximations. Using this notion of a finite approximation we can now formulate the

> *Principle of finite support.* If $\mathcal{H}(\Phi)$ is defined with value n, then there is a finite approximation Φ_0 of Φ such that $\mathcal{H}(\Phi_0)$ is defined with value n.

The monotonicity principle formalizes the simple idea that once $\mathcal{H}(\Phi)$ is evaluated, then the same value will be obtained no matter how the argument Φ is extended. This requires the notion of "extension". Φ' extends Φ if for any piece of data (φ_0, n) in Φ there exists another (φ_0', n) in Φ' such that φ_0 extends φ_0' (note the contravariance!). The second basic principle is then

> *Monotonicity principle*. If $\mathcal{H}(\Phi)$ is defined with value n and Φ' extends Φ, then $\mathcal{H}(\Phi')$ is defined with value n.

An immediate consequence of finite support and monotonicity is that the behaviour of any functional is indeed determined by its set of finite approximations. For if Φ, Φ' have the same finite approximations and $\mathcal{H}(\Phi)$ is defined with value n, then by finite support, $\mathcal{H}(\Phi_0)$ is defined with value n for some finite approximation Φ_0 of Φ, and then by monotonicity $\mathcal{H}(\Phi')$ is defined with value n. Thus $\mathcal{H}(\Phi) = \mathcal{H}(\Phi')$, for all $\mathcal{H}$.

This observation now allows us to formulate a notion of abstract computability:

> *Effectivity principle*. An object is computable just in case its set of finite approximations is (primitive) recursively enumerable (or equivalently, Σ^0_1-definable).

The general theory of computability concerns partial functions and partial operations on them. However, we might be interested in particular objects only, so once the theory of general (partial) objects is developed, we can look for ways to restrict attention to the particular ones. Examples for interesting sets of objects are (i) the total functions on natural numbers, (ii) so-called cototal objects like "streams" of signed digits $\{-1, 0, 1\}$ (useful to represent real numbers), and (iii) extensional functionals mapping functions on natural numbers to natural numbers.

2.1. Abstract computability via information systems

We need to define appropriate domains for our to-be-defined computable functionals, viewed as limits of their finite approximations. Information systems are a convenient setting to introduce and study the latter.

2.1.1. Information systems. The basic idea of information systems is to provide an axiomatic setting to describe approximations of abstract objects (like functions or functionals) by concrete, finite ones. We do not attempt to analyze the notion of "concreteness" or finiteness here, but rather 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 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

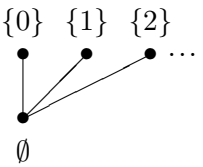
$$
\begin{aligned}
&U \vdash V \quad \text{for} \quad U \in \mathrm{Con} \wedge \forall_{a \in V}(U \vdash a),\\
&a \uparrow b \quad \text{for} \quad \{a, b\} \in \mathrm{Con} \qquad (a, b \text{ are } \textit{consistent}),\\
&U \uparrow V \quad \text{for} \quad \forall_{a \in U, b \in V}(a \uparrow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

DEFINITION. The *ideals* (also called *objects*) 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 ideal. The set of all ideals of $\boldsymbol{A}$ is denoted by $|\boldsymbol{A}|$.

EXAMPLES. 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 ideals are just the elements of Con. For $A = \mathbb{N}$ we have the following picture of the Con-sets.



A rather important example is the following, which concerns approximations of functions from a countable set A into a countable set B. The

tokens are the pairs (a, b) with $a \in A$ and $b \in B$, and

$$\mathrm{Con} := \{\, \{\, (a_i, b_i) \mid i < k \,\} \mid \forall_{i,j<k}(a_i = a_j \to b_i = b_j) \,\},$$
$$U \vdash (a, b) := (a, b) \in U.$$

It is easy to verify that this defines an information system whose ideals are (the graphs of) all partial functions from A to B.

Remark. One can show that for an arbitrary information system $\boldsymbol{A} = (A, \mathrm{Con}, \vdash)$ the structure $(|\boldsymbol{A}|, \subseteq, \overline{\emptyset})$ is a "domain" (also called Scott-Ershov domain, or "bounded complete algebraic cpo"), whose set of "compact elements" can be represented as $|\boldsymbol{A}|_c = \{\, \overline{U} \mid U \in \mathrm{Con} \,\}$. The converse holds as well: every countable domain can be represented as an information system. We will not need this relation to standard (non-constructive) domain theory, and hence not even define these notions here.

2.1.2. Function spaces. We define the "function space" $\boldsymbol{A} \to \boldsymbol{B}$ between two information systems $\boldsymbol{A}$ and $\boldsymbol{B}$.

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

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

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 2.1.1. *If $\boldsymbol{A}$ and $\boldsymbol{B}$ are information systems, then so is $\boldsymbol{A} \to \boldsymbol{B}$ defined as above.*

PROOF. Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$. The first, second and fourth property of the definition are clearly satisfied. For the third, suppose

$$\{(U_1, b_1), \ldots, (U_n, b_n)\} \vdash (U, b), \quad \text{i.e.,} \quad \{\, b_j \mid U \vdash_A U_j \,\} \vdash_B b.$$

We have to show that $\{(U_1, b_1), \ldots, (U_n, b_n), (U, b)\} \in \mathrm{Con}$. So let $I \subseteq \{1, \ldots, n\}$ and suppose

$$U \cup \bigcup_{i \in I} U_i \in \mathrm{Con}_A.$$

We must show that $\{b\} \cup \{\, b_i \mid i \in I \,\} \in \mathrm{Con}_B$. Let $J \subseteq \{1, \ldots, n\}$ consist of those j with $U \vdash_A U_j$. Then also

$$U \cup \bigcup_{i \in I} U_i \cup \bigcup_{j \in J} U_j \in \mathrm{Con}_A.$$

Since

$$\bigcup_{i \in I} U_i \cup \bigcup_{j \in J} U_j \in \mathrm{Con}_A,$$

from the consistency of $\{(U_1, b_1), \ldots, (U_n, b_n)\}$ we can conclude that

$$\{\, b_i \mid i \in I \,\} \cup \{\, b_j \mid j \in J \,\} \in \mathrm{Con}_B.$$

But $\{\, b_j \mid j \in J \,\} \vdash_B b$ by assumption. Hence

$$\{\, b_i \mid i \in I \,\} \cup \{\, b_j \mid j \in J \,\} \cup \{b\} \in \mathrm{Con}_B.$$

For the final property, suppose

$$W \vdash W' \quad \text{and} \quad W' \vdash (U, b).$$

We have to show $W \vdash (U, b)$, i.e., $WU \vdash_B b$. We obtain $WU \vdash_B W'U$ by monotonicity in the first argument, and $W'U \vdash_B b$ by definition. □

We shall now give an alternative characterization of the ideals in $\boldsymbol{A} \to \boldsymbol{B}$, as "approximable maps". The basic idea for approximable maps is the desire to study "information respecting" maps from $\boldsymbol{A}$ into $\boldsymbol{B}$. Such a map is given by a relation r between Con_A and B, where $(U, b) \in r$ intuitively means that whenever we are given the information $U \in \mathrm{Con}_A$, then we know that at least the token b appears in the value.

DEFINITION. Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. A relation $r \subseteq \mathrm{Con}_A \times B$ is an *approximable map* if it satisfies the following:

(a) if $(U, b_1), \ldots, (U, b_n) \in r$, then $\{b_1, \ldots, b_n\} \in \mathrm{Con}_B$;
(b) if $(U, b_1), \ldots, (U, b_n) \in r$ and $\{b_1, \ldots, b_n\} \vdash_B b$, then $(U, b) \in r$;
(c) if $(U', b) \in r$ and $U \vdash_A U'$, then $(U, b) \in r$.

THEOREM 2.1.2. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be information systems. Then the ideals of $\boldsymbol{A} \to \boldsymbol{B}$ are exactly the approximable maps from $\boldsymbol{A}$ to $\boldsymbol{B}$.*

PROOF. Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$. If $r \in |\boldsymbol{A} \to \boldsymbol{B}|$ then $r \subseteq \mathrm{Con}_A \times B$ is consistent and deductively closed. We have to show that r satisfies the axioms for approximable maps.

(a) Let $(U, b_1), \ldots, (U, b_n) \in r$. We must show that $\{b_1, \ldots, b_n\} \in \mathrm{Con}_B$. But this clearly follows from the consistency of r.

(b) Let $(U, b_1), \ldots, (U, b_n) \in r$ and $\{b_1, \ldots, b_n\} \vdash_B b$. We must show that $(U, b) \in r$. But

$$\{(U, b_1), \ldots, (U, b_n)\} \vdash (U, b)$$

by the definition of the entailment relation $\vdash$ in $\boldsymbol{A} \to \boldsymbol{B}$, hence $(U, b) \in r$ since r is deductively closed.

(c) Let $U \vdash_A U'$ and $(U', b) \in r$. We must show that $(U, b) \in r$. But

$$\{(U', b)\} \vdash (U, b)$$

since $\{(U', b)\}U = \{b\}$ (which follows from $U \vdash_A U'$), hence $(U, b) \in r$, again since r is deductively closed.

For the other direction suppose that $r \subseteq \mathrm{Con}_A \times B$ is an approximable map. We must show that $r \in |\boldsymbol{A} \to \boldsymbol{B}|$.

Consistency of r. Suppose $(U_1, b_1), \ldots, (U_n, b_n) \in r$ and $U = \bigcup\{ U_i \mid i \in I \} \in \mathrm{Con}_A$ for some $I \subseteq \{1, \ldots, n\}$. We must show that $\{ b_i \mid i \in I \} \in \mathrm{Con}_B$. Now from $(U_i, b_i) \in r$ and $U \vdash_A U_i$ we obtain $(U, b_i) \in r$ by axiom (c) for all $i \in I$, and hence $\{ b_i \mid i \in I \} \in \mathrm{Con}_B$ by axiom (a).

Deductive closure of r. Suppose $(U_1, b_1), \ldots, (U_n, b_n) \in r$ and

$$W := \{(U_1, b_1), \ldots, (U_n, b_n)\} \vdash (U, b).$$

We must show $(U, b) \in r$. By definition of $\vdash$ for $\boldsymbol{A} \to \boldsymbol{B}$ we have $WU \vdash_B b$, which is $\{ b_i \mid U \vdash_A U_i \} \vdash_B b$. Further by our assumption $(U_i, b_i) \in r$ we know $(U, b_i) \in r$ by axiom (c) for all i with $U \vdash_A U_i$. Hence $(U, b) \in r$ by axiom (b). □

2.1.3. Continuous functions. We can also characterize approximable maps in a different way, which is closer to usual characterizations of continuity[1]:

LEMMA 2.1.3. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be information systems and $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ monotone (i.e., $x \subseteq y$ implies $f(x) \subseteq f(y)$). Then the following are equivalent.*

[1]In fact, approximable maps are exactly the continuous functions w.r.t. the so-called Scott topology. However, we will not enter this subject here.

(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 ideal in $|\boldsymbol{B}|$. Note also that from PFS and monotonicity of f it follows immediately that if $V \subseteq f(x)$, then $V \subseteq f(\overline{U})$ for some $U \subseteq x$.

Proof. Let f satisfy PFS, and $D \subseteq |\boldsymbol{A}|$ be directed. $f(\bigcup_{x\in D} x) \supseteq \bigcup_{x\in D} f(x)$ follows from monotonicity. For the reverse inclusion let $b \in f(\bigcup_{x\in D} x)$. Then by PFS $b \in f(\overline{U})$ for some $U \subseteq \bigcup_{x\in D} x$. From the directedness and the fact that U is finite we obtain $U \subseteq z$ for some $z \in D$. From $b \in f(\overline{U})$ and monotonicity infer $b \in f(z)$. Conversely, let f commute with directed unions, and assume $b \in f(x)$. Then

$$b \in f(x) = f(\bigcup_{U\subseteq x} \overline{U}) = \bigcup_{U\subseteq x} f(\overline{U}),$$

hence $b \in f(\overline{U})$ for some $U \subseteq x$. □

We call a function $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ continuous if it satifies the conditions in Lemma 2.1.3. 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 2.1.4. *Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Then the ideals 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 approximable map $r \subseteq \mathrm{Con}_A \times B$ we can associate a continuous function $|r|\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ by*

$$|r|(z) := \{\, b \in B \mid (U, b) \in r \text{ for some } U \subseteq z \,\}.$$

We call $|r|(z)$ the application *of r to z.*

(b) *Conversely, with any continuous function* $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ *we can associate an approximable map* $\hat{f} \subseteq \mathrm{Con}_A \times B$ *by*

$$\hat{f} := \{\, (U, b) \mid b \in f(\overline{U}) \,\}.$$

These assignments are inverse to each other, i.e., $f = |\hat{f}|$ *and* $r = \widehat{|r|}$.

Proof. Let r be an ideal of $\boldsymbol{A} \to \boldsymbol{B}$; then by Theorem 2.1.2 we know that r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\boldsymbol{A}|$.

$|r|(z)$ is consistent: let $b_1, \dots, b_n \in |r|(z)$. Then there are $U_1, \dots, U_n \subseteq z$ such that $(U_i, b_i) \in r$. Hence $U := U_1 \cup \dots \cup U_n \subseteq z$ and $(U, b_i) \in r$ by axiom (c) of approximable maps. Now from axiom (a) we can conclude that $\{b_1, \dots, b_n\} \in \mathrm{Con}_B$.

$|r|(z)$ is deductively closed: let $b_1, \dots, b_n \in |r|(z)$ and $\{b_1, \dots, b_n\} \vdash_B b$. We must show $b \in |r|(z)$. As before we find $U \subseteq z$ such that $(U, b_i) \in r$. Now from axiom (b) we can conclude $(U, b) \in r$ and hence $b \in |r|(z)$.

Continuity of $|r|$ follows immediately from part (a) of Lemma 2.1.3 above, since by definition $|r|$ is monotone and satisfies PFS.

Now let $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ be continuous. It is easy to verify that $\hat{f}$ is indeed an approximable map. Furthermore

$$\begin{aligned} b \in |\hat{f}|(z) &\leftrightarrow (U, b) \in \hat{f} && \text{for some } U \subseteq z \\ &\leftrightarrow b \in f(\overline{U}) && \text{for some } U \subseteq z \\ &\leftrightarrow b \in f(z) && \text{by monotonicity and PFS.} \end{aligned}$$

Finally, for any approximable map $r \subseteq \mathrm{Con}_A \times B$ we have

$$\begin{aligned} (U, b) \in r &\leftrightarrow \exists_{V \subseteq \overline{U}} (V, b) \in r && \text{by axiom (c) for approximable maps} \\ &\leftrightarrow b \in |r|(\overline{U}) \\ &\leftrightarrow (U, b) \in \widehat{|r|}, \end{aligned}$$

hence $r = \widehat{|r|}$. □

Consequently we can (and will) view approximable maps $r \subseteq \mathrm{Con}_A \times B$ as continuous functions from $|\boldsymbol{A}|$ to $|\boldsymbol{B}|$.

Equality of two subsets $r, s \subseteq \mathrm{Con}_A \times B$ means that they consist of the same tokens (U, b). We can characterize equality $r = s$ by extensional equality of the associated functions $|r|, |s|$. It even suffices that $|r|$ and $|s|$ coincide on all compact elements $\overline{U}$ for $U \in \mathrm{Con}_A$.

Lemma 2.1.5 (Extensionality). *Assume that* $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ *and* $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ *are information systems and* $r, s \subseteq \mathrm{Con}_A \times B$ *approximable maps. Then the following are equivalent.*

(a) $r = s$,
(b) $|r|(z) = |s|(z)$ *for all* $z \in |\boldsymbol{A}|$,
(c) $|r|(\overline{U}) = |s|(\overline{U})$ *for all* $U \in \mathrm{Con}_A$.

PROOF. It suffices to prove (c) $\to$ (a). As above this follows from

$$\begin{aligned}(U,b) \in r &\leftrightarrow \exists_{V \subseteq \overline{U}}(V,b) \in r \quad \text{by axiom (c) for approximable maps} \\ &\leftrightarrow b \in |r|(\overline{U}).\end{aligned}$$

□

Moreover, one can easily check that

$$s \circ r := \{\, (U,c) \mid \exists_V((V,c) \in s \wedge (U,V) \subseteq r) \,\}$$

is an approximable map (where $(U,V) := \{\, (U,b) \mid b \in V \,\}$), and

$$|s \circ r| = |s| \circ |r|, \quad \widehat{g \circ f} = \hat{g} \circ \hat{f}.$$

We usually write $r(z)$ for $|r|(z)$, and similarly $(U,b) \in f$ for $(U,b) \in \hat{f}$. It should always be clear from the context where the mods and hats should be inserted.

2.1.4. 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. Let α, β, ξ denote type variables.

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 *non-recursive* (or *explicit*) if it does not have recursive argument types.

EXAMPLES. We list some algebra forms without parameters, with standard names for the constructors added to each constructor type.

$$\begin{aligned}
&\mathbb{U} := \mu_\xi(\mathrm{Dummy}\colon \xi) && \text{(unit)},\\
&\mathbb{B} := \mu_\xi(\mathtt{tt}\colon \xi, \mathtt{ff}\colon \xi) && \text{(booleans)},\\
&\mathbb{N} := \mu_\xi(0\colon \xi, S\colon \xi \to \xi) && \text{(natural numbers, unary)},\\
&\mathbb{P} := \mu_\xi(1\colon \xi, S_0\colon \xi \to \xi, S_1\colon \xi \to \xi) && \text{(positive numbers, binary)},\\
&\mathbb{Y} := \mu_\xi(-\colon \xi, \mathrm{Branch}\colon \xi \to \xi \to \xi) && \text{(binary trees, or derivations)}.
\end{aligned}$$

Algebra forms with type parameters are

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

The default name for the i-th constructor of an algebra form is 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

$$\begin{aligned}
\mathrm{lev}(\alpha) &:= 0,\\
\mathrm{lev}(\iota(\vec{\rho})) &:= \max(\mathrm{lev}(\vec{\rho})),\\
\mathrm{lev}(\tau \to \sigma) &:= \max(\mathrm{lev}(\sigma), 1 + \mathrm{lev}(\tau)).
\end{aligned}$$

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

EXAMPLES. 1. $\mathbb{L}(\alpha)$, $\mathbb{L}(\mathbb{L}(\alpha))$, $\alpha \times \beta$ are algebras.
2. $\mathbb{L}(\mathbb{L}(\mathbb{N}))$, $\mathbb{N} + \mathbb{B}$, $\mathbb{Z} := \mathbb{P} + \mathbb{U} + \mathbb{P}$, $\mathbb{Q} := \mathbb{Z} \times \mathbb{P}$ are closed base types.
3. $\mathbb{R} := (\mathbb{N} \to \mathbb{Q}) \times (\mathbb{P} \to \mathbb{N})$ is a closed algebra of level 1.

There can be many equivalent ways to define a particular type. For instance, we could take $\mathbb{U} + \mathbb{U}$ to be the type of booleans, $\mathbb{L}(\mathbb{U})$ to be the type of natural numbers, and $\mathbb{L}(\mathbb{B})$ to be the type of positive binary numbers.

2.1.5. Partial continuous functionals. 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* $\iota(\rho)$. For simplicity assume that there is only one parameter type ρ.

(a) *Tokens* $a \in C_{\iota(\rho)}$ are the type correct constructor expressions $\mathrm{C}Va_1^* \dots a_n^*$ 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_ρ.

(b) A finite set U of tokens in $C_{\iota(\rho)}$ is *consistent* (i.e., $\in \mathrm{Con}_{\iota(\rho)}$) if all its elements start with the same constructor C, say of arity $\rho \to \iota(\rho) \dots \to \iota(\rho) \to \iota(\rho)$. Let $U = \{\mathrm{C}V_1 a_{11}^* \dots a_{1n}^*, \dots, \mathrm{C}V_m a_{m1}^* \dots a_{mn}^*\}$. Then we require that (i) $V_1 \cup \dots \cup V_m$ is consistent (i.e., $\in \mathrm{Con}_\rho$) 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., $\in \mathrm{Con}_{\iota(\rho)}$).

(c) $\{\mathrm{C}V_1 a_{11}^* \dots a_{1n}^*, \dots, \mathrm{C}V_m a_{m1}^* \dots a_{mn}^*\} \vdash_{\iota(\rho)} \mathrm{C}Va_1^* \dots a_n^*$ if and only if (i) $V_1 \cup \dots \cup V_m \vdash_\rho V$ and (ii) for each set U_i as in (b) above we have $U_i \vdash_{\iota(\rho)} a_i^*$ (where $U_i \vdash *$ is taken to be true).

Case $\tau \to \sigma$. Tokens, consistency and entailment for $\boldsymbol{C}_{\tau\to\sigma}$ are defined as done generally in Section 2.1.2 (on function spaces). In more detail:

(a) Tokens in $\boldsymbol{C}_{\tau\to\sigma}$ are pairs (U, b) with $U \in \mathrm{Con}_\tau$ and $b \in C_\sigma$.

(b) $\{\,(U_i, b_i) \mid i \in I\,\} \in \mathrm{Con}_{\tau\to\sigma}$ is defined to mean

$$\forall_{J\subseteq I}\Big(\bigcup_{j\in J} U_j \in \mathrm{Con}_\tau \to \{\,b_j \mid j \in J\,\} \in \mathrm{Con}_\sigma\Big).$$

(c) $W \vdash_{\tau\to\sigma} (U, b)$ is defined to mean $WU \vdash_\sigma b$.

LEMMA 2.1.6. $(C_\tau, \mathrm{Con}_\tau, \vdash_\tau)$ *is an information system.*

PROOF. *Case* $\iota(\vec{\rho})$. For every constructor, for instance $\mathrm{C}\colon \rho \to \iota(\rho) \to \iota(\rho)$, we need to prove the axioms of information systems. We only give details for the last one, transitivity. By definition we can assume

$$\{\mathrm{C}U_1 a_1^*, \dots \mathrm{C}U_n a_n^*\} \vdash_{\iota(\rho)} \mathrm{C}V_j b_j^* \quad \text{and} \quad \{\mathrm{C}V_1 b_1^*, \dots, \mathrm{C}V_m b_m^*\} \vdash_{\iota(\rho)} \mathrm{C}Wc^*$$

By definition we have

$$\begin{array}{ll} U_1 \cup \dots \cup U_n \vdash_\rho V_j & \\ \{a_1^*, \dots, a_n^*\} \vdash_{\iota(\rho)} b_j^* & \text{and} \\ & V_1 \cup \dots \cup V_m \vdash_\rho W \\ & \{b_1^*, \dots, b_m^*\} \vdash_{\iota(\rho)} c^* \end{array}$$

By the (main and side) induction hypotheses we obtain

$$\begin{array}{l} U_1 \cup \dots \cup U_n \vdash_\rho W \\ \{a_1^*, \dots, a_n^*\} \vdash_{\iota(\rho)} c^* \end{array}$$

Hence the goal $\{\mathrm{C}U_1 a_1^*, \dots \mathrm{C}U_n a_n^*\} \vdash_{\iota(\rho)} \mathrm{C}Wc^*$ follows by definition.

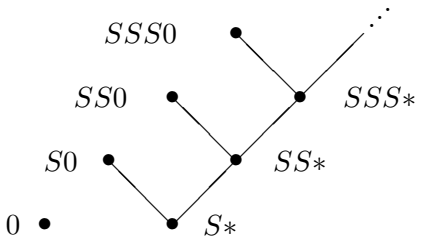


FIGURE 1. Tokens and entailment for $\mathbb{N}$

Case $\tau \to \sigma$. As in Section 2.1.2. □

Observe that all the notions involved are computable: $a \in C_\tau$, $U \in \mathrm{Con}_\tau$ and $U \vdash_\tau a$.

DEFINITION (Partial continuous functionals). The ideals $x \in |\boldsymbol{C}_\tau|$ 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.

DEFINITION (Cototal and total ideals of closed base type). Let $\iota(\vec{\rho})$ be a closed base type. Its tokens can be seen as constructor trees with some recursive argument positions occupied by $*$. An ideal x in $\boldsymbol{C}_\rho$ is *cototal* if for each of its tokens $P(*)$ with a distinguished occurrence of $*$ there is another token of the form $P(\mathrm{C}\vec{\emptyset}\vec{*})$ in x. We call x *total* if it is cototal and finite.

2.1.6. Examples. The tokens for the algebra $\mathbb{N}$ are shown in Figure 1. For tokens a, b we have $\{a\} \vdash b$ if and only if there is a path from a (up) to b (down).

Dyadic rational numbers in the interval $(-1, 1)$ are those of the form

$$\sum_{n<m} \frac{k_n}{2^{n+1}} \qquad \text{with } k_n \in \{-1, 1\}.$$

A pictorial representation is in Figure 2. Irrational real numbers like $\frac{1}{2}\sqrt{2}$ then can be seen as infinite paths (or "streams"). Both of these objects appear in our present setting as total or cototal ideals of certain closed base types. This connection will make it possible to extract algorithms on stream-represented real numbers from proofs talking about ordinary reals given by Cauchy sequences with moduli.

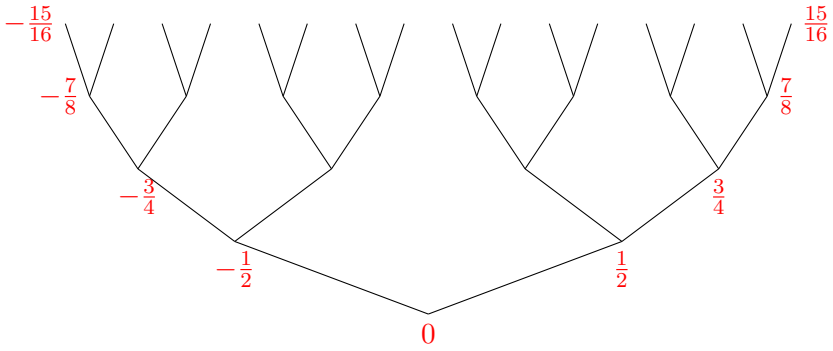
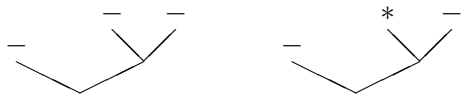


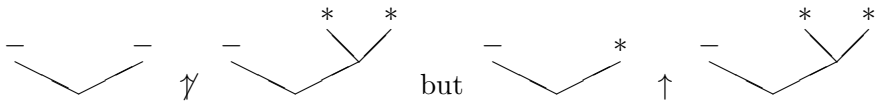
FIGURE 2. Dyadic rationals.

To establish such a connection we first consider the algebra $\mathbb{Y}$ of binary trees. Tokens in $\boldsymbol{C}_{\mathbb{Y}}$:

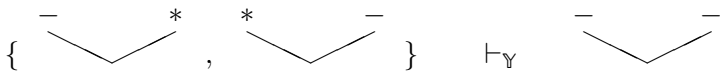


Consistency in $\boldsymbol{C}_{\mathbb{Y}}$:

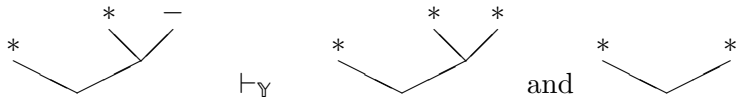
Moreover



Entailment in $\boldsymbol{C}_{\mathbb{Y}}$:

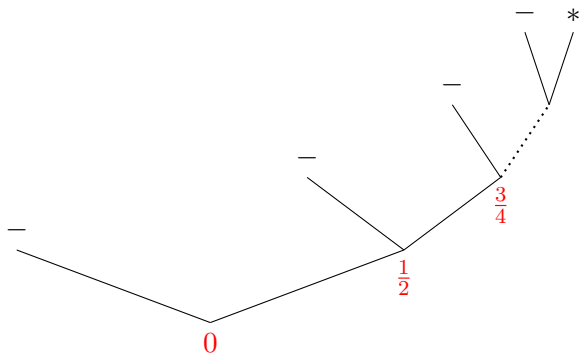


and also

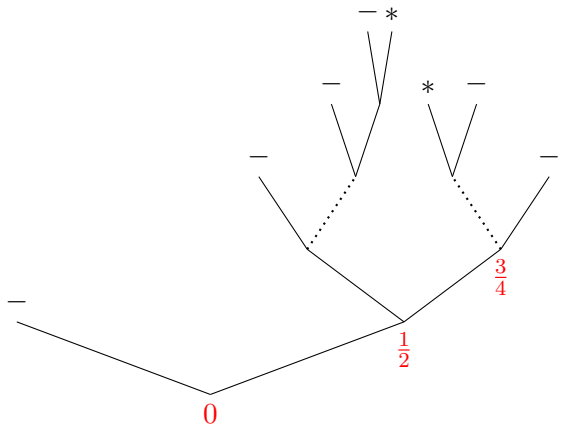


Ideals in $\boldsymbol{C}_{\mathbb{Y}}$:

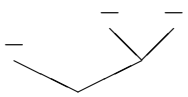
(1) $R :=$ closure of all



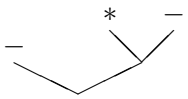
(2) L is defined similarly
(3) $L \cup R$
(4) $\frac{1}{2} :=$ closure of all



(5) Closure of



(6) Closure of



Among these are

$(1)-(4)$	cototal ideals,
(5)	a total ideal,
(6)	a general ideal.

Next we consider the algebra $\mathbb{L}(\mathbb{B})$ of lists of booleans, to represent finite or infinite paths. Tokens in $\boldsymbol{C}_{\mathbb{L}(\mathbb{B})}$ are for instance

$$\{\mathtt{ff}\} :: \{\mathtt{ff}\} :: \{\mathtt{tt}\} :: \mid \qquad \text{abbreviated} \qquad RRL \mid$$

Consistency in $\boldsymbol{C}_{\mathbb{L}(\mathbb{B})}$:

$$RL \mid \not\vdash RRL*, \qquad RR* \uparrow RRL*$$

Entailment in $\boldsymbol{C}_{\mathbb{L}(\mathbb{B})}$:

$$RRL* \vdash RR*, R* \qquad \text{(and also e.g. } RR\emptyset*),$$
$$RRL \mid \vdash RRL*, RR*, R*$$

Ideals in $\boldsymbol{C}_{\mathbb{L}(\mathbb{B})}$:

total:	Closure of $RLRR \mid$
cototal:	Closure of all $RLRR \ldots R*$ and all $RRLL \ldots L*$
general:	Closure of $RLRR*$

2.1.7. Bisimilarity. For closed ground types equality of cototal ideals 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 $\approx_{\mathbb{Y}}^{-}$:

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

with 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 $\approx_{\mathbb{Y}}^{+}$:

$$\forall_{x,x'}(Xxx' \to (x \equiv - \wedge x' \equiv -) \vee \exists_{x_1,x_2,x_1',x_2'}((x_1 \approx x_1' \vee Xx_1x_1') \wedge (x_2 \approx x_2' \vee Xx_2x_2') \wedge x \equiv \mathrm{C}x_1x_2 \wedge x' \equiv \mathrm{C}x_1'x_2')) \to$$
$$X \subseteq \approx.$$

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

LEMMA 2.1.7 (Bisimilarity). *$x \approx_{\mathbb{Y}} x'$ implies $x \equiv x'$.*

PROOF. Let a range over tokens for $\mathbb{Y}$, and define the *height* $|a^*|$ of an extended token a^* by $|*| := 0$, $|-| := 1$, $|\mathrm{C}a_1^*a_2^*| := 1 + \max(|a_1^*|, |a_2^*|)$. By induction on the height $|a^*|$ of extended tokens a^* we prove that for all ideals x, x' and extended tokens $a^* \in x$ we have $a^* \in x'$. It suffices to consider the case $\mathrm{C}a_1^*a_2^*$. From $x \approx_{\mathbb{Y}} x'$ we obtain by the closure axiom x_1, x_2, x_1', x_2' with

$$x_1 \approx x_1' \wedge x_2 \approx x_2' \wedge x \equiv \mathrm{C}x_1x_2 \wedge x' \equiv \mathrm{C}x_1'x_2'.$$

Then $a_i^* \in x_i$ (for $i = 1, 2$), and by IH $a_i^* \in x_i'$. Thus $\mathrm{C}a_1^*a_2^* \in x'$. □

From the Bisimilarity Lemma 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 ${}^{\mathrm{co}}T_{\mathbb{Y}}^-$:

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

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

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

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

LEMMA 2.1.8 (Characterization of $\approx_{\mathbb{Y}}$).

$$x \approx_{\mathbb{Y}} x' \leftrightarrow x, x' \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x \equiv x'.$$

PROOF. "$\to$". By Lemma 2.1.7 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

$$(x \equiv -) \vee \exists_{x_1,x_2}((x_1 \in {}^{\mathrm{co}}T \vee \exists_{x_1'}(x_1 \approx x_1')) \wedge (x_2 \in {}^{\mathrm{co}}T \vee \exists_{x_2'}(x_2 \approx x_2')) \wedge x \equiv \mathrm{C}x_1x_2).$$

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

$$(x \equiv - \wedge x' \equiv -) \vee \exists_{x_1,x_2,x_1',x_2'}(x_1 \approx x_1' \wedge x_2 \approx x_2' \wedge x \equiv \mathrm{C}x_1x_2 \wedge x' \equiv \mathrm{C}x_1'x_2')).$$

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 $x \equiv \mathrm{C}x_1x_2$, 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 $X := \{\, x, x' \mid x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x \equiv x' \,\}$. It suffices to prove the premise. Fix x, x' with $x \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x \equiv x'$. The goal is

$$(x \equiv - \wedge x' \equiv -) \vee \exists_{x_1,x_2,x_1',x_2'}((x_1 \approx x_1' \vee (x_1 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x_1 \equiv x_1')) \wedge (x_2 \approx x_2' \vee (x_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x_2 \equiv x_2')) \wedge x \equiv \mathrm{C}x_1x_2 \wedge x' \equiv \mathrm{C}x_1'x_2')).$$

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

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

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 $x \equiv \mathrm{C}x_1x_2$. Then we are done as well with $x_1' := x_1$ and $x_2' := x_2$, since again $x \equiv x'$. □

2.1.8. Constructors as continuous functions. Let ι be an algebra. Every constructor C generates the following ideal in the function space determined by the type of the constructor:

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

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

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

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

(For $\mathbb{N}$ we have $|r_S|(\{0\}) = \{S0, S*\}$ and $|r_S|(\{S0, S*\}) = \{SS0, SS*, S*\}$.) 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 2.1.9 (Constructors are injective and have disjoint ranges). *Let ι be an algebra and* C *be a constructor of ι. Then*

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

If $\mathrm{C}_1, \mathrm{C}_2$ *are distinct constructors of ι, then $|r_{\mathrm{C}_1}|(\vec{x}\,) \neq |r_{\mathrm{C}_2}|(\vec{y}\,)$, since the two ideals 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 ideal, then the value is as well). But then we have

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

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

2.2. A term language for computable functionals

To work with computable functionals in a formal theory we need to have a language to denote them. Again lambda calculus is the appropriate tool, this time extended by constants for computable functionals.

Recall that a partial continuous functional is defined to be computable if it is the limit of a recursively enumerable set of finite approximations. We introduce a convenient way to define computable functionals, by means of defining equations or more precisely, computation rules. Therefore we extend the term language by constants D defined by certain "computation rules". The resulting term system can be seen as a common extension of Gödel's T (1958) and Plotkin's PCF (1977); we call it T^+.

2.2.1. A common extension T^+ of Gödel's T and Plotkin's PCF.

Definition (Terms). *Terms* of T^+ are built from (typed) variables and (typed) constants (constructors C or defined constants D; see the definition below) by application and abstraction:

$$M, N ::= x^\tau \mid \mathrm{C}^\tau \mid D^\tau \mid (\lambda_{x^\tau} M^\sigma)^{\tau\to\sigma} \mid (M^{\tau\to\sigma} N^\tau)^\sigma.$$

The set $\mathrm{FV}(M)$ of free variables of a term M is defined by

$$\mathrm{FV}(x) := \{x\}, \qquad \mathrm{FV}(C) := \mathrm{FV}(D) := \emptyset,$$
$$\mathrm{FV}(\lambda_x M) := \mathrm{FV}(M) \setminus \{x\}, \qquad \mathrm{FV}(MN) := \mathrm{FV}(M) \cup \mathrm{FV}(N).$$

Definition (Conversion). We define a conversion relation $\mapsto_\beta$ for terms similarly to what we did for derivation terms:

$$(\lambda_x M(x))^{\tau\to\sigma} N^\tau \mapsto_\beta M(N)^\sigma.$$

In addition we will employ another conversion relation $\mapsto_\eta$ defined by

$$\lambda_x(Mx) \mapsto_\eta M \quad \text{if } x \notin \mathrm{FV}(M) \ (M \text{ not an abstraction}).$$

Definition (Computation rule). Every defined constant D comes with a system of *computation rules*, consisting of finitely many equations

$$D\vec{P}_i(\vec{y}_i) = M_i \qquad (i = 1, \dots, n \text{ where } n \geq 0) \tag{7}$$

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 $D\vec{P}_i(\vec{y}_i) = M_i$ is that the lengths of all $\vec{P}_i(\vec{y}_i)$ are the same; this number is called the *arity* of D, denoted by $\mathrm{ar}(D)$. A substitution instance of a left hand side of (7) is called a *D-redex*.

More formally, constructor patterns are defined inductively by (we write $\vec{P}(\vec{x}\,)$ to indicate all variables in $\vec{P}$):

(a) x is a constructor pattern.
(b) The empty list is a constructor pattern.
(c) If $\vec{P}(\vec{x}\,)$ and $Q(\vec{y}\,)$ are constructor patterns whose variables $\vec{x}$ and $\vec{y}$ are disjoint, then $(\vec{P}, Q)(\vec{x}, \vec{y}\,)$ is a constructor pattern.
(d) If C is a constructor and $\vec{P}$ a constructor pattern, then so is $\mathrm{C}\vec{P}$.

Remark. The requirement of disjoint variables in constructor patterns $\vec{P}_i$ and $\vec{P}_j$ used in computation rules of a defined constant D is needed to ensure that applying the most general unifier produces constructor patterns again. However, for readability we take this as an implicit convention, and write computation rules with possibly non-disjoint variables.

Examples of constants D defined by computation rules are abundant. In particular, the (structural) recursion and corecursion operators will be defined by computation rules.

The simplest example is the constant $\bot_\iota$ of type ι with no computation rules. The boolean connectives andb, impb and orb are defined by

$$\begin{array}{lll} \mathtt{tt}\ \mathrm{andb}\ y = y, & \mathtt{ff}\ \mathrm{impb}\ y = \mathtt{tt}, & \mathtt{tt}\ \mathrm{orb}\ y = \mathtt{tt}, \\ x\ \mathrm{andb}\ \mathtt{tt} = x, & \mathtt{tt}\ \mathrm{impb}\ y = y, & x\ \mathrm{orb}\ \mathtt{tt} = \mathtt{tt}, \\ \mathtt{ff}\ \mathrm{andb}\ y = \mathtt{ff}, & x\ \mathrm{impb}\ \mathtt{tt} = \mathtt{tt}, & \mathtt{ff}\ \mathrm{orb}\ y = y, \\ x\ \mathrm{andb}\ \mathtt{ff} = \mathtt{ff}, & & x\ \mathrm{orb}\ \mathtt{ff} = x. \end{array}$$

Notice that when two such rules overlap, their right hand sides are equal under any unifier of the left hand sides.

Decidable *equality* $=_\iota \colon \iota \to \iota \to \mathbb{B}$ for a closed base type ι can be defined easily by computation rules. For example,

$$\begin{array}{ll} (0 =_{\mathbb{N}} 0) = \mathtt{tt}, & (Sn =_{\mathbb{N}} 0) = \mathtt{ff}, \\ (0 =_{\mathbb{N}} Sm) = \mathtt{ff}, & (Sn =_{\mathbb{N}} Sm) = \ (n =_{\mathbb{N}} m). \end{array}$$

For the algebra $\mathbb{Y}$ of binary trees with constructors 0 (leaf) and C (construct a new tree from two given ones) we have

$$\begin{array}{ll} (0 =_{\mathbb{Y}} 0) = \mathtt{tt}, & (\mathrm{C}ts =_{\mathbb{Y}} 0) = \mathtt{ff}, \\ (0 =_{\mathbb{Y}} \mathrm{C}ts) = \mathtt{ff}, & (\mathrm{C}ts =_{\mathbb{Y}} \mathrm{C}t's') = \ (t =_{\mathbb{Y}} t'\ \mathrm{andb}\ s =_{\mathbb{Y}} s'). \end{array}$$

For the algebra $\mathbb{N}$ of natural numbers we have the doubling function

$$\begin{aligned}\mathrm{Double}(0) &:= 0,\\ \mathrm{Double}(S(n)) &:= S(S(\mathrm{Double}(n))).\end{aligned}$$

Addition (written infix) is defined similarly, this time with a parameter m:

$$\begin{aligned}m + 0 &:= m,\\ m + S(n) &:= S(m + n).\end{aligned}$$

Multiplication (again written infix) is defined using addition by

$$\begin{aligned}m \cdot 0 &:= 0,\\ m \cdot S(n) &:= (m \cdot n) + m.\end{aligned}$$

Similarly we can define all primitive recursive functions.

Up to now we have only considered examples of total functions, in the sense that total arguments are mapped to total values. But recall that in our setting functions need not be total. To give an example consider the algebra of streams defined by

$$\mathbb{S}(\alpha) := \mu_\xi(\alpha \to \xi \to \xi)$$

with a single constructor $\mathrm{C}\colon \alpha \to \mathbb{S}(\alpha) \to \mathbb{S}(\alpha)$. We write $\mathrm{C}_a u$ or $a :: u$ for $\mathrm{C}au$. This algebra differs from the ones previously considered by not having a nullary constructor. As a consequence it does not have non-empty total ideals, but clearly cototal ones. An example is $\{\, \mathrm{C}_a^n(*) \mid n \geq 1 \,\}$. We define the function Map of type $(\rho \to \sigma) \to \mathbb{S}(\rho) \to \mathbb{S}(\sigma)$ mapping its function argument $h\colon \rho \to \sigma$ over a stream u of type $\mathbb{S}(\rho)$ by the computation rule

$$\mathrm{Map}_h(a :: u) := (ha) :: \mathrm{Map}_h(u).$$

2.2.2. Recursion operators. Important examples of such constants D are the (structural) higher type *recursion operators* $\mathcal{R}_\iota^\tau$ introduced by Hilbert (1925) and Gödel (1958). They are used to construct maps from the algebra ι to the value type τ, by recursion on the structure of ι.

For instance, $\mathcal{R}_{\mathbb{N}}^\tau$ has type $\mathbb{N} \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \tau$. It is defined by the computation rules

$$\begin{aligned}\mathcal{R}_{\mathbb{N}}^\tau 0af &:= a,\\ \mathcal{R}_{\mathbb{N}}^\tau (Sn)af &:= fn(\mathcal{R}_{\mathbb{N}}^\tau naf).\end{aligned}$$

The first argument is the recursion argument, the second one gives the base value, and the third one gives the step function, mapping the recursion argument and the previous value to the next value. For example, $\mathcal{R}_{\mathbb{N}}^{\mathbb{N}} nm\lambda_{n,l}(Sl)$ defines addition $m+n$ by recursion on n. For $\lambda_{n,l}(Sl)$ we often write $\lambda_{\_,l}(Sl)$ since the bound variable n is not used.

It will be convenient to write an algebra form

$$\mu_\xi(\vec{\alpha}_i \to (\xi)_{\nu<n_i} \to \xi)_{i<k} \quad \text{as} \quad \mu_\xi((\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi)_{i<k}.$$

Definition (Type of $\mathcal{R}^\tau_\iota$). For the algebra $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi)_{i<k}$ and the result type τ we define the type of the recursion operator $\mathcal{R}^\tau_\iota$ to be

$$\iota \to ((\rho_{i\nu}(\iota \times \tau))_{\nu<n_i} \to \tau)_{i<k} \to \tau.$$

Here ι is the type of the recursion argument, and each $(\rho_{i\nu}(\iota\times\tau))_{\nu<n_i} \to \tau$ is called a *step type*. Usage of $\iota \times \tau$ rather than τ in the step types can be seen as a "strengthening", since then one has more data available to construct the value of type τ. Moreover, for recursive argument types we avoid the product type in $\iota \times \tau$ and take the two argument types ι and τ instead ("duplication").

Definition (Computation rules for $\mathcal{R}^\tau_\iota$). Let

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

be the type of the i-th constructor C_i of ι and consider a term $\mathrm{C}_i\vec{x}$ of type ι. We write $\vec{x}^P = x^P_0, \ldots, x^P_{m-1}$ for the *parameter arguments* $x^{\alpha_0}_0, \ldots, x^{\alpha_{m-1}}_{m-1}$ and $\vec{x}^R = x^R_0, \ldots, x^R_{n-1}$ for the *recursive arguments* $x^\iota_m, \ldots, x^\iota_{m+n-1}$. Writing $\mathcal{R}$ for $\mathcal{R}^\tau_\iota$ we take as its computation rules

$$\mathcal{R}(\mathrm{C}_i\vec{x})\vec{f} := f_i\vec{x}(\mathcal{R}x^R_0\vec{f}\,)\ldots(\mathcal{R}x^R_{n-1}\vec{f}\,).$$

Examples.

$$\begin{aligned}
&\mathcal{R}^\tau_{\mathbb{B}}\colon \mathbb{B} \to \tau \to \tau \to \tau,\\
&\mathcal{R}^\tau_{\mathbb{N}}\colon \mathbb{N} \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \tau,\\
&\mathcal{R}^\tau_{\mathbb{P}}\colon \mathbb{P} \to \tau \to (\mathbb{P} \to \tau \to \tau) \to (\mathbb{P} \to \tau \to \tau) \to \tau,\\
&\mathcal{R}^\tau_{\mathbb{Y}}\colon \mathbb{Y} \to \tau \to (\mathbb{Y} \to \tau \to \mathbb{Y} \to \tau \to \tau) \to \tau,\\
&\mathcal{R}^\tau_{\mathbb{L}(\rho)}\colon \mathbb{L}(\rho) \to \tau \to (\rho \to \mathbb{L}(\rho) \to \tau \to \tau) \to \tau,\\
&\mathcal{R}^\tau_{\rho+\sigma}\colon \rho + \sigma \to (\rho \to \tau) \to (\sigma \to \tau) \to \tau,\\
&\mathcal{R}^\tau_{\rho\times\sigma}\colon \rho \times \sigma \to (\rho \to \sigma \to \tau) \to \tau.
\end{aligned}$$

It is a helpful exercise to write out the computation rules for these particular recursion operators.

There is an important variant of recursion, where no recursive calls occur. This variant is called the *cases operator*; it distinguishes cases according to the outer constructor form. For the algebra $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi)_{i<k}$ and result type τ the type of the cases operator $\mathcal{C}^\tau_\iota$ is

$$\iota \to ((\rho_{i\nu}(\iota))_{\nu<n_i} \to \tau)_{i<k} \to \tau.$$

The simplest example (for type $\mathbb{B}$) is *if-then-else*. Another example is

$$\mathcal{C}^\tau_{\mathbb{N}} \colon \mathbb{N} \to \tau \to (\mathbb{N} \to \tau) \to \tau.$$

It could be used to define the *predecessor* function on $\mathbb{N}$ by the term

$$\mathrm{P}m := \mathcal{C}^{\mathbb{N}}_{\mathbb{N}} m 0 (\lambda_n n).$$

However, it is easier to define the predecessor function by the computation rules $\mathrm{P}0 := 0$ and $\mathrm{P}(Sn) := n$.

Remark. When computing the value of a cases term, we do not want to (eagerly) evaluate all arguments, but rather compute the test argument first and depending on the result (lazily) evaluate at most one of the other arguments. This phenomenon is well known in functional languages; for instance, in Scheme the `if`-construct is called a *special form* (as opposed to an operator). Therefore instead of taking the cases operator applied to a full list of arguments, one rather uses a `case`-construct to build this term; it differs from the former only in that it employs lazy evaluation. Hence the predecessor function could be written in the form

$$[\mathbf{case}\ m^{\mathbb{N}}\ \mathbf{of}\ (0 \mapsto 0 \mid Sn \mapsto n)].$$

2.2.3. Corecursion. The computation rules for $\mathcal{R}$ work from the leaves towards the root, and terminate because total ideals are finite. If, however, we deal with cototal ideals, then a similar operator is available to define functions with cototal ideals as values, namely "corecursion".

To understand the type of a corecursion operator recall the constructor types $\kappa_i(\iota)$ of an algebra $\iota = \mu_\xi(\kappa_0, \dots, \kappa_{k-1})$, written as

$$(\rho_{i\nu}(\iota))_{\nu<n_i} \to \iota \quad (i < k).$$

The product of these k constructor types is isomorphic to

$$\sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota) \to \iota$$

and the type of the recursion operator $\mathcal{R}^\tau_\iota$ is isomorphic to

$$\iota \to \Big(\sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota \times \tau) \to \tau \Big) \to \tau.$$

Dually for the algebra ι the type of its *destructor* D_ι (disassembling a constructor-built object into its parts) is

$$\iota \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota).$$

The corecursion operator ${}^{\mathrm{co}}\mathcal{R}^\tau_\iota$ is used to construct a map from τ to ι by "corecursion" on the structure of ι. Its type is

$$\tau \to \Big(\tau \to \sum_{i<k} \prod_{\nu<n_i} \rho_{i\nu}(\iota+\tau)\Big) \to \iota. \tag{8}$$

Examples (Types and computation rules of corecursion operators).

$$\begin{aligned}
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{B}} \colon \tau \to (\tau \to \mathbb{U}+\mathbb{U}) \to \mathbb{B},\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{N}} \colon \tau \to (\tau \to \mathbb{U}+(\mathbb{N}+\tau)) \to \mathbb{N},\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{P}} \colon \tau \to (\tau \to \mathbb{U}+(\mathbb{P}+\tau)+(\mathbb{P}+\tau)) \to \mathbb{P},\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{Y}} \colon \tau \to (\tau \to \mathbb{U}+(\mathbb{Y}+\tau)\times(\mathbb{Y}+\tau)) \to \mathbb{Y},\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}(\rho)} \colon \tau \to (\tau \to \mathbb{U}+\rho\times(\mathbb{L}(\rho)+\tau)) \to \mathbb{L}(\rho),\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{S}(\rho)} \colon \tau \to (\tau \to \rho\times(\mathbb{S}(\rho)+\tau)) \to \mathbb{S}(\rho).
\end{aligned}$$

The computation rule for each of these is defined below. For $f\colon \rho\to\tau$ and $g\colon \sigma\to\tau$ we denote $\lambda_x(\mathcal{R}^\tau_{\rho+\sigma}xfg)$ of type $\rho+\sigma\to\tau$ by $[f,g]$, and similary for ternary sumtypes etcetera. The identity functions id below are of type $\iota\to\iota$ with ι the respective algebra.

$$\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{B}}xf &:= [\lambda_\mathtt{tt}, \lambda_\mathtt{ff}](fx),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{N}}xf &:= [\lambda_0, \lambda_y(S([\mathrm{id}^{\mathbb{N}\to\mathbb{N}}, P_{\mathbb{N}}]y))](fx),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{P}}xf &:= [\lambda_1, \lambda_y(S_0([\mathrm{id}, P_{\mathbb{P}}]y)), \lambda_y(S_1([\mathrm{id}, P_{\mathbb{P}}]y))](fx),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{Y}}xf &:= [\lambda_0, \lambda_{y_0,y_1}(\mathrm{C}([\mathrm{id}, P_{\mathbb{Y}}]y_0)([\mathrm{id}, P_{\mathbb{Y}}]y_1))](fx),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}(\rho)}xf &:= [\lambda_[], \lambda_{y_0,y_1}(y_0 :: [\mathrm{id}, P_{\mathbb{L}(\rho)}]y_1)](fx),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{S}(\rho)}xf &:= (fx)_0 :: [\mathrm{id}, P_{\mathbb{S}(\rho)}](fx)_1
\end{aligned}$$

with $(fx)_i$ the i-th component of the pair fx, and $P_\alpha := \lambda_x({}^{\mathrm{co}}\mathcal{R}^\tau_\alpha xf)$ for $\alpha \in \{\mathbb{N}, \mathbb{P}, \mathbb{Y}, \mathbb{L}(\rho), \mathbb{S}(\rho)\}$.

Definition. The (single) computation rule for ${}^{\mathrm{co}}\mathcal{R}^\tau_\iota$ of type (8) is

$${}^{\mathrm{co}}\mathcal{R}^\tau_\iota xf := [g_0, \dots, g_{k-1}](fx)$$

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

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

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

and $P := \lambda_x({}^{\mathrm{co}}\mathcal{R}^\tau_\iota xf)$ contains the corecursive call.

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

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

$$
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} x f = \begin{cases} 0 & \text{if } fx = \mathrm{DummyL}^{\mathbb{U}+(\mathbb{Y}+\tau)\times(\mathbb{Y}+\tau)} \\ \mathrm{C} t s & \text{if } fx = \mathrm{Inr}\langle \mathrm{InL}^{\mathbb{Y}\to\mathbb{Y}+\tau} t, \mathrm{InL}^{\mathbb{Y}\to\mathbb{Y}+\tau} s\rangle \\ \mathrm{C} t({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} y f) & \text{if } fx = \mathrm{Inr}\langle \mathrm{InL}^{\mathbb{Y}\to\mathbb{Y}+\tau} t, \mathrm{InR}^{\tau\to\mathbb{Y}+\tau} y\rangle \\ \mathrm{C}({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} y f) s & \text{if } fx = \mathrm{Inr}\langle \mathrm{InR}^{\tau\to\mathbb{Y}+\tau} y, \mathrm{InL}^{\mathbb{Y}\to\mathbb{Y}+\tau} s\rangle \\ \mathrm{C}({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} y f)({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} z f) & \text{if } fx = \mathrm{Inr}\langle \mathrm{InR}^{\tau\to\mathbb{Y}+\tau} y, \mathrm{InR}^{\tau\to\mathbb{Y}+\tau} z\rangle \end{cases}
$$

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

Recall that Map of type $(\rho \to \sigma) \to \mathbb{S}(\rho) \to \mathbb{S}(\sigma)$ maps its function argument $h\colon \rho \to \sigma$ over a stream u of type $\mathbb{S}(\rho)$ (see Section 2.2.1, page 38). It is an easy exercise to give an alternative definition of the function Map by means of the corecursion operator.

REMARK. It is possible to define interesting cototal objects by means of corecursion operators. For instance the rightmost infinite path in the algebra $\mathbb{Y}$ of binary trees is $t_R := {}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{Y}} x_0 f_0$ with τ, x_0 arbitrary (for instance $\tau := \mathbb{U}$, $x_0 := \mathrm{Dummy}^{\mathbb{U}}$) and

$$
f_0 x^{\tau} := \mathrm{Inr}\langle \mathrm{InL}^{\mathbb{Y}\to\mathbb{Y}+\tau} 0, \mathrm{InR}^{\tau\to\mathbb{Y}+\tau} x\rangle.
$$

For the leftmost path we similary have t_L.

Another example is a function converting a real number given as a Cauchy sequence of rationals together a Cauchy modulus into an infinite stream of signed digits $\{-1, 0, 1\}$. Such a function can be defined by a simple corecursion. One can extract it from a proof (using "coinduction") of the fact that such a conversion exists.

2.3. Denotational semantics

We now set up a connection between the model $(|\mathbf{C}_\rho|)_\rho$ of partial continuous functionals described in Section 2.1 and the term system T^+ from Section 2.2. The main point is to clarify how we can use computation rules to define an ideal z in a function space. The general 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 $\langle\rangle$ we take the empty list. $(\vec{P}, Q)(\vec{V}, \vec{w})$ is covered by the induction hypothesis. Finally

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

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 C and defined constant D we have

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

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

This "denotational semantics" has good properties; however, we do not carry out the proofs here but rather refer to the literature. First of all, one can prove that $[\![\lambda_{\vec{x}} M]\!]$ is an ideal. Moreover, our 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.

CHAPTER 3

A theory TCF of computable functionals

After getting clear about the domains we intend to reason about, the partial continuous functionals and in particular the computable ones, we now set up a theory to prove their properties. The main concepts are those of inductively and coinductively defined predicates.

3.1. Formulas and their computational content

Formulas are built up from prime formulas $P\vec{t}$ by implication $\to$ and universal quantification $\forall_x$; here the t_i are terms, x is a variable and P is a predicate of a certain arity (a list of types). We often write $\vec{t} \in P$ for $P\vec{t}$. Predicates can be inductively or coinductively defined. An example for the former is $T_{\mathbb{L}(\mathbb{N})}$, which is defined by the clauses (i) $[] \in T_{\mathbb{L}(\mathbb{N})}$ and (ii) $\forall_{n\in\mathbb{N}}(\ell \in T_{\mathbb{L}(\mathbb{N})} \to n :: \ell \in T_{\mathbb{L}(\mathbb{N})})$. An example for the latter is ${}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ defined by a closure axiom saying that every $\ell \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ is of the form $n :: \ell'$ with $n \in \mathbb{N}$ and $\ell' \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ again. According to Kolmogorov (1932) a formula can be seen as a problem, asking for a solution. In the inductive example a solution for $4 :: 2 :: 0 :: [] \in T_{\mathbb{L}(\mathbb{N})}$ would be the generating (finite) sequence $[]$, $0 :: []$, $2 :: 0 :: []$, $4 :: 2 :: 0 :: []$, and in the coinductive example a solution for a prime formula $t \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ would be an (infinite) stream of natural numbers. Generally, a solution for an inductive predicate is a finite construction tree, and for a coinductive predicate a finitely branching possibly infinite destruction tree. Such trees can be seen as ideals of the free algebras considered in Section 2.1.4. A solution for a problem posed by the formula $A \to B$ is a computable functional mapping solutions of A into solutions of B.

Sometimes the solution of a problem does not need all available input. We therefore mark the sources of such computationally superfluous input – that is, some (co)inductive predicates – as "non-computational" (n.c.).

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 $X^c, Y^c, Z^c, W^c \dots$ and "non-computational" ones X^{nc}, Y^{nc}, Z^{nc}, W^{nc} ..., and use $X, Y, Z, W \dots$ for both.

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

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

with all predicate variables Y_i^c, Z_i^{nc}, W_i^{nc} 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.

EXAMPLES. Recall that $[]^{\mathbb{L}(\alpha)}$ and $\mathrm{Cons}^{\alpha\to\mathbb{L}(\alpha)\to\mathbb{L}(\alpha)}$ are the two constructors of the algebra form $\mathbb{L}(\alpha)$ of lists, written $[]$ and $::$ (infix).

1. Let Y of arity (α,α) and X of arity $(\mathbb{L}(\alpha),\mathbb{L}(\alpha))$ be predicate variables. Then

$$K_0 := X([],[]),$$
$$K_1 := \forall_{x,x'}(Y(x,x') \to \forall_{\ell,\ell'}(X(\ell,\ell') \to X(x :: \ell, x' :: \ell')))$$

are clauses and both *similarity* $\sim_{\mathbb{L}}(Y)$ (or $\sim_{\mathbb{L},Y}$) defined by $\mu_X(K_0,K_1)$ and *bisimilarity* $\approx_{\mathbb{L}}(Y)$ (or $\approx_{\mathbb{L},Y}$) defined by $\nu_X(K_0,K_1)$ are predicate forms with Y a parameter predicate variable. Note that we can omit the type parameter α, since it can be read off from the arity of Y.

2. Alternatively let Y of arity (α) and X of arity $(\mathbb{L}(\alpha))$ be predicate variables. Then

$$K_0 := ([] \in X),$$
$$K_1 := \forall_x(x \in Y \to \forall_\ell(\ell \in X \to x :: \ell \in X))$$

are clauses and both *relative totality* $T_{\mathbb{L}}(Y)$ (or $T_{\mathbb{L},Y}$) defined by $\mu_X(K_0,K_1)$ and also *relative cototality* ${}^{\mathrm{co}}T_{\mathbb{L}}(Y)$ (or ${}^{\mathrm{co}}T_{\mathbb{L},Y}$) defined by $\nu_X(K_0,K_1)$ are predicate forms with Y a parameter predicate variable.

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 $\iota_{I^c} := \mu_\xi\vec{\kappa}$.

EXAMPLES. For each of the predicate forms $\sim_{\mathbb{L}}$, $\approx_{\mathbb{L}}$, $T_{\mathbb{L}}$, ${}^{\mathrm{co}}T_{\mathbb{L}}$ defined above the associated algebra form is $\mathbb{L}$. Notice that the distinction whether a predicate form I is defined as a least or greatest fixed point is ignored.

DEFINITION (Predicates and formulas).

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

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.). Now we require that all predicate substitutions involved in $(I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)$ 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 2.2.1, 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 $\{\, \vec{y} \mid Ct\vec{y} \,\}$.

It is a natural question to ask what the type of a "realizer" or "witness" of a c.r. predicate or formula C should be.

DEFINITION (Type $\tau(C)$ of a c.r. predicate or formula C). Assume a global injective assignment of type variables ξ to c.r. predicate variables X^c.

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

where $\vec{P}^c$ are the c.r. predicates among $\vec{P}$ and ι_{I^c} is the algebra form associated with the predicate form $I^c/{}^{\mathrm{co}}I^c$. We call $\iota_{I^c}(\tau(\vec{P}^c))$ the *algebra associated with* $(I^c/{}^{\mathrm{co}}I^c)(\vec{\rho},\vec{P})$.

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

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

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

2. For every closed base type $\iota(\vec{\rho})$ we define c.r. *equality* predicates $\sim_\iota(\vec{P}\,)$ and $\approx_\iota(\vec{P}\,)$ of arity $(\iota(\vec{\rho}), \iota(\vec{\rho}))$ by induction on the height $|\iota(\vec{\rho})| := 1 + \max|\vec{\rho}|$. Here ι is the name of an algebra form (for instance $\mathbb{L}$), the ρ_i are closed base types and the P_i are predicates of arity (ρ_i, ρ_i).

(i). $\sim_{\mathbb{N}} := \mu_{X^c}(K_0, K_1)$ and $\approx_{\mathbb{N}} := \nu_{X^c}(K_0, K_1)$ with clauses

$$K_0 := X^c(0,0),$$
$$K_1 := \forall_{n,n'}(X^c(n,n') \to X^c(Sn, Sn')).$$

Then $\tau(\sim_{\mathbb{N}}) = \tau(\approx_{\mathbb{N}}) = \mathbb{N}$.

(ii). $\sim_{\mathbb{L}}(\approx_{\mathbb{N}}) := \mu_{X^c}(K_0, K_1)$ and $\approx_{\mathbb{L}}(\approx_{\mathbb{N}}) := \nu_{X^c}(K_0, K_1)$ with clauses

$$K_0 := X^c([\,],[\,]),$$
$$K_1 := \forall_{n,n'}(n \approx_{\mathbb{N}} n' \to \forall_{\ell,\ell'}(X^c(\ell,\ell') \to X^c(n :: \ell, n' :: \ell'))).$$

Then $\tau(\sim_{\mathbb{L}}(\approx_{\mathbb{N}})) = \tau(\approx_{\mathbb{L}}(\approx_{\mathbb{N}})) = \mathbb{L}(\mathbb{N})$.

(iii). $\sim_{\mathbb{L}}(\sim_{\mathbb{L}}(\approx_{\mathbb{N}})) := \mu_{X^c}(K_0, K_1)$ and $\approx_{\mathbb{L}}(\sim_{\mathbb{L}}(\approx_{\mathbb{N}})) := \nu_{X^c}(K_0, K_1)$ with clauses

$$K_0 := X^c([\,],[\,]),$$
$$K_1 := \forall_{\ell,\ell'}(\ell \sim_{\mathbb{L}}(\approx_{\mathbb{N}}) \ell' \to \forall_{u,u'}(X^c(u,u') \to X^c(\ell :: u, \ell' :: u'))).$$

Then $\tau(\sim_{\mathbb{L}}(\sim_{\mathbb{L}}(\approx_{\mathbb{N}}))) = \tau(\approx_{\mathbb{L}}(\sim_{\mathbb{L}}(\approx_{\mathbb{N}}))) = \mathbb{L}(\mathbb{L}(\mathbb{N}))$.

3. The *transitive closure* of a binary relation is defined as follows. Let

$$K_0 := \forall_{x,y}(Yxy \to Xxy),$$
$$K_1 := \forall_{x,y,z}(Yxy \to Xyz \to Xxz)$$

where x, y, z are variables of type α and Y, X predicate variables of arity (α, α). From these clauses we define the predicate form

$$\mathrm{TC} := \mu_X(K_0, K_1)$$

with parameter type variable α and parameter predicate variable Y.

Let $<$ be a c.r. "less than" relation on the natural numbers, of arity $(\mathbb{N}, \mathbb{N})$. Then $\mathrm{TC}(\mathbb{N}, <)$ is its transitive closure. The constructor types of $\tau(\mathrm{TC})$ are $\beta \to \xi$ and $\beta \to \xi \to \xi$, hence $\tau(\mathrm{TC}(\mathbb{N}, <))$ is the type of non-empty lists of natural numbers.

4. An important example of an inductive predicate is *Leibniz equality*, defined simply by

$$\mathrm{EqD} := \mu_{X^{\mathrm{nc}}}(\forall_x X^{\mathrm{nc}} xx) \qquad (\text{D for "inductively } \mathbf{d}\text{efined"}).$$

We will use the abbreviation

$$(x \equiv y) := \mathrm{EqD}(x, y).$$

The missing logical connectives disjunction, conjunction and existence can now be defined inductively. 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 the parameter predicates Y, Z can be chosen as either c.r. or n.c. we obtain the variants

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

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

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

We use the abbreviations

$$\begin{aligned}
P \cup^d Q &:= \mathrm{CupD}_{P,Q},\\
P \cup^l Q &:= \mathrm{CupL}_{P,Q},\\
P \cup^r Q &:= \mathrm{CupR}_{P,Q},\\
P \cup^u Q &:= \mathrm{CupU}_{P,Q},\\
P \cup^{\mathrm{nc}} Q &:= \mathrm{CupNc}_{P,Q}.
\end{aligned}$$

In case of nullary predicates we use

$$\begin{aligned} A \vee^{\mathrm{d}} B &:= \mathrm{CupD}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{l}} B &:= \mathrm{CupL}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{r}} B &:= \mathrm{CupR}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{u}} B &:= \mathrm{CupU}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{nc}} B &:= \mathrm{CupNc}_{\{|A\},\{|B\}}. \end{aligned}$$

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

Similarly conjunction is a special case of intersection

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

and we obtain the variants

$$\begin{aligned} \mathrm{CapD}_{Y^{\mathrm{c}},Z^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapL}_{Y^{\mathrm{c}},Z^{\mathrm{nc}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to Z^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapR}_{Y^{\mathrm{nc}},Z^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapNc}_{Y,Z} &:= \mu_{X^{\mathrm{nc}}}(\forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X^{\mathrm{nc}}\vec{x})). \end{aligned}$$

Then by definition

$$\begin{aligned} \tau(\mathrm{CapD}) = \mu_\xi(\beta_0 \to \beta_1 \to \xi) &= \beta_0 \times \beta_1 \\ \tau(\mathrm{CapL}) = \tau(\mathrm{CapR}) = \mu_\xi(\beta \to \xi) &= \mathbb{I}(\beta). \end{aligned}$$

We use the abbreviations

$$\begin{aligned} P \cap^{d} Q &:= \mathrm{CapD}_{P,Q}, \\ P \cap^{l} Q &:= \mathrm{CapL}_{P,Q}, \\ P \cap^{r} Q &:= \mathrm{CapR}_{P,Q}, \\ P \cap^{\mathrm{nc}} Q &:= \mathrm{CapNc}_{P,Q}. \end{aligned}$$

In case of nullary predicates we use

$$\begin{aligned} A \wedge^{\mathrm{d}} B &:= \mathrm{CapD}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{l}} B &:= \mathrm{CapL}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{r}} B &:= \mathrm{CapR}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{nc}} B &:= \mathrm{CapNc}_{\{|A\},\{|B\}}. \end{aligned}$$

Again since the decoration is determined by the c.r./n.c. status of the two parameter predicates we usually leave out the decoration and just write $\wedge$.

Finally existence is defined inductively by

$$\begin{aligned}\mathrm{Ex}_{Y^c} &:= \mu_{X^c}(\forall_x(x \in Y^c \to X^c)),\\ \mathrm{ExNc}_Y &:= \mu_{X^{\mathrm{nc}}}(\forall_x(x \in Y \to X^{\mathrm{nc}})).\end{aligned}$$

Then by definition

$$\tau(\mathrm{Ex}) = \mu_\xi(\beta \to \xi) = \mathbb{I}(\beta).$$

We use the abbreviation

$$\begin{aligned}\exists_x A &:= \mathrm{Ex}_{\{x|A\}},\\ \exists^{\mathrm{nc}}_x A &:= \mathrm{ExNc}_{\{x|A\}},\end{aligned}$$

and again since the decoration is determined by the c.r./n.c. status of the parameter predicate we usually leave out the decoration and just write $\exists$.

3.2. Axioms of TCF

We define a theory of computable functionals, called TCF. Formulas are the ones defined above, involving typed variables. Derivations use the rules of minimal logic for $\to$ and $\forall$, and the axioms introduced below. However, because of the presence of decorations we have an extra degree of freedom. By an *n.c. part* of a derivation we mean a subderivation with an n.c. end formula. Such n.c. parts will not contribute to the computational content of the whole derivation, and hence we can ignore all decorations in those parts (i.e., use a modified notion of equality of formulas there).

For each inductive predicate there are "clauses" or introduction axioms, together with a "least-fixed-point" or elimination axiom. 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 (Introduction and elimination axioms for inductive predicates). 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^-:

$$I_i^+ \colon \forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to I\vec{t}_i), \tag{9}$$

$$I^- \colon (\forall_{\vec{x}_i}((A_{i\nu}(I \cap X))_{\nu<n_i} \to X\vec{t}_i))_{i<k} \to I \subseteq X \tag{10}$$

($I \cap X$ was inductively defined above). (10) 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.

Remarks. (i) We use a "strengthened" form of the "step formula", 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 of the least-fixed-point axiom this simplifies the proof of the "step", since we have an additional I-hypothesis available.

(ii) Notice that there is no circularity here for the inductive predicate $Y \cap Z := \mathrm{Cap}_{Y,Z}$, since there are no recursive calls in this particular inductive definition and hence $\cap$ does not occur in

$$\mathrm{Cap}^-_{Y,Z} \colon \forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X\vec{x}) \to \mathrm{Cap}_{Y,Z} \subseteq X.$$

(iii) The elimination axiom (10) could equivalently be written as

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

In this form it fits better with our (i.e., Gentzen's) way to write the logical elimination rules, where the main premise comes first. More importantly, its type (cf. Section 3.1) will then be the type of the recursion operator $\mathcal{R}^\tau_\iota$ taken as the "computational content" (cf. Section 4.1) of the elimination axiom for I. Therefore in the implementation of TCF this form is used. However, for readability we often prefer the form (10) in the present notes.

Examples. 1. For Even the introduction axioms are

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

and the elimination axiom is Even^-:

$$0 \in X \to \forall_n(n \in \mathrm{Even} \to n \in X \to S(Sn) \in X) \to \mathrm{Even} \subseteq X.$$

As an example of how to use these axioms we prove an "inversion" property:

$$n \in \mathrm{Even} \leftrightarrow n \equiv 0 \vee \exists_{n'}(n' \in \mathrm{Even} \wedge n \equiv S(Sn')).$$

"$\leftarrow$". Use the introduction axioms. "$\rightarrow$". Use Even^- with competitor predicate $X := \{\, n \mid n \equiv 0 \vee \exists_{n'}(n' \in \mathrm{Even} \wedge n \equiv S(Sn'))\,\}$. It suffices to prove the premises of Even^- for this X. For the first premise this is clear. For the second assume n with $n \in \mathrm{Even} \cap X$. The goal is $S(Sn) \in X$. It suffices to show $\exists_{n'}(n' \in \mathrm{Even} \wedge S(Sn) \equiv S(Sn'))$. Take n.

2. Consider the algebra $\mathbb{Y}$ of binary trees. We want to represent the set of total ideals by an inductively defined predicate $T_{\mathbb{Y}}$. The clauses are

$$(T_{\mathbb{Y}})^+_0 \colon - \in T_{\mathbb{Y}},$$
$$(T_{\mathbb{Y}})^+_1 \colon \forall_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \to \mathrm{C}t_1t_2 \in T_{\mathbb{Y}})$$

and the elimination axiom is

$$T^-_{\mathbb{Y}} \colon - \in X \to \forall_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \cap X \to \mathrm{C}t_1t_2 \in X) \to T_{\mathbb{Y}} \subseteq X.$$

From these axioms we prove the inversion property:

$$t \in T_{\mathbb{Y}} \leftrightarrow (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2).$$

"$\leftarrow$". Use the introduction axioms. "$\rightarrow$". Use $T^-_{\mathbb{Y}}$ with competitor predicate $X := \{\, t \mid (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2\,\}$. It suffices to prove the premises of $T^-_{\mathbb{Y}}$ for this X. For the first premise this is clear. For the second

assume t_1, t_2 with $t_1, t_2 \in T_\Upsilon \cap X$. The goal is $\mathrm{C}t_1t_2 \in X$. It suffices to show $\exists_{t_1',t_2'}(t_1', t_2' \in T_\Upsilon \wedge \mathrm{C}t_1't_2' \equiv \mathrm{C}t_1t_2)$. Take t_1, t_2.

3. For the transitive closure $\mathrm{TC}_\prec := \mathrm{TC}(\rho, \prec)$ of a binary relation $\prec$ on objects of type ρ the introduction axioms are

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

and the elimination axiom is $\mathrm{TC}^-_\prec$:

$$\forall_{x,y}(x \prec y \to Xxy) \to \forall_{x,y,z}(x \prec z \to \mathrm{TC}_\prec(z,y) \to Xzy \to Xxy) \to \mathrm{TC}_\prec \subseteq X.$$

The inversion property

$$\mathrm{TC}_\prec(x,y) \leftrightarrow x \prec y \vee \exists_z(x \prec z \wedge \mathrm{TC}_\prec(z,y))$$

can be proved as above. For "$\leftarrow$" use the introduction axioms, and for "$\rightarrow$" use $\mathrm{TC}^-_\prec$ with competitor $X := \{\, x, y \mid x \prec y \vee \exists_z(x \prec z \wedge \mathrm{TC}_\prec(z,y)) \,\}$. It suffices to prove the premises of $\mathrm{TC}^-_\prec$ for this X. For the first premise this is clear. For the second assume x, y, z with $x \prec z$ and $\mathrm{TC}_\prec(z,y)$ and Xzy. The goal is Xxy. It suffices to show $\exists_z(x \prec z \wedge \mathrm{TC}_\prec(z,y))$. Take z.

4. For $\sim_{X\times Y} := \sim_\times(\rho, \sigma, X, Y)$ the introduction axiom is

$$\forall_{x,x',y,y'}(Xxx' \to Yyy' \to \langle x,y\rangle \sim_{X\times Y} \langle x',y'\rangle)$$

and the elimination axiom is $\sim^-_{X\times Y}$:

$$\forall_{x,x',y,y'}(Xxx' \to Yyy' \to Z(\langle x,y\rangle, \langle x',y'\rangle)) \to \sim_{X\times Y} \subseteq Z.$$

Hence $\sim_{X\times Y} = \{\, (\langle x,y\rangle, \langle x',y'\rangle) \mid Xxx', Yyy' \,\}$. The inversion property now is

$$p \sim_{X\times Y} p' \leftrightarrow \exists_{x,x',y,y'}(Xxx' \wedge Yyy' \wedge p \equiv \langle x,y\rangle \wedge p' \equiv \langle x',y'\rangle).$$

For "$\leftarrow$" use the introduction axioms, and for "$\rightarrow$" use $\sim^-_{X\times Y}$ with competitor $Z := \{\, p, p' \mid \exists_{x,x',y,y'}(Xxx' \wedge Yyy' \wedge p \equiv \langle x,y\rangle \wedge p' \equiv \langle x',y'\rangle \,\}$. It suffices to prove the premise of $\sim^-_{X\times Y}$ for this Z. Assume x, x', y, y' with Xxx' and Yyy' and $p \equiv \langle x,y\rangle$ and $p' \equiv \langle x',y'\rangle$. The goal is Zpp'. It suffices to show $\exists_{x,x',y,y'}(Xxx' \wedge Yyy' \wedge p \equiv \langle x,y\rangle \wedge p' \equiv \langle x',y'\rangle)$. Take x, x', y, y'.

Similarly for $\sim_{X+Y} := \sim_+(\rho, \sigma, X, Y)$ the introduction axioms are

$$\forall_{x,x'}(Xxx' \to \mathrm{InL}(x) \sim_{X+Y} \mathrm{InL}(x')),$$
$$\forall_{y,y'}(Yyy' \to \mathrm{InR}(y) \sim_{X+Y} \mathrm{InR}(y'))$$

and the elimination axiom is

$$\forall_{x,x'}(Xxx' \to Z(\mathrm{InL}(x), \mathrm{InL}(x')) \to$$
$$\forall_{y,y'}(Yyy' \to Z(\mathrm{InR}(y), \mathrm{InR}(y')) \to \sim_{X+Y} \subseteq Z.$$

Hence $\sim_{X+Y} = \{\,(\mathrm{InL}(x), \mathrm{InL}(x')) \mid Xxx'\,\} \cup \{\,(\mathrm{InR}(y), \mathrm{InR}(y')) \mid Yyy'\,\}$.

Recall the inductive definitions of existence, intersection and union given above. For nullary predicates $P = \{\mid A\,\}$ and $Q = \{\mid B\,\}$ we write $A \wedge B$ for $P \cap Q$ and $A \vee B$ for $P \cup Q$. Then – as in Chapter 1 – the introduction axioms are

$$\begin{aligned}
&\forall_x(A \to \exists_x A),\\
&A \to B \to A \wedge B,\\
&A \to A \vee B, \qquad B \to A \vee B
\end{aligned}$$

and the elimination axioms are (now written in the equivalent form mentioned above, where the main premise comes first)

$$\begin{aligned}
&\exists_x A \to \forall_x(A \to B) \to B \qquad (x \notin \mathrm{FV}(B)),\\
&A \wedge B \to (A \to B \to C) \to C,\\
&A \vee B \to (A \to C) \to (B \to C) \to C.
\end{aligned}$$

To understand the axioms for coinductive predicates note that the conjunction of the k clauses (9) of an inductive predicate I is equivalent to

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

Definition (Closure and greatest-fixed-point axioms). 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 dual ${}^{\mathrm{co}}I$ by the *closure axiom* ${}^{\mathrm{co}}I^-$ and the *greatest-fixed-point axiom* ${}^{\mathrm{co}}I^+$:

$$(11) \qquad {}^{\mathrm{co}}I^-\colon \forall_{\vec{x}}({}^{\mathrm{co}}I\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I) \wedge \vec{x} \equiv \vec{t}_i))$$

$$(12) \qquad {}^{\mathrm{co}}I^+\colon \forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I \cup X) \wedge \vec{x} \equiv \vec{t}_i)) \to X \subseteq {}^{\mathrm{co}}I.$$

(${}^{\mathrm{co}}I \cup X$ was inductively defined above). The axiom 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. substitutions for type and predicate variables) as axioms.

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 the greatest-fixed-point axiom this simplifies the proof of the "step", since its conclusion is weaker.

Remark. The greatest-fixed-point axiom (12) could be written as

$$\forall_{\vec{x}}(X\vec{x} \to \forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I \cup X) \wedge \vec{x} \equiv \vec{t}_i)) \to {}^{\mathrm{co}}I\vec{x}).$$

Then its type will be the type of the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^\tau_\iota$ taken as the "computational content" (cf. Section 4.1) of the greatest-fixed-point axiom

for ${}^{\mathrm{co}}I$. Therefore in the implementation of TCF this form is used. However, for readability we prefer the form (12) in the present notes.

REMARK. Instead of Leibniz equality $\equiv$ in (11) and (12) we could also use a different equality relation, for instance the n.c. variant $\doteq^{\mathrm{nc}}$ of pointwise equality to be introduced in Section 3.3. This leads to a new variant of ${}^{\mathrm{co}}I$.

EXAMPLES. 1. To show how to construct the dual ${}^{\mathrm{co}}I$ of an inductive predicate I we consider the predicate Even. The conjunction of its two clauses is equivalent to

$$\forall_n(n \equiv 0 \vee \exists_{n'}(n' \in \mathrm{Even} \wedge n \equiv S(Sn')) \to n \in \mathrm{Even}).$$

Now the dual ${}^{\mathrm{co}}\mathrm{Even}$ of Even is defined by its closure axiom ${}^{\mathrm{co}}\mathrm{Even}^-$:

$$\forall_n(n \in {}^{\mathrm{co}}\mathrm{Even} \to n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \wedge n \equiv S(Sn')))$$

and its greatest-fixed-point axiom ${}^{\mathrm{co}}\mathrm{Even}^+$:

$$\forall_n(Xn \to n \equiv 0 \vee \exists_{n'}(n' \in ({}^{\mathrm{co}}\mathrm{Even} \cup X) \wedge n \equiv S(Sn'))) \to X \subseteq {}^{\mathrm{co}}\mathrm{Even}.$$

Also for ${}^{\mathrm{co}}\mathrm{Even}$ from these axioms we can prove an inversion property:

$$n \in {}^{\mathrm{co}}\mathrm{Even} \leftrightarrow n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \wedge n \equiv S(Sn')).$$

"$\to$". Use the closure axiom. "$\leftarrow$". Use ${}^{\mathrm{co}}\mathrm{Even}^+$, with competitor $X := \{\, n \mid n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \wedge n \equiv S(Sn')) \,\}$. It suffices to prove the premise of ${}^{\mathrm{co}}\mathrm{Even}^+$ for this X. Assume n with $n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \wedge n \equiv S(Sn'))$. The goal is $n \equiv 0 \vee \exists_{n'}(n' \in ({}^{\mathrm{co}}\mathrm{Even} \cup X) \wedge n \equiv S(Sn'))$. We argue by cases on the assumed disjunction. In the first case we are done. In the second case take the n' provided.

2. Consider the algebra $\mathbb{Y}$ of binary trees. We want to represent the set of cototal binary trees (cf. Section 2.1.6) by a coinductively defined predicate ${}^{\mathrm{co}}T_{\mathbb{Y}}$. Recall that the conjunction of the two clauses of $T_{\mathbb{Y}}$ is equivalent to

$$\forall_t((t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2) \to t \in T_{\mathbb{Y}}.)$$

Since $T_{\mathbb{Y}}$ is the least predicate with this property we even have equivalence

$$\forall_t(t \in T_{\mathbb{Y}} \leftrightarrow (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2)),$$

called inversion property on page 52. Now how can we formally represent cototality of a binary tree? The idea is to define ${}^{\mathrm{co}}T_{\mathbb{Y}}$ as the *largest* set satisfying the equivalence. Formulated differently, cototal ideals are not built from initial objects by construction (synthesized), but rather defined by the property that they can always be destructed (analysed). Therefore we require

$${}^{\mathrm{co}}T^-_{\mathbb{Y}} : \forall_t(t \in {}^{\mathrm{co}}T_{\mathbb{Y}} \to (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2))$$

A set built by construction steps (synthesis) is meant to be the least set closed under these steps. Similary, a set described by destruction (analysis) is meant to be the largest set closed under destruction. Hence we require

$$ {}^{\mathrm{co}}T^{+}_{\mathbb{Y}} : \forall_t(t \in X \to (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \cup X \wedge t \equiv \mathrm{C}t_1t_2)) \to X \subseteq {}^{\mathrm{co}}T_{\mathbb{Y}}. $$

${}^{\mathrm{co}}T^{+}_{\mathbb{Y}}$ expresses that every competitor X satisfying the closure property is below ${}^{\mathrm{co}}T_{\mathbb{Y}}$. As an example of how to use these axioms we prove that the equivalence above holds for ${}^{\mathrm{co}}T_{\mathbb{Y}}$ as well:

$$ \forall_t(t \in {}^{\mathrm{co}}T_{\mathbb{Y}} \leftrightarrow (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2)), $$

"$\to$". Use the closure axiom ${}^{\mathrm{co}}T^{-}_{\mathbb{Y}}$. "$\leftarrow$". Use ${}^{\mathrm{co}}T^{+}_{\mathbb{Y}}$, with competitor $X := \{\, t \mid (t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2) \,\}$. It suffices to prove the premise of ${}^{\mathrm{co}}T^{+}_{\mathbb{Y}}$ for this X. Assume t with $(t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge t \equiv \mathrm{C}t_1t_2)$. The goal is $(t \equiv -) \vee \exists_{t_1,t_2}(t_1, t_2 \in {}^{\mathrm{co}}T_{\mathbb{Y}} \cup X \wedge t \equiv \mathrm{C}t_1t_2)$. We argue by cases on the assumed disjunction. In the first case we are done. In the second case take the t_1, t_2 provided.

For n.c. inductive or coinductive predicates the axioms are formed as in the c.r. case, using $\vee^{\mathrm{nc}}$ 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. 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.

Generally, an inductive predicate is always contained in its dual.

LEMMA 3.2.1. $I \subseteq {}^{\mathrm{co}}I$, $I^{\mathrm{nc}} \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$.

PROOF. The least-fixed-point axiom (10) for I (i.e., I^{-}) is equivalent to

$$ \forall_{\vec{x}}(\bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I \cap X) \wedge \vec{x} \equiv \vec{t}_i) \to X\vec{x}\,) \to I \subseteq X. $$

It suffices that its premise holds with ${}^{\mathrm{co}}I$ for X. To this end we use the greatest-fixed-point axiom (12) (i.e., ${}^{\mathrm{co}}I^{+}$), with the competitor predicate

$$ X := \{\, \vec{x} \mid \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I \cap {}^{\mathrm{co}}I) \wedge \vec{x} \equiv \vec{t}_i) \,\}. $$

This means that we have to show the premise of (12) with this X, i.e.,

$$ \forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I \cup X) \wedge \vec{x} \equiv \vec{t}_i)). $$

But if we unfold the premise $X\vec{x}$, this follows from $I \cap {}^{\mathrm{co}}I \subseteq {}^{\mathrm{co}}I \cup X$. For I^{nc} the proof is similar. □

REMARK. In case of an inductive predicate with non-recursive clauses only also the reverse inclusions ${}^{\mathrm{co}}I_\iota \subseteq I_\iota$, ${}^{\mathrm{co}}I^{\mathrm{nc}} \subseteq I^{\mathrm{nc}}$. Hence it is not necessary to consider ${}^{\mathrm{co}}I$. Examples are the inductively defined logical connectives $\exists$, $\wedge$, $\vee$ with assigned algebras product $\times$, sum $+$ and $\mathtt{uysum}$, $\mathtt{ysumu}$, where the algebras $\mathtt{uysum}(\alpha)$, $\mathtt{ysumu}(\alpha)$ replace $\mathbb{U} + \alpha$, $\alpha + \mathbb{U}$.

LEMMA 3.2.2. $I \subseteq I^{\mathrm{nc}}$, ${}^{\mathrm{co}}I \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$.

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

For $I \subseteq I^{\mathrm{nc}}$ we use the elimination axiom (10) with I^{nc} as competitor predicate:

$$(\forall_{\vec{x}_i}((A_{i\nu}(I \cap I^{\mathrm{nc}}))_{\nu<n_i} \to I^{\mathrm{nc}}\vec{t}_i))_{i<k} \to I \subseteq I^{\mathrm{nc}}.$$

It suffices to prove the premises. Let $i < k$, fix $\vec{x}_i$ and assume $A_{i\nu}(I \cap I^{\mathrm{nc}})$ for all $\nu < n_i$. Since $A_{i\nu}(X)$ is strictly positive in X we obtain $A_{i\nu}(I^{\mathrm{nc}})$ for all $\nu < n_i$ and hence $I^{\mathrm{nc}}\vec{x}_i$ by $(I^{\mathrm{nc}})^+_i$.

For ${}^{\mathrm{co}}I \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$ we use the greatest-fixed-point axiom for ${}^{\mathrm{co}}I^{\mathrm{nc}}$ with ${}^{\mathrm{co}}I$ as competitor predicate:

$$\forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I^{\mathrm{nc}} \cup {}^{\mathrm{co}}I) \wedge \vec{x} \equiv \vec{t}_i)) \to {}^{\mathrm{co}}I \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}.$$

It suffices to prove the premise, which again follows from the fact that $A_{i\nu}(X)$ is strictly positive in X. □

Recall that for the n.c. Leibniz equality the introduction axiom is $x^\rho \equiv x^\rho$ and the elimination axiom is $x \equiv y \to \forall_X Xxx \to Xxy$. From this definition we can deduce the property Leibniz used as a definition.

LEMMA 3.2.3 (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 $\mathbf{F} := (\mathtt{ff} \equiv \mathtt{tt})$.

THEOREM 3.2.4 (Ex-falso-quodlibet). *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 $\mathrm{Ef}_{\mathrm{EqD}}\colon \mathbf{F} \to x^\rho \equiv y^\rho$. To see this, we first obtain $\mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy \equiv \mathcal{R}^\rho_{\mathbb{B}}\mathtt{ff}xy$ from the introduction axiom. 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 3.2.1. The cases $Y\vec{s}$, $A \to B$ and $\forall_x A$ are obvious. □

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 $St =_{\mathbb{N}} Ss$, or more precisely $=_{\mathbb{N}}(St, Ss)$, is identified with $t =_{\mathbb{N}} s$. We can now turn this boolean term into the formula $(St =_{\mathbb{N}} Ss) \equiv \mathtt{tt}$, which again is abbreviated by $St =_{\mathbb{N}} Ss$, but this time with the understanding that it is a formula. Then (importantly) the two formulas $St =_{\mathbb{N}} Ss$ 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 $St =_{\mathbb{N}} Ss \to t =_{\mathbb{N}} s$ explicitly.

Remark. One might wonder whether the weak (or classical) existential quantifier $\tilde{\exists}_x A$ is the same as the non-computational existential quantifier $\exists^{\mathrm{nc}}_x A$, since both in some sense computationally disregard the quantified variable x and the kernel A. We will argue that both quantifiers are equivalent if and only if a certain (non-computational) Markov axiom holds.

Note first that in our comparison we should use the arithmetical form $\tilde{\exists}_x A := \forall_x(A \to \mathbf{F}) \to \mathbf{F}$ rather than the logical form $\forall_x(A \to \bot) \to \bot$ of the weak existential quantifier. Otherwise there can be no relation, because $\bot$ is a predicate variable with computational content.

Clearly $\exists^{\mathrm{nc}}_x A$ implies $\tilde{\exists}_x A$, by $(\exists^{\mathrm{nc}})^-$ (take $\mathbf{F}$ for B). However, the converse is problematic. We do have an elimination scheme for $\tilde{\exists}_x A$ as well, but only for formulas B satisfying $((B \to \mathbf{F}) \to \mathbf{F}) \to B$. This means that for the converse we need a Markov axiom for the (non-computational) formula $\exists^{\mathrm{nc}}_x A$:

$$((\exists^{\mathrm{nc}}_x A \to \mathbf{F}) \to \mathbf{F}) \to \exists^{\mathrm{nc}}_x A; \tag{13}$$

from $\tilde{\exists}_x A \to \exists^{\mathrm{nc}}_x A$ we can easily derive (13). Although usage of such an n.c. axiom does not have any effect on computational content, we prefer to avoid it.

3.3. Equality and extensionality w.r.t. cotypes

The notion of equality is of central importance in any mathematical theory involving higher type objects. In our setting allowing infinite base type data the distinction between least and greatest fixed points $\mu_X \vec{K}$ and $\nu_X \vec{K}$ must be taken into account. This is already so at closed base types: similarity $\sim_{\mathbb{N}}$ and bisimilarity $\approx_{\mathbb{N}}$ are very different concepts. However, this difference is ignored in the definition of the type $\tau(C)$ of a c.r. predicate or

formula C. We therefore refine the definition of type $\tau(C)$ to *cotype* $\varphi(C)$. Using ideas from Gandy (1953, 1956) and Takeuti (1953) we can define a more appropriate concept of (pointwise) equality $\doteq_\varphi$, which is relative to a cotype φ. Extensionality Ext_φ relative to φ is defined by $(x \in \mathrm{Ext}_\varphi) := (x \doteq_\varphi x)$. We write $\doteq_C$ for $\doteq_{\varphi(C)}$ and Ext_C for $\mathrm{Ext}_{\varphi(C)}$.

3.3.1. Equality for closed base types. We take the algebra $\mathbb{Y}$ of binary trees as an example of a closed base type. In previous examples we have seen that the set of total binary trees can be represented by an inductive predicate $T_\mathbb{Y}$ (page 52), and that the set of cototal binary trees can be represented by a coinductive predicate ${}^{\mathrm{co}}T_\mathbb{Y}$ (page 55). 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 $\approx_\mathbb{Y}$. The introduction axioms for $\sim_\mathbb{Y}$ are

$$(\sim_\mathbb{Y})_0^+ \colon - \sim_\mathbb{Y} -,$$
$$(\sim_\mathbb{Y})_1^+ \colon \forall_{t_1,t_1'}(t_1 \sim_\mathbb{Y} t_1' \to \forall_{t_2,t_2'}(t_2 \sim_\mathbb{Y} t_2' \to \mathrm{C}t_1t_2 \sim_\mathbb{Y} \mathrm{C}t_1't_2'))$$

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

$$X(-,-) \to$$
$$\forall_{t_1,t_2}((t_1 \sim_\mathbb{Y} t_2 \land Xt_1t_2) \to \forall_{t_1',t_2'}((t_1' \sim_\mathbb{Y} t_2' \land Xt_1't_2') \to X(\mathrm{C}t_1t_2, \mathrm{C}t_1't_2'))) \to$$
$$\sim_\mathbb{Y} \subseteq X.$$

The elimination (or closure) axiom for $\approx_\mathbb{Y}$ is $\approx_\mathbb{Y}^-$:

$$\forall_{t,t'}(t \approx_\mathbb{Y} t' \to ((t \equiv -) \land (t' \equiv -)) \lor$$
$$\exists_{t_1,t_2,t_1',t_2'}(t_1 \approx_\mathbb{Y} t_1' \land t_2 \approx_\mathbb{Y} t_2' \land t \equiv \mathrm{C}t_1t_2 \land t' \equiv \mathrm{C}t_1't_2'))$$

and the introduction (or greatest-fixed-point or coinduction) axiom is $\approx_\mathbb{Y}^+$:

$$\forall_{t,t'}(Xtt' \to ((t \equiv -) \land (t' \equiv -)) \lor$$
$$\exists_{t_1,t_2,t_1',t_2'}((t_1 \approx_\mathbb{Y} t_1' \lor Xt_1t_1') \land (t_2 \approx_\mathbb{Y} t_2' \lor Xt_2t_2') \land$$
$$t \equiv \mathrm{C}t_1t_2 \land t' \equiv \mathrm{C}t_1't_2')) \to$$
$$X \subseteq {}^{\mathrm{co}}T_\mathbb{Y}.$$

As an instance of Lemma 3.2.1 we have $\sim_\mathbb{Y} \subseteq \approx_\mathbb{Y}$.

We now 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 be the fact that it holds in our intended model (cf. Lemma 2.1.7).

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

Lemma 3.3.1 (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 2.1.8 has been given in enough detail to make its formalization immediate. We need ${}^{\mathrm{co}}T^{\pm}_{\mathbb{Y}}$, $\approx^{\pm}_{\mathbb{Y}}$.

(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 $\sim_{\mathbb{Y}} \subseteq \approx_{\mathbb{Y}}$. □

COROLLARY 3.3.2.

(a) $\forall_t(t \sim_{\mathbb{Y}} t \leftrightarrow t \in T_{\mathbb{Y}})$.
(b) $\forall_t(t \approx_{\mathbb{Y}} t \leftrightarrow t \in {}^{\mathrm{co}}T_{\mathbb{Y}})$.

PROOF. Immediate from Lemma 3.3.1. □

COROLLARY 3.3.3.

(a) $\sim_{\mathbb{Y}}$ *is an equivalence relation on* $T_{\mathbb{Y}}$.
(b) $\approx_{\mathbb{Y}}$ *is an equivalence relation on* ${}^{\mathrm{co}}T_{\mathbb{Y}}$.

PROOF. Immediate from Lemma 3.3.1. □

REMARK. For closed base types like $\mathbb{Y}$ we can also relate $\sim_{\mathbb{Y}}$ to the binary boolean-valued function $=_{\mathbb{Y}} : \mathbb{Y} \to \mathbb{Y} \to \mathbb{B}$ defined in Section 2.2.1. One easily proves that

$$\forall_t(t \in T_{\mathbb{Y}} \to t = t),$$
$$\forall_t(t \in T_{\mathbb{Y}} \to \forall_{t'}(t' \in T_{\mathbb{Y}} \to t = t' \to t \sim_{\mathbb{Y}} t')).$$

Usage of $=_{\mathbb{Y}}$ has the advantage that proofs may become shorter, since we identify terms with a common reduct.

3.3.2. Equality at higher type levels. Using cotypes we refine the assignment given in Section 3.1 (page 47) of a type $\tau(C)$ to a predicate or formula C.

DEFINITION (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^c.

$$\begin{aligned}
\varphi(X^c) &:= \xi,\\
\varphi(\{\,\vec{x} \mid A\,\}) &:= \varphi(A),\\
\varphi(I(\vec{\rho}, \vec{P})) &:= \iota_I(\varphi(\vec{P}^c)),\\
\varphi({}^{\mathrm{co}}I(\vec{\rho}, \vec{P})) &:= \begin{cases} \iota_I(\varphi(\vec{P}^c)) & \text{if } \iota_I \text{ is a non-recursive algebra}\\ {}^{\mathrm{co}}\iota_I(\varphi(\vec{P}^c)) & \text{otherwise,}\end{cases}\\
\varphi(P\vec{t}) &:= \varphi(P),
\end{aligned}$$

$$\varphi(A \to B) := \begin{cases} \varphi(A) \to \varphi(B) & \text{if } A \text{ is c.r.} \\ \varphi(B) & \text{if } A \text{ is n.c.,} \end{cases}$$
$$\varphi(\forall_x A) := \varphi(A)$$

where $\vec{P}^c$ are the c.r. predicates among $\vec{P}$ and ι_I is the algebra associated with the predicate I.

In case of a non-recursive algebra ι_I the marking ${}^{\mathrm{co}}\iota_I$ is not necessary, because we not only have $I_\iota \subseteq {}^{\mathrm{co}}I_\iota$, but also the reverse inclusion ${}^{\mathrm{co}}I_\iota \subseteq I_\iota$.

Examples. $\varphi(\sim_{\mathbb{N}}) = \mathbb{N}$ and $\varphi(\approx_{\mathbb{N}}) = {}^{\mathrm{co}}\mathbb{N}$ are cotypes, and

$$\varphi(\sim_{\mathbb{L}}(\sim_{\mathbb{N}})) = \mathbb{L}(\mathbb{N}),$$
$$\varphi(\sim_{\mathbb{L}}(\approx_{\mathbb{N}})) = \mathbb{L}({}^{\mathrm{co}}\mathbb{N}),$$
$$\varphi(\approx_{\mathbb{L}}(\sim_{\mathbb{N}})) = {}^{\mathrm{co}}\mathbb{L}(\mathbb{N}),$$
$$\varphi(\approx_{\mathbb{L}}(\approx_{\mathbb{N}})) = {}^{\mathrm{co}}\mathbb{L}({}^{\mathrm{co}}\mathbb{N}).$$

Similarly, $\varphi(T_{\mathbb{N}}) = \mathbb{N}$ and $\varphi({}^{\mathrm{co}}T_{\mathbb{N}}) = {}^{\mathrm{co}}\mathbb{N}$ are cotypes, and

$$\varphi(T_{\mathbb{L}}(T_{\mathbb{N}})) = \mathbb{L}(\mathbb{N}),$$
$$\varphi(T_{\mathbb{L}}({}^{\mathrm{co}}T_{\mathbb{N}})) = \mathbb{L}({}^{\mathrm{co}}\mathbb{N}),$$
$$\varphi({}^{\mathrm{co}}T_{\mathbb{L}}(T_{\mathbb{N}})) = {}^{\mathrm{co}}\mathbb{L}(\mathbb{N}),$$
$$\varphi({}^{\mathrm{co}}T_{\mathbb{L}}({}^{\mathrm{co}}T_{\mathbb{N}})) = {}^{\mathrm{co}}\mathbb{L}({}^{\mathrm{co}}\mathbb{N}).$$

As a preparation for the definition of pointwise equality at higher type levels we generalize the examples of the predicate forms $\sim_{\mathbb{L}}$, $\approx_{\mathbb{L}}$, $T_{\mathbb{L}}$ and ${}^{\mathrm{co}}T_{\mathbb{L}}$ on page 46 from lists $\mathbb{L}$ to arbitrary algebra forms.

Definition (Similarity and bisimilarity). 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

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

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

Definition (Pointwise equality $\doteq_\varphi$ w.r.t. a cotype φ). By recursion on $\mathrm{lev}(\varphi)$ with a subordinate recursion on the height $|\varphi|$ we define *pointwise*

equality $\doteq_\varphi$ w.r.t. a cotype φ by

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

where the *height* $|\varphi|$ of a cotype φ is defined by $|\alpha| := 0$, $|\iota(\vec\varphi)| := |{}^{\mathrm{co}}\iota(\vec\varphi)| := 1 + \max|\vec\varphi|$ and $|\varphi\to\psi| := 1 + \max(|\varphi|, |\psi|)$.

Examples. For $\mathbb{Y}$ pointwise equality $\doteq_{\mathbb{Y}}$ is $\sim_{\mathbb{Y}}$, and pointwise equality $\doteq_{({}^{\mathrm{co}}\mathbb{Y})}$ is $\approx_{\mathbb{Y}}$. For $\mathbb{L}(\alpha)$ we need the binary predicate Y of arity (α,α) uniquely assigned to α. Then pointwise equality $\doteq_{\mathbb{L}(\alpha)}$ is $\sim_{\mathbb{L}}(Y)$, and pointwise equality $\doteq_{({}^{\mathrm{co}}\mathbb{L}(\alpha))}$ is $\approx_{\mathbb{L}}(Y)$.

Definition (Extensionality Ext_φ w.r.t. a cotype φ). Extensionality w.r.t. a cotype φ is defined by

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

Example (A non-extensional functional). Define f, g of type $\mathbb{N}\to\mathbb{N}$ by the computation rules $fn = 0$ and $g0 = 0$, $g(Sn) = gn$. Then $f\bot_{\mathbb{N}} = 0$ because of the computation rules for f. For $g\bot_{\mathbb{N}}$ no computation rule fits, but because of the inductive definition of $(\vec U, a) \in [\![\lambda_{\vec x} M]\!]$ in Section 2.3 $[\![g\bot_{\mathbb{N}}]\!]$ is the empty ideal $[\![\bot_{\mathbb{N}}]\!]$. 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 Corollary 3.3.2 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. Since extensionality (i.e., the diagonalization of $\doteq$) can be reasonably defined at higher types as well (again by diagonalization of $\doteq$ at higher types) this gives us a way to generalize totality to higher types: just take extensionality. The equivalence of Ext_φ and T_φ on closed base cotypes can be extended to closed cotypes of level 1:

Lemma 3.3.4. *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 Corollary 3.3.2 (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.

$$
\begin{aligned}
&f \in \mathrm{Ext}_{\varphi\to\psi}\\
&f \doteq_{\varphi\to\psi} f\\
&\forall_{x,y}(x \doteq_\varphi y \to fx \doteq_\psi fy)\\
&\forall_{x\in T_\varphi}(fx \doteq_\psi fx) && \text{by Corollary 3.3.2, since } \mathrm{lev}(\varphi)=0\\
&\forall_{x\in T_\varphi}(fx \in \mathrm{Ext}_\psi).
\end{aligned}
$$

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

For arbitrary closed cotypes the relation $\doteq_\varphi$ is a "partial equivalence relation", which means the following.

LEMMA 3.3.5. *For every closed cotype φ the relation $\doteq_\varphi$ is an equivalence relation on* 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 Corollary 3.3.3. 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 $fx \doteq_\psi hy$. □

LEMMA 3.3.6 (Compatibility of terms). *For every term $t(\vec x)$ with extensional constants and free variables among $\vec x$ we have*

$$\forall_{\vec x,\vec y}(\vec x \doteq_{\vec\chi} \vec y \to t(\vec x) \doteq_\varphi t(\vec y)).$$

PROOF. This is proved by induction on t. *Case* x. Immediate. *Case* c. By assumption $c \doteq_\varphi c$. *Case* $\lambda_x t(x,\vec x)$. Let $\vec x \doteq_{\vec\chi} \vec y$. The goal is $\lambda_x t(x,\vec x) \doteq_{\varphi\to\psi} \lambda_x t(x,\vec y)$, which by definition means

$$\forall_{x,y}(x \doteq_\varphi y \to t(x,\vec x) \doteq_\psi t(y,\vec y)).$$

Assume $x \doteq_\varphi y$. With $\vec x \doteq_{\vec\chi} \vec y$ the claim $t(x,\vec x) \doteq_\psi t(y,\vec y)$ holds by the IH. *Case* $t(\vec x)s(\vec x)$. Let $\vec x \doteq_{\vec\chi} \vec y$. By IH we have $t(\vec x) \doteq_{\varphi\to\psi} t(\vec y)$, i.e.,

$$\forall_{x,y}(x \doteq_\varphi y \to t(\vec x)x \doteq_\psi t(\vec y)y).$$

Again by IH we have $s(\vec x) \doteq_\varphi s(\vec y)$. Hence $t(\vec x)s(\vec x) \doteq_\psi t(\vec y)s(\vec y)$. □

LEMMA 3.3.7 (Extensionality of terms). *For every term $t(\vec{x}\,)$ with extensional constants and free variables among $\vec{x}$ we have*

$$\forall_{\vec{x}}(\vec{x} \in \mathrm{Ext}_{\vec{\chi}} \to t(\vec{x}\,) \in \mathrm{Ext}_{\varphi}).$$

PROOF. Let $t(\vec{x}\,)$ with free variables among $\vec{x}$ be given, and assume $\vec{x} \in \mathrm{Ext}_{\vec{\chi}}$. By Lemma 3.3.6 applied to $\vec{x}, \vec{x}$ we obtain $t(\vec{x}\,) \doteq_{\varphi} t(\vec{x}\,)$, hence $t(\vec{x}\,) \in \mathrm{Ext}_{\varphi}$. □

CHAPTER 4

Computational content of proofs

We have already mentioned that (co)inductive predicates can be declared as either computationally relevant (c.r.) or non-computational (n.c.). But what is the computational content in the c.r. case? We first address this question for (co)inductive predicates, and then extend it to arbitrary formulas. Next we study in what sense a proof of a c.r. formula A provides us with concrete computational content. This can be seen as a "witness" for the validity of A, or (in the sense of Kolmogorov (1932)) a "solution" to problem A.

Finally we take a step back and reflect on what we have done. We formally define what it means for a term to "realize" the c.r. formula A. We extract from a proof M of A a term $\mathrm{et}(M)$ and (again 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, formally $A \leftrightarrow \exists_z(z\ \mathbf{r}\ A)$, where $z\ \mathbf{r}\ A$ means "z realizes A".

4.1. Realizers

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 $\mathrm{C}_i\colon \kappa_i(\iota_I)$. 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

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

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

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

Example. As an easy 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 $K_1(X) := \forall_n(n \in X \to S(Sn) \in X)$. 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 $\mathrm{C}_0 := 0$ and $\mathrm{C}_1 := S$. Let ϑ be the substitution $\xi \mapsto \mathbb{N}$, $X^{\mathbf{r}} \mapsto \{\, n, m \mid Ynm \,\}$. Since $S\ \mathbf{r}\ K_1(X)$ is $\forall_{n,m}(X^{\mathbf{r}}nm \to X^{\mathbf{r}}(S(Sn), Sm))$ we obtain

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

We express Kolmogorov's view of formulas as problems by means of *invariance axioms*:

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

$$\mathrm{InvAll}_A\colon \forall_z(z\ \mathbf{r}\ A \to A). \tag{14}$$

$$\mathrm{InvEx}_A\colon A \to \exists_z(z\ \mathbf{r}\ A). \tag{15}$$

Realizers of totality and cototality predicates will be of special interest for us. Notice that the types $\tau(T_\iota)$ and $\tau({}^{\mathrm{co}}T_\iota)$ are both ι. Moreover we have

Lemma 4.1.1 (Realizers of totality). *For closed base types* ι *the following are equivalent.*

(a) $T_\iota^r xy$,
(b) $x \sim_\iota^{\mathrm{nc}} y$,
(c) $x \in T_\iota^{\mathrm{nc}} \wedge x \equiv y$.

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

(b) $\leftrightarrow$ (c). Use Lemma 3.3.1. □

LEMMA 4.1.2 (Realizers of cototality). *For closed base types ι the following are equivalent.*

(a) ${}^{\mathrm{co}}T_\iota^{\boldsymbol{r}} xy$,
(b) $x \approx_\iota^{\mathrm{nc}} y$,
(c) $x \in {}^{\mathrm{co}}T_\iota^{\mathrm{nc}} \wedge x \equiv y$.

PROOF. As an example we give the proof for $\mathbb{N}$. Since we have n.c. goals only, decorations are omitted.

(a) $\to$ (b). We use the greatest-fixed-point axiom for $\approx_{\mathbb{N}}$:

$$\forall_{n,m}(Xnm \to (n \equiv 0 \wedge m \equiv 0) \vee \exists_{n',m'}((n' \approx_{\mathbb{N}} m' \vee Xn'm') \wedge n \equiv Sn' \wedge m \equiv Sm')) \to X \subseteq \approx_{\mathbb{N}}$$

and apply it with ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}}$ for X. It suffices to prove the premise. Assume ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} nm$; the goal is

$$C := {}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} 00 \vee \exists_{n',m'}((n' \approx_{\mathbb{N}} m' \vee {}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} n'm') \wedge n \equiv Sn' \wedge m \equiv Sm').$$

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

$$(n \equiv 0 \wedge m \equiv 0) \vee \exists_{n',m'}({}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} n'm' \wedge n \equiv Sn' \wedge m \equiv Sm').$$

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 ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} 00$. But this follows from the greatest-fixed-point axiom for ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}}$ with competitor predicate $\{\, n, m \mid n \equiv 0 \wedge m \equiv 0 \,\}$.

Case 2. $\exists_{n',m'}({}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} n'm' \wedge n \equiv Sn' \wedge m \equiv Sm')$. Go for the r.h.s. of C.

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

$${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} := \nu_{X^{\mathbf{r}}}(X^{\mathbf{r}} 00,\ \forall_{n,m}(X^{\mathbf{r}} nm \to X^{\mathbf{r}}(Sn)(Sm))).$$

We need to show $m \approx_{\mathbb{N}} n \to {}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} mn$. To this end we use the greatest-fixed-point axiom for ${}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}}$:

$$\forall_{n,m}(Xnm \to X00 \vee \exists_{n',m'}(n', m' \in ({}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} \cup X) \wedge n \equiv Sn' \wedge m \equiv Sm')) \to X \subseteq {}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}}$$

and apply it with $\approx_{\mathbb{N}}$ for X. It suffices to prove the premise. Assume $n \approx_{\mathbb{N}} m$; the goal is

$$C := (0 \approx_{\mathbb{N}} 0) \vee \exists_{n',m'}((n', m' \in ({}^{\mathrm{co}}T_{\mathbb{N}}^{\mathbf{r}} \cup \approx_{\mathbb{N}}) \wedge n \equiv Sn' \wedge m \equiv Sm')).$$

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

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

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

(b) $\leftrightarrow$ (c). Use the Bisimilarity axiom and Lemma 3.3.1. $\square$

Next we study what our general definition says about realizers for the c.r. inductively defined decorated connectives.

LEMMA 4.1.3 (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 $(\mathrm{Ex}^{\mathbf{r}}_{\{x,z|z\mathbf{r}A\}})^-$. $\square$

Similarly we have

LEMMA 4.1.4 (Realizers for $\wedge$). *$z\ \mathbf{r}\ (A \wedge B)$ is equivalent to*

$z \equiv \langle \mathrm{lft}(z), \mathrm{rht}(z)\rangle \wedge (\mathrm{lft}(z)\ \mathbf{r}\ A) \wedge (\mathrm{rht}(z)\ \mathbf{r}\ B)$	*for A c.r. and B c.r.*
$(z\ \mathbf{r}\ A) \wedge B$	*for A c.r. and B n.c.*
$A \wedge (z\ \mathbf{r}\ B)$	*for A n.c. and B c.r.*

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

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

Now substituting $X^{\mathbf{r}}$ by $\{\, x \mid x\ \mathbf{r}\ A \,\}$ and $Y^{\mathbf{r}}$ by $\{\, y \mid y\ \mathbf{r}\ B \,\}$ in the introduction axiom 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 $\{\, z \mid z \equiv \langle \mathrm{lft}(z), \mathrm{rht}(z)\rangle \wedge (\mathrm{lft}(z)\ \mathbf{r}\ A) \wedge (\mathrm{rht}(z)\ \mathbf{r}\ B) \,\}$. Then 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 and we obtain

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

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

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

Now substituting $X^{\mathbf{r}}$ by $\{\, z \mid z\ \mathbf{r}\ A \,\}$ and Y^{nc} by B in the introduction axiom 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 4.1.5 (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 we consider the case A n.c. and B c.r. Recall $\mathrm{OrR}_{X^{\mathrm{nc}},Y^c} := \mu_Z(X^{\mathrm{nc}} \to Z, Y^c \to Z)$. Then

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

Now substituting X^{nc} by A and $Y^{\mathbf{r}}$ by $\{\, y \mid y\ \mathbf{r}\ B \,\}$ in the introduction axioms 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 two premises become provable and we obtain

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

4.2. Extracted terms, soundness

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 $\mathrm{et}(I_i^+) := \mathrm{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where both the constructor C_i and the recursion operator $\mathcal{R}$ refer to the algebra ι_I associated with I. For the closure and greatest-fixed-point axioms of ${}^{\mathrm{co}}I$ define $\mathrm{et}({}^{\mathrm{co}}I^-) := \mathrm{D}$ and $\mathrm{et}({}^{\mathrm{co}}I_i^+) := {}^{\mathrm{co}}\mathcal{R}$, where both the destructor 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.

From the Soundness Theorem 4.2.6 below it will follow that the term extracted from a closed $\mathbf{r}$-free proof of a c.r. formula A realizes A. As a preparation we first attend the axioms. Let I be an inductive predicate and ι_I its associated algebra. One can show that the extracted term of $I^{\pm}$, ${}^{\mathrm{co}}I^{\pm}$ realizes the respective axiom.

For the first two claims we only consider the inductive predicate $\sim_{\mathbb{L}}$ with $\mathbb{L}$ the algebra of lists of signed digits.

LEMMA 4.2.1. *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 $\sim^{\mathbf{r}}_{\mathbb{D}}(s_1, s_2, s)$. 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 $\sim^{\mathbf{r}}_{\mathbb{L}}(\ell_1, \ell_2, \ell)$. The goal then is

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

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

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

LEMMA 4.2.2. *The recursion operator* $\mathcal{R}^{\alpha}_{\mathbb{L}}$ *realizes the least-fixed-point axiom* $\sim^{-}_{\mathbb{L}}$.

PROOF. We equivalently rewrite $\sim^{-}_{\mathbb{L}}$ as $C :=$

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

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

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

We must show $\mathcal{R}^{\alpha}_{\mathbb{L}}\ \mathbf{r}\ C$. Pick ℓ_1, ℓ_2. The goal then is

$$\begin{aligned}
&\mathcal{R}^{\alpha}_{\mathbb{L}}\ \mathbf{r}\ (\ell_1 \sim_{\mathbb{L}} \ell_2 \to \\
&\quad X[][] \to \\
&\quad \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to X(s_1 :: \ell_1, s_2 :: \ell_2)) \to \\
&\quad X\ell_1\ell_2).
\end{aligned}$$

Pick ℓ with $\sim^{\mathbf{r}}_{\mathbb{L}}(\ell_1, \ell_2, \ell)$. Then the goal is

$$\begin{aligned}
&\mathcal{R}^{\alpha}_{\mathbb{L}}\ell\ \mathbf{r}\ (X[][] \to \\
&\quad \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to X\ell_1\ell_2 \to X(s_1 :: \ell_1, s_2 :: \ell_2)) \to \\
&\quad X\ell_1\ell_2).
\end{aligned}$$

Pick x with $X^{\mathbf{r}}[][]x$. Then the goal is

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

Pick f with

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

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

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

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

$$\begin{aligned}\forall_{\ell_1,\ell_2,\ell}(&\sim^{\mathbf{r}}_{\mathbb{L}}(\ell_1,\ell_2,\ell) \to\\ &Q[][][] \to\\ &\forall_{s_1,s_2,s,\ell_1,\ell_2,\ell}(\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \to \sim^{\mathbf{r}}_{\mathbb{L}}(\ell_1,\ell_2,\ell) \to Q\ell_1\ell_2\ell \to\\ &\qquad Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)) \to\\ &Q\ell_1\ell_2\ell).\end{aligned}$$

It suffices to prove the premises $Q[][][]$ and $\forall_{s_1,s_2,s,\ell_1,\ell_2,\ell}(\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \to \sim^{\mathbf{r}}_{\mathbb{L}}(\ell_1,\ell_2,\ell) \to Q\ell_1\ell_2\ell \to Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell))$. By a computation rule for $\mathcal{R}^{\alpha}_{\mathbb{L}}$ 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.,

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

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

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

But with $y := \mathcal{R}^{\alpha}_{\mathbb{L}}\ell xf$ this follows from what we have. □

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

Lemma 4.2.3. *The destructor* $\mathrm{D}_{\mathbb{S}}$ *realizes the closure axiom* $\approx^{-}_{\mathbb{S}}$.

Proof. Recall $\approx^{-}_{\mathbb{S}}$:

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

with cotype ${}^{\mathrm{co}}\mathbb{S} \to \mathbb{D} \times {}^{\mathrm{co}}\mathbb{S}$. The goal is $\mathrm{D}_{\mathbb{S}}\,\mathbf{r} \approx^{-}_{\mathbb{S}}$, which unfolds into

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

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

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

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

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

Take s_1, s_2, u_1', u_2'. It remains to show $\mathrm{D}_{\mathbb{S}}u\ \mathbf{r}\ (s_1 \sim_{\mathbb{D}} s_2 \land u_1' \approx_{\mathbb{S}} u_2')$. By the computation rule of $\mathrm{D}_{\mathbb{S}}$ we know $\mathrm{D}_{\mathbb{S}}u \equiv \mathrm{D}_{\mathbb{S}}(s :: u') \equiv \langle s, u' \rangle$. 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 4.2.4. *The corecursion operator* ${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}$ *realizes the greatest-fixed-point axiom* $\approx^{+}_{\mathbb{S}}$.

Proof. We equivalently rewrite $\approx^{+}_{\mathbb{S}}$ as $C :=$

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

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 must show that ${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}$ realizes C, or more formally ${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}\ \mathbf{r}\ C$. The goal then is

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

Pick u with $X^{\mathbf{r}}u_1u_2u$. Then the goal is

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

Pick f such that

$$\begin{aligned}
f\ \mathbf{r}\ \forall_{u_1,u_2}(Xu_1u_2 \to \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \land (u_1' \approx_{\mathbb{S}} u_2' \lor Xu_1'u_2') \land \\
u_1 \equiv s_1 :: u_1' \land u_2 \equiv s_2 :: u_2'))
\end{aligned}$$

i.e.,

$$\forall_{u_1,u_2,u}(X^{\mathbf{r}}u_1u_2u \to fu\ \mathbf{r}\ \exists_{s_1,s_2,u_1',u_2'}(s_1 \sim_{\mathbb{D}} s_2 \land (u_1' \approx_{\mathbb{S}} u_2' \lor Xu_1'u_2') \land u_1 \equiv s_1 :: u_1' \land u_2 \equiv s_2 :: u_2')).$$

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 the greatest-fixed-point axiom for $\approx^{\mathbf{r}}_{\mathbb{S}}$ in the form

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

with

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

It suffices to prove the closure property of Q. Let u_1, u_2, u and also u' be given such that

$$X^{\mathbf{r}}u_1u_2u' \land u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u'f.$$

We need to show

$$\begin{aligned}&\exists_{s_1,s_2,s,u_1',u_2',u'}(\\ &\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s) \land (\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \lor \exists_{u'}(X^{\mathbf{r}}u_1u_2u' \land u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u'f)) \land\\ &u_1 \equiv s_1 :: u_1' \land u_2 \equiv s_2 :: u_2' \land u \equiv s :: u').\end{aligned} \tag{16}$$

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

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

Then $fu' \equiv \langle s, w\rangle$ with $\sim^{\mathbf{r}}_{\mathbb{D}}(s_1,s_2,s)$ and $w\ \mathbf{r}\ (u_1' \approx_{\mathbb{S}} u_2' \lor Xu_1'u_2')$, 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

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

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. $\approx^{\mathbf{r}}_{\mathbb{S}}(u_1',u_2',u') \land w \equiv \mathrm{InL}(u')$ for some u'. Then (16) holds, since $u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u'f \equiv s :: u'$.

Case R. $X^{\mathbf{r}}u_1'u_2'u'' \land w \equiv \mathrm{InR}(u'')$ for some u''. Then again (16) holds with $u' := {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u''f$, since $u \equiv {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u'f \equiv s :: {}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}u''f \equiv s :: u'$. □

LEMMA 4.2.5 (Identities realize I^- for one-clause-nc I). *For one-clause-nc inductive predicates the elimination axiom with a c.r. competitor predicate is realized by the identity.*

PROOF. For $\vec{A}$ n.c. we have

$$(\lambda_z z)\ \mathbf{r}\ \forall_{\vec{x}}(I\vec{x} \to \forall_{\vec{y}}(\vec{A} \to X\vec{t}\,) \to X\vec{x}\,)$$
$$\forall_{\vec{x}}(I\vec{x} \to (\lambda_z z)\ \mathbf{r}\ (\forall_{\vec{y}}(\vec{A} \to X\vec{t}\,) \to X\vec{x}\,))$$
$$\forall_{\vec{x}}(I\vec{x} \to \forall_z(z\ \mathbf{r}\ \forall_{\vec{y}}(\vec{A} \to X\vec{t}\,) \to z\ \mathbf{r}\ X\vec{x}\,))$$
$$\forall_{\vec{x}}(I\vec{x} \to \forall_z(\forall_{\vec{y}}(\vec{A} \to z\ \mathbf{r}\ X\vec{t}\,) \to z\ \mathbf{r}\ X\vec{x}\,))$$

which is an instance of the same elimination axiom. □

THEOREM 4.2.6 (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*

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

from assumptions

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

PROOF. *Case* $u\colon A$. *Subcase* A c.r. Then $\mathrm{et}(u) = z_u$. *Subcase* A n.c. Immediate.

Case $c\colon A$. *Subcase* A c.r. The axioms have been treated above. *Subcase* A n.c. Immediate.

Case $(\lambda_{u^A} M^B)^{A\to B}$ with B c.r. We must derive $\mathrm{et}(\lambda_u M)\ \mathbf{r}\ (A \to B)$. To this end we distinguish subcases. *Subcase* A c.r. Then the goal

$$\forall_z(z\ \mathbf{r}\ A \to \mathrm{et}(M)(z)\ \mathbf{r}\ B)$$

follows from the induction hypothesis by $\to^+$ and $\forall^+$.

Subcase A n.c. Then the goal is

$$A \to \mathrm{et}(\lambda_u M)\ \mathbf{r}\ B.$$

Recall that $\mathrm{et}(\lambda_u M) = \mathrm{et}(M)$. By induction hypothesis we have a derivation of $\mathrm{et}(M)\ \mathbf{r}\ B$ from A, which is what we want.

Case $(\lambda_{u^A} M^B)^{A\to B}$ with B n.c. We need a derivation of $A \to B$.

Subcase A c.r. 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.

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

Subcase A n.c. By induction hypothesis we have a derivation of B from A, which is what we want.

Case $(M^{A\to B}N^A)^B$ with B c.r. We need a derivation of $\mathrm{et}(MN)\ \mathbf{r}\ B$. To this end we distinguish subcases. *Subcase* A c.r. Then $\mathrm{et}(MN) = \mathrm{et}(M)\mathrm{et}(N)$. By induction hypothesis we have derivations of $\mathrm{et}(M)\ \mathbf{r}\ (A \to B)$ and hence of

$$\forall_z(z\ \mathbf{r}\ A\ \to\ \mathrm{et}(M)z\ \mathbf{r}\ B)$$

and of $\mathrm{et}(N)\ \mathbf{r}\ A$. This gives the claim. *Subcase* A n.c. Then $\mathrm{et}(MN) = \mathrm{et}(M)$. By induction hypothesis we have derivations of $\mathrm{et}(M)\ \mathbf{r}\ (A \to B)$ and hence of

$$A\ \to\ \mathrm{et}(M)\ \mathbf{r}\ B$$

and of A. Applying the former to the latter gives $\mathrm{et}(M)\ \mathbf{r}\ B$.

Case $(M^{A\to B}N^A)^B$ with B n.c. The goal is to find a derivation of B. *Subcase* A c.r. By induction hypothesis we have derivations of $A \to B$ and of $\mathrm{et}(N)\ \mathbf{r}\ A$. Now using the invariance axiom $\forall_z(z\ \mathbf{r}\ A \to A)$ we obtain the required derivation of B by $\to^-$ from the derivation of $A \to B$ and

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

Subcase A n.c. By induction hypothesis we have derivations of $A \to B$ and of A, hence also a derivation of B.

Case $(\lambda_x M^A)^{\forall_x A}$ with $\forall_x A$ c.r. We need a derivation of $\mathrm{et}(\lambda_x M)\ \mathbf{r}\ \forall_x A$. By definition $\mathrm{et}(\lambda_x M) = \mathrm{et}(M)$. Hence we must derive

$$\mathrm{et}(M)\ \mathbf{r}\ \forall_x A, \quad \text{which is} \quad \forall_x(\mathrm{et}(M)\ \mathbf{r}\ A).$$

This follows from the induction hypothesis.

Case $(\lambda_x M^A)^{\forall_x A}$ with $\forall_x A$ n.c. By induction hypothesis we have a derivation of A. Apply $\forall^+$.

Case $(M^{\forall_x A(x)}t)^{A(t)}$ with $A(t)$ c.r. We must derive $\mathrm{et}(Mt)\ \mathbf{r}\ A(t)$. By definition $\mathrm{et}(Mt) = \mathrm{et}(M)$, and by induction hypothesis we can derive

$$\mathrm{et}(M)\ \mathbf{r}\ \forall_x A(x), \quad \text{which is} \quad \forall_x(\mathrm{et}(M)\ \mathbf{r}\ A(x)).$$

Case $(M^{\forall_x A(x)}t)^{A(t)}$ with $A(t)$ n.c. By induction hypothesis we have a derivation of $\forall_x A(x)$. Apply $\forall^-$. □

4.3. 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^+,
- $\mathcal{R}^\alpha_{\iota_I}$ 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
- ${}^{\mathrm{co}}\mathcal{R}^\alpha_{\iota_I}$ is extensional w.r.t. the greatest-fixed-point axiom ${}^{\mathrm{co}}I^+$.

Since the term $\mathrm{et}(M)$ extracted from a closed proof M of a c.r. formula A is built from these constants by abstraction and application, by Lemma 3.3.7 on extensionality of terms we can conclude that $\mathrm{et}(M)$ is extensional w.r.t. the cotype of A.

We prove the claim above 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

$$(\sim_{\mathbb{L}})^+_0 \colon [] \sim_{\mathbb{L}} [],$$
$$(\sim_{\mathbb{L}})^+_1 \colon \forall_{s_1,s_2,\ell_1,\ell_2}(s_1 \sim_{\mathbb{D}} s_2 \to \ell_1 \sim_{\mathbb{L}} \ell_2 \to s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2)$$

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 4.3.1. *The constructors of $\mathbb{L}$ are extensional w.r.t. the clauses of $\sim_{\mathbb{L}}$.*

Proof. We only consider the second constructor C. The goal is to show that 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 4.3.2. *$\mathcal{R}^\alpha_{\mathbb{L}}$ is extensional w.r.t. the least-fixed-point axiom $\sim^-_{\mathbb{L}}$.*

PROOF. We equivalently rewrite $\sim^-_{\mathbb{L}}$ as $C :=$

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

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

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

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

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

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 by definition of $\doteq$ from $f_1 \doteq f_2$, $s_1 \sim s_2$, $\ell_1 \sim \ell_2$ and $y_1 \doteq_\alpha y_2$. □

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

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

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

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

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

LEMMA 4.3.3. *The destructor* $\mathrm{D}_{\mathbb{S}}$ *is extensional w.r.t. the closure axiom* $\approx^-_{\mathbb{S}}$.

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

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

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

$$s_1 \sim_{\mathbb{D}} s_2 \land u_1' \approx_{\mathbb{S}} u_2' \land u_1 \equiv s_1 :: u_1' \land u_2 \equiv s_2 :: u_2'.$$

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

Lemma 4.3.4. ${}^{\mathrm{co}}\mathcal{R}^{\alpha}_{\mathbb{S}}$ *is extensional w.r.t. the greatest-fixed-point axiom* $\approx^+_{\mathbb{S}}$.

Proof. We equivalently rewrite $\approx^+_{\mathbb{S}}$ as

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

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

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

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

$$X := \{\, u_1, u_2 \mid \exists_{y_1,y_2}(u_1 \equiv {}^{\mathrm{co}}\mathcal{R} y_1 f_1 \land u_2 \equiv {}^{\mathrm{co}}\mathcal{R} y_2 f_2 \land y_1 \doteq_{\alpha} y_2) \,\}.$$

This means that we have to show

$$\begin{aligned}
&\exists_{s_1,s_2,u_1',u_2'}(\\
&s_1 \sim_{\mathbb{D}} s_2 \land (u_1' \approx_{\mathbb{S}} u_2' \lor \exists_{y_1,y_2}(u_1' \equiv {}^{\mathrm{co}}\mathcal{R} y_1 f_1 \land u_2' \equiv {}^{\mathrm{co}}\mathcal{R} y_2 f_2 \land y_1 \doteq_{\alpha} y_2)) \land \\
&u_1 \equiv s_1 :: u_1' \land u_2 \equiv s_2 :: u_2').
\end{aligned}$$

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

$$f_1 x_1 \equiv \langle s_1, a_1 \rangle \land f_2 x_2 \equiv \langle s_2, a_2 \rangle \land s_1 \sim_{\mathbb{D}} s_2 \land a_1 \sim_{({}^{\mathrm{co}}\mathbb{S}+\alpha)} a_2,$$

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

$$\begin{aligned}
&(a_1 \equiv \mathrm{InL}(u_1') \land a_2 \equiv \mathrm{InL}(u_2') \land u_1' \approx_{\mathbb{S}} u_2') \lor \\
&(a_1 \equiv \mathrm{InR}(x_1') \land a_2 \equiv \mathrm{InR}(x_2') \land x_1' \doteq_{\alpha} x_2').
\end{aligned}$$

We argue by cases on this disjunction. Recall

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

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 4.3.5 (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 4.3.1, 4.3.2, 4.3.3 and 4.3.4.

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 an assumption $z_u' \doteq_A z_u''$. 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 defined to be

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

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.

The cases $\lambda_x M$ and Mt are obvious since for them the extracted term does not change. □

COROLLARY 4.3.6 (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* $i = 1, \ldots, n$*.*

PROOF. In the proof constructed in Theorem 4.3.5 we only need to substitute z'_{u_i}, z''_{u_i} by z_{u_i}. □

CHAPTER 5

Applications

Real numbers in the exact (as opposed to floating-point) sense can be given in different formats, for instance

(i) as Cauchy sequences (of rationals, with a Cauchy modulus), or else
(ii) as infinite sequences ("streams") of "signed digits". Intuitively, the stream $d_0, d_1, d_2 \ldots$ represents the real number

$$\sum_{i=0}^{\infty} \frac{d_i}{2^{i+1}} \qquad \text{with } d_i \in \{1, 0, -1\}.$$

We are interested in formally verified algorithms on real numbers in an arbitrary format. To this end we generate such algorithms as extracted terms $\mathrm{et}(M)$ of formal existence proofs M. Recall that in our setting universal quantifiers are ignored by realizability. This makes it possible to carry out proofs using a standard representation of reals x (e.g., Cauchy sequences with modulus), and let computational content only arise by relativising x to appropriate c.r. predicates. An example is the relativization of x to a predicate ${}^{\mathrm{co}}I$ coinductively defined in such a way that the computational content of $x \in {}^{\mathrm{co}}I$ is a stream representing x. The required verification is provided by a formal soundness proof of the realizability interpretation.

5.1. Exact real arithmetic

5.1.1. Cauchy sequences, equality. We shall view a real as a Cauchy sequence of rationals with a separately given modulus.

Definition 5.1.1. A real number x is a pair $((a_n)_{n\in\mathbb{N}}, M)$ with $a_n \in \mathbb{Q}$ and $M\colon \mathbb{Z}^+ \to \mathbb{N}$ such that $(a_n)_n$ is a *Cauchy sequence* with modulus M, that is

$$|a_n - a_m| \le \frac{1}{2^p} \quad \text{for } n, m \ge M(p)$$

and M is weakly increasing (that is $M(p) \le M(q)$ for $p \le q$). M is called *Cauchy modulus* of x.

We shall loosely speak of a real $(a_n)_n$ if the Cauchy modulus M is clear from the context or inessential. Every rational a is tacitly understood as the

real represented by the constant sequence $a_n = a$ with the constant modulus $M(p) = 0$.

DEFINITION 5.1.2. Two reals $x := ((a_n)_n, M)$, $y := ((b_n)_n, N)$ are called *equivalent* (or *equal* and written $x = y$, if the context makes clear what is meant), if

$$|a_{M(p+1)} - b_{N(p+1)}| \leq \frac{1}{2^p} \quad \text{for all } p \in \mathbb{Z}^+.$$

We want to show that this is an equivalence relation. Reflexivity and symmetry are clear. For transitivity we use the following lemma:

LEMMA 5.1.3 (RealEqChar). *For reals $x := ((a_n)_n, M)$, $y := ((b_n)_n, N)$ the following are equivalent:*

(a) $x = y$;
(b) $\forall_p \exists_{n_0} \forall_{n \geq n_0} (|a_n - b_n| \leq \frac{1}{2^p})$.

PROOF. (a) implies (b). For $n \geq M(p+2), N(p+2)$ we have

$$\begin{aligned} |a_n - b_n| &\leq |a_n - a_{M(p+2)}| + |a_{M(p+2)} - b_{N(p+2)}| + |b_{N(p+2)} - b_n| \\ &\leq \frac{1}{2^{p+2}} + \frac{1}{2^{p+1}} + \frac{1}{2^{p+2}}. \end{aligned}$$

(b) implies (a). Let $q \in \mathbb{Z}^+$, and $n \geq n_0, M(p+1), N(p+1)$ with n_0 provided for q by (b). Then

$$\begin{aligned} |a_{M(p+1)} - b_{N(p+1)}| &\leq |a_{M(p+1)} - a_n| + |a_n - b_n| + |b_n - b_{N(p+1)}| \\ &\leq \frac{1}{2^{p+1}} + \frac{1}{2^q} + \frac{1}{2^{p+1}}. \end{aligned}$$

The claim follows, because this holds for every $q \in \mathbb{Z}^+$. □

REMARK 5.1.4 (RealSeqEqToEq). An immediate consequence is that any two reals with the same Cauchy sequence (but possibly different moduli) are equal.

LEMMA 5.1.5 (RealEqTrans). *Equality between reals is transitive.*

PROOF. Let $(a_n)_n$, $(b_n)_n$, $(c_n)_n$ be the Cauchy sequences for x, y, z. Assume $x = y$, $y = z$ and pick n_1, n_2 for $p + 1$ according to the lemma above. Then $|a_n - c_n| \leq |a_n - b_n| + |b_n - c_n| \leq \frac{1}{2^{p+1}} + \frac{1}{2^{p+1}}$ for $n \geq n_1, n_2$. □

5.1.2. The Archimedian property. For every function on the reals we certainly want compatibility with equality. This however is not always the case; here is an important example.

LEMMA 5.1.6 (RealBound). *For every real $x := ((a_n)_n, M)$ we can find p_x such that $|a_n| \leq 2^{p_x}$ for all n.*

PROOF. Let $n_0 := M(1)$ and p_x be such that $\max\{\, |a_n| \mid n \le n_0 \,\} + \frac{1}{2} \le 2^{p_x}$. Then $|a_n| \le 2^{p_x}$ for all n. □

Clearly this assignment of p_x to x is not compatible with equality.

5.1.3. Nonnegative and positive reals. A real $x := ((a_n)_n, M)$ is called *nonnegative* (written $x \in \mathbb{R}^{0+}$) if

$$-\frac{1}{2^p} \le a_{M(p)} \quad \text{for all } p \in \mathbb{Z}^+.$$

It is *p-positive* (written $x \in_p \mathbb{R}^+$, or $x \in \mathbb{R}^+$ if p is not needed) if

$$\frac{1}{2^p} \le a_{M(p+1)}.$$

We want to show that both properties are compatible with equality. First we prove a useful characterization of nonnegative reals.

LEMMA 5.1.7 (RealNNegChar). *For a real $x := ((a_n)_n, M)$ the following are equivalent:*

(a) $x \in \mathbb{R}^{0+}$;
(b) $\forall_p \exists_{n_0} \forall_{n \ge n_0} (-\frac{1}{2^p} \le a_n)$.

PROOF. (a) implies (b). For $n \ge M(p+1)$ we have

$$\begin{aligned} -\frac{1}{2^p} &\le -\frac{1}{2^{p+1}} + a_{M(p+1)} \\ &= -\frac{1}{2^{p+1}} + (a_{M(p+1)} - a_n) + a_n \\ &\le -\frac{1}{2^{p+1}} + \frac{1}{2^{p+1}} + a_n. \end{aligned}$$

(b) implies (a). Let $q \in \mathbb{Z}^+$ and $n \ge n_0, M(p)$ with n_0 provided by (b) (for q). Then

$$\begin{aligned} -\frac{1}{2^p} - \frac{1}{2^q} &\le -\frac{1}{2^p} + a_n \\ &= -\frac{1}{2^p} + (a_n - a_{M(p)}) + a_{M(p)} \\ &\le -\frac{1}{2^p} + \frac{1}{2^p} + a_{M(p)}. \end{aligned}$$

The claim follows, because this holds for every q. □

LEMMA 5.1.8 (RealNNegCompat). *If $x \in \mathbb{R}^{0+}$ and $x = y$, then $y \in \mathbb{R}^{0+}$.*

PROOF. Let $x := ((a_n)_n, M)$ and $y := ((b_n)_n, N)$. Assume $x \in \mathbb{R}^{0+}$ and $x = y$, and let p be given. Pick n_0 according to the lemma above

and n_1 according to the characterization of equality of reals in Lemma 5.1.3 (RealEqChar) (both for $p+1$). Then for $n \geq n_0, n_1$

$$-\frac{1}{2^p} \leq -\frac{1}{2^{p+1}} + a_n \leq (b_n - a_n) + a_n.$$

Hence $y \in \mathbb{R}^{0+}$ by definition. □

LEMMA 5.1.9 (RealPosChar). *For a real $x := ((a_n)_n, M)$ with $x \in_p \mathbb{R}^+$ we have*

$$\frac{1}{2^{p+1}} \leq a_n \quad \textit{for } M(p+1) \leq n.$$

Conversely, from $\forall_{n \geq n_0}(\frac{1}{2^q} \leq a_n)$ we can infer $x \in_{q+1} \mathbb{R}^+$.

PROOF. Assume $x \in_p \mathbb{R}^+$, that is $\frac{1}{2^p} \leq a_{M(p+1)}$. Then

$$\frac{1}{2^{p+1}} \leq -\frac{1}{2^{p+1}} + a_{M(p+1)} = -\frac{1}{2^{p+1}} + (a_{M(p+1)} - a_n) + a_n \leq a_n$$

for $M(p+1) \leq n$. Conversely,

$$\begin{aligned} \frac{1}{2^{q+1}} &< -\frac{1}{2^{q+2}} + \frac{1}{2^q} \\ &\leq -\frac{1}{2^{q+2}} + a_n && \text{for } n_0 \leq n \\ &\leq (a_{M(q+2)} - a_n) + a_n && \text{for } M(q+2) \leq n. \end{aligned}$$

Hence $x \in_{q+1} \mathbb{R}^+$. □

Positivity is compatible with equality, but only up to a shift of p:

LEMMA 5.1.10 (RealPosCompat). *If $x \in_p \mathbb{R}^+$ and $x = y$, then $y \in_{p+2} \mathbb{R}^+$.*

PROOF. Let $x := ((a_n)_n, M)$ and $y := ((b_n)_n, N)$. Assume $x = y$ and $x \in_p \mathbb{R}^+$, that is $\frac{1}{2^p} \leq a_{M(p+1)}$. The goal is $\frac{1}{2^{p+2}} \leq b_{N(p+3)}$. We have

$$\frac{1}{2^{p+2}} = \frac{1}{2^{p+1}} - \frac{1}{2^{p+2}} \leq a_{M(p+3)} + (b_{N(p+3)} - a_{M(p+3)})$$

using Lemma 5.1.9 (RealPosChar) with the monotonicity of M, and the definition of $x = y$. □

5.1.4. Arithmetical functions. Given real numbers $x := ((a_n)_n, M)$ and $y := ((b_n)_n, N)$, we define $x + y$, $-x$, $|x|$, $x \cdot y$, and $\frac{1}{x}$ (the latter only provided that $|x| \in_q \mathbb{R}^+$) as represented by the respective sequence (c_n) of

rationals with modulus L:

	c_n	$L(p)$
$x + y$	$a_n + b_n$	$\max\big(M(p+1), N(p+1)\big)$
$-x$	$-a_n$	$M(p)$
$\lvert x\rvert$	$\lvert a_n\rvert$	$M(p)$
$x \cdot y$	$a_n \cdot b_n$	$\max\big(M(p+1+p_y), N(p+1+p_x)\big)$
$\frac{1}{x}$ for $\lvert x\rvert \in_q \mathbb{R}^+$	$\begin{cases} \frac{1}{a_n} & \text{if } a_n \neq 0 \\ 0 & \text{if } a_n = 0 \end{cases}$	$M(2(q+1)+p)$

where 2^{p_x} is the upper bound provided by Lemma 5.1.6 (RealBound).

LEMMA 5.1.11. *For reals x, y also $x+y$, $-x$, $|x|$, $x \cdot y$ and (provided that $|x| \in_q \mathbb{R}^+$) also $1/x$ are reals.*

PROOF. We restrict ourselves to the cases $x \cdot y$ and $1/x$.

$$\begin{aligned} |a_n b_n - a_m b_m| &= |a_n(b_n - b_m) + (a_n - a_m)b_m| \\ &\leq |b_n - b_m| \cdot |a_n| + |a_n - a_m| \cdot |b_m| \\ &\leq |b_n - b_m| \cdot 2^{p_x} + |a_n - a_m| \cdot 2^{p_y} \leq \frac{1}{2^p} \end{aligned}$$

for $n, m \geq \max\big(M(p+1+p_y), N(p+1+p_x)\big)$.

For $1/x$ assume $|x| \in_q \mathbb{R}^+$. Then by the (proof of our) characterization of positivity in Lemma 5.1.9 (RealPosChar), $\frac{1}{2^{q+1}} \leq |a_n|$ for $n \geq M(q+1)$. Hence

$$\begin{aligned} \Big|\frac{1}{a_n} - \frac{1}{a_m}\Big| &= \frac{|a_m - a_n|}{|a_n a_m|} \\ &\leq 2^{2(q+1)}|a_m - a_n| \quad \text{for } n, m \geq M(q+1) \\ &\leq \frac{1}{2^p} \quad \text{for } n, m \geq M(2(q+1)+p). \end{aligned}$$

The claim now follows from the assumption that M is weakly increasing. □

LEMMA 5.1.12. *For reals x, y, z*

$$\begin{aligned} x + (y + z) &= (x + y) + z & x \cdot (y \cdot z) &= (x \cdot y) \cdot z \\ x + 0 &= x & x \cdot 1 &= x \\ x + (-x) &= 0 & 0 < |x| \rightarrow x \cdot \frac{1}{x} &= 1 \\ x + y &= y + x & x \cdot y &= y \cdot x \end{aligned}$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

PROOF. For $0 < |x| \to x \cdot \frac{1}{x} = 1$ the Cauchy sequences are finally the same, which suffices. In all other cases both the Cauchy sequences and the moduli are the same, hence both sides are actually identical. □

LEMMA 5.1.13. *The functions $x + y$, $-x$, $|x|$, $x \cdot y$ and (provided that $|x| \in_q \mathbb{R}^+$) also $1/x$ are compatible with equality.*

PROOF. Routine. For instance in case $x + y$ because of the commutativity of $+$ it suffices to prove $x = y \to x + z = y + z$. But this follows immediately from Lemma 5.1.3 (RealEqChar): the n_0 for the conclusion can be the same as for the premise. □

LEMMA 5.1.14. *For reals x, y from $x \cdot y = 1$ we can infer $0 < |x|$.*

PROOF. Pick p such that $|b_n| \leq 2^p$ for all n. Pick n_0 such that $n_0 \leq n$ implies $\frac{1}{2} \leq a_n \cdot b_n$. Then $\frac{1}{2} \leq |a_n| \cdot 2^p$ for $n_0 \leq n$, and hence $\frac{1}{2^{p+1}} \leq |a_n|$. □

LEMMA 5.1.15. *For reals x, y,*

(a) $x, y \in \mathbb{R}^{0+} \to x + y, x \cdot y \in \mathbb{R}^{0+}$,
(b) $x, y \in \mathbb{R}^{+} \to x + y, x \cdot y \in \mathbb{R}^{+}$,
(c) $x \in \mathbb{R}^{0+} \to -x \in \mathbb{R}^{0+} \to x = 0$.

PROOF. (a), (b). Routine. (c). Let p be given. Pick n_0 such that $-\frac{1}{2^p} \leq a_n$ and $-\frac{1}{2^p} \leq -a_n$ for $n \geq n_0$. Then $|a_n| \leq \frac{1}{2^p}$. □

5.1.5. Comparison of reals. We write $x \leq y$ for $y - x \in \mathbb{R}^{0+}$ and $x < y$ for $y - x \in \mathbb{R}^{+}$. Unwinding the definitions yields that $x \leq y$ is to say that for every p, $a_{L(p)} \leq b_{L(p)} + \frac{1}{2^p}$ with $L(p) := \max(M(p), N(p))$, or equivalently (using Lemma 5.1.7 (RealNNegChar)) that for every p there exists n_0 such that $a_n \leq b_n + \frac{1}{2^p}$ for all $n \geq n_0$. Furthermore, $x < y$ is a shorthand for the presence of p with $a_{L(p+1)} + \frac{1}{2^p} \leq b_{L(p+1)}$ with L the maximum of M and N, or equivalently (using Lemma 5.1.9 (RealPosChar)) for the presence of p, q with $a_n + \frac{1}{2^p} \leq b_n$ for all $n \geq q$; we then write $x <_p y$ (or $x <_{p,q} y$) whenever we want to call these witnesses.

LEMMA 5.1.16 (RealApprox). $\forall_{x,p} \exists_a (|a - x| \leq \frac{1}{2^p})$.

PROOF. Let $x = ((a_n), M)$. Given p, pick $a_{M(p)}$. We show $|a_{M(p)} - x| \leq \frac{1}{2^p}$, that is $|a_{M(p)} - a_{M(q)}| \leq \frac{1}{2^p} + \frac{1}{2^q}$ for every q. But this follows from

$$|a_{M(p)} - a_{M(q)}| \leq |a_{M(p)} - a_{M(p+q)}| + |a_{M(p+q)} - a_{M(q)}| \leq \frac{1}{2^p} + \frac{1}{2^q}. \quad \square$$

LEMMA 5.1.17. *For reals x, y, z,*

$$x \le x \qquad x \not< x$$
$$x \le y \to y \le x \to x = y \qquad x < y \to y < z \to x < z$$
$$x \le y \to y \le z \to x \le z \qquad x < y \to x + z < y + z$$
$$x \le y \to x + z \le y + z \qquad x < y \to 0 < z \to x \cdot z < y \cdot z$$
$$x \le y \to 0 \le z \to x \cdot z \le y \cdot z$$

PROOF. From Section 5.1.4. □

Here we have left out information on witnesses p for the statements proving a $<$-formula. Such estimates can easily be given explicitely. Here are two examples.

LEMMA 5.1.18 (RealPosPlus). $0 \le x \to 0 <_p y \to 0 <_{p+3} x + y$.

PROOF. From $0 \le x$ we have $\forall_q \exists_{n_0} \forall_{n \ge n_0} (-\frac{1}{2^q} \le a_n)$. From $0 <_p y$ we have some n_1 such that $\forall_{n \ge n_1} (\frac{1}{2^{p+1}} \le b_n)$. Pick n_0 for $p+2$. Then $n_0, n_1 \le n$ implies $0 \le a_n + \frac{1}{2^{p+2}}$ and $\frac{1}{2^{p+2}} \le b_n - \frac{1}{2^{p+2}}$, hence $\frac{1}{2^{p+2}} \le a_n + b_n$. Now Lemma 5.1.9 (RealPosChar) gives $0 <_{p+3} x + y$. □

LEMMA 5.1.19. $x \le y \to y <_p z \to x <_{p+5} z$.

PROOF. This follows from Lemma 5.1.18 (RealPosPlus). □

As is to be expected in view of the existential and universal character of the predicates $<$ and $\le$ on the reals, we have:

LEMMA 5.1.20 (LeIsNotGt). $x \le y \leftrightarrow y \not< x$.

PROOF. $\to$. Assume $x \le y$ and $y < x$. By Lemma 5.1.19 we obtain $x < x$, a contradiction.

$\leftarrow$. It clearly suffices to show $0 \not< z \to z \le 0$, for a real z given by $(c_n)_n$. Assume $0 \not< z$. We must show $\forall_p \exists_{n_0} \forall_{n \ge n_0} (c_n \le \frac{1}{2^p})$. Let p be given. By assumption $0 \not< z$, hence $\neg \exists_q (\frac{1}{2^q} \le c_{M(q+1)})$. For $q := p + 1$ this implies $c_{M(p+2)} < \frac{1}{2^{p+1}}$, hence $c_n \le c_{M(p+2)} + \frac{1}{2^{p+2}} < \frac{1}{2^p}$ for $M(p+2) \le n$. □

Constructively, we cannot compare two reals, but we can compare every real with a nontrivial interval.

LEMMA 5.1.21 (ApproxSplit). *Let x, y, z be given and assume $x < y$. Then either $z \le y$ or $x \le z$.*

PROOF. Let $x := ((a_n)_n, M)$, $y := ((b_n)_n, N)$, $z := ((c_n)_n, L)$. Assume $x <_p y$, that is (by definition) $\frac{1}{2^p} \le b_n - a_n$ for $n := \max(M(p+2), N(p+2))$. Let $m := \max(n, L(p+2))$.

Case $c_m \leq \frac{a_n+b_n}{2}$. We show $z \leq y$. It suffices to prove $c_l \leq b_l$ for $l \geq m$. This follows from

$$c_l \leq c_m + \frac{1}{2^{p+2}} \leq \frac{a_n+b_n}{2} + \frac{b_n-a_n}{4} = b_n - \frac{b_n-a_n}{4} \leq b_n - \frac{1}{2^{p+2}} \leq b_l.$$

Case $c_m \not\leq \frac{a_n+b_n}{2}$. We show $x \leq z$. This follows from $a_l \leq c_l$ for $l \geq m$:

$$a_l \leq a_n + \frac{1}{2^{p+2}} \leq a_n + \frac{b_n-a_n}{4} \leq \frac{a_n+b_n}{2} - \frac{b_n-a_n}{4} \leq c_m - \frac{1}{2^{p+2}} \leq c_l. \quad \square$$

Notice that the boolean object determining whether $z \leq y$ or $x \leq z$ depends on the representation of x, y and z. In particular this assignment is *not* compatible with our equality relation.

One might think that the non-available comparison of two reals could be circumvented by using a maximum function. Indeed, such a function can easily be defined (component-wise), and it has the expected properties $x, y \leq \max(x, y)$ and $x, y \leq z \to \max(x, y) \leq z$. But what is missing is the knowledge that $\max(x, y)$ equals one of its arguments, i.e., we do not have $\max(x, y) = x \vee \max(x, y) = y$.

However, in many cases it is sufficient to pick the up to ε largest real out of finitely many given ones. This is indeed possible. We give the proof for two reals; it can be easily generalized.

LEMMA 5.1.22 (Maximum of two reals). *Let* $x := ((a_n)_n, M)$ *and* $y := ((b_n)_n, N)$ *be reals, and* $p \in \mathbb{Z}^+$. *Then either* $x \leq y + \frac{1}{2^p}$ *or else* $y \leq x + \frac{1}{2^p}$.

PROOF. Let $m := \max\big(M(p+1), N(p+1)\big)$.

Case $a_m \leq b_m$. Then for $m \leq n$

$$a_n \leq a_m + \frac{1}{2^{p+1}} \leq b_m + \frac{1}{2^{p+1}} \leq b_n + \frac{1}{2^p}.$$

This holds for all $n \geq m$, therefore $x \leq y + \frac{1}{2^p}$.

Case $b_m < a_m$. Then for $m \leq n$

$$b_n \leq b_m + \frac{1}{2^{p+1}} < a_m + \frac{1}{2^{p+1}} \leq a_n + \frac{1}{2^p}.$$

This holds for all $n \geq m$, therefore $y \leq x + \frac{1}{2^p}$. $\square$

5.2. Algorithms on stream-represented real numbers

5.2.1. The predicates I and ${}^{\mathrm{co}}I$. We model infinite sequences of signed digits (streams) as objects in the algebra $\mathbb{S}(\mathbb{D}) := \mu_\xi(\mathrm{C}\colon \mathbb{D} \to \xi \to \xi)$, where $\mathbb{D} := \mu_\xi(\mathrm{SdR}\colon \xi, \mathrm{SdM}\colon \xi, \mathrm{SdL}\colon \xi)$ is a 3-element variant of the booleans $\mathbb{B}$. Such streams will appear as realizers of an inductive predicate I defined by the single clause

$$\forall_{d,x',x}(d \in \mathrm{Sd} \to x' \in I \to x = \frac{x'+d}{2} \to x \in I). \tag{17}$$

Here (and later) x ranges over real numbers and d over integers. Sd is a (formally inductive) predicate expressing that its integer argument d is a signed digit, i.e., $|d| \le 1$. We have chosen (17) rather than the simpler

$$\forall_{d,x}(d \in \mathrm{Sd} \to x \in I \to \frac{x+d}{2} \in I), \tag{18}$$

since we want I to be compatible with the defined equality $=$ on real numbers

$$\forall_{x,y}(x = y \to x \in I \to y \in I), \tag{19}$$

which easily follows from (17) (with reflexivity, symmetry and transitivity of $=$). Using (19) we then obtain (18) from (17) as a lemma.

The dual ${}^{\mathrm{co}}I$ of I is defined by its closure axiom

$$x \in {}^{\mathrm{co}}I \to \exists_{d,x',y}(d \in \mathrm{Sd} \land x' \in {}^{\mathrm{co}}I \land y = \frac{x'+d}{2} \land x = y).$$

Similar to what was done above it can be simplified to

$$x \in {}^{\mathrm{co}}I \to \exists_{d,x'}(d \in \mathrm{Sd} \land x' \in {}^{\mathrm{co}}I \land x = \frac{x'+d}{2}). \tag{20}$$

As examples we consider proofs that

(I) any real given in format (i) can be converted into format (ii), and
(II) the average of two real numbers in $[-1,1]$ is in $[-1,1]$ again, w.r.t. format (ii) for both input and output.

5.2.2. Conversion of real numbers into streams. An essential tool in the proof for the first example is Lemma 5.1.21 (ApproxSplit) on page 89, expressing the possibility to compare a real z given as pair $((c_n)_n, M)$ with a proper rational interval $[a,b]$.

THEOREM 5.2.1. $|x| \le 1 \to x \in {}^{\mathrm{co}}I$.

PROOF. We use coinduction with competitor predicate $\{\, x \mid |x| \le 1 \,\}$. It suffices to prove

$$|x| \le 1 \to \exists_{d,x'}(d \in \mathrm{Sd} \land (x' \in {}^{\mathrm{co}}I \lor |x'| \le 1) \land x = \frac{x'+d}{2}).$$

Compare x with $[-\frac{1}{2}, 0]$. If $x \le 0$, return SdL and continue with $2x+1$. If $-\frac{1}{2} \le x$, compare x with $[0, \frac{1}{2}]$. If $x \le \frac{1}{2}$, return SdM and continue with $2z$. If $0 \le x$, return SdR and continue with $2x-1$. □

More precisely, we have proved (with f of type $\mathbb{N} \to \mathbb{Q}$)

$$f \in T \to M \in T \to x \equiv (f, M) \to x \in \mathrm{Real} \to |x| \le 1 \to x \in {}^{\mathrm{co}}I$$

of cotype $(\mathbb{N} \to \mathbb{Q}) \to (\mathbb{P} \to \mathbb{N}) \to {}^{\mathrm{co}}\mathbb{S}(\mathbb{D})$. Here $x \in \mathrm{Real}$ is an n.c. formula expressing that x is a real number in format (i). The term extracted from

this proof has type $(\mathbb{N} \to \mathbb{Q}) \to (\mathbb{P} \to \mathbb{N}) \to \mathbb{S}(\mathbb{D})$ and is built by the corecursion operator with step function

$$x \mapsto \begin{cases} (\mathrm{SdL}, 2x+1) & \text{if } g(-\frac{1}{2}, 0, x) = \mathtt{tt} \\ (\mathrm{SdM}, 2x) & \text{if } g(-\frac{1}{2}, 0, x) = \mathtt{ff} \text{ and } g(0, \frac{1}{2}, x) = \mathtt{tt} \\ (\mathrm{SdR}, 2x-1) & \text{if } g(0, \frac{1}{2}, x) = \mathtt{ff}. \end{cases}$$

Here g is a function of type $\mathbb{Q} \to \mathbb{Q} \to \mathbb{R} \to \mathbb{B}$ representing the computational content of Lemma 5.1.21 (ApproxSplit).

5.2.3. Average of real numbers. In our second example we consider a proof that the average of two real numbers in $[-1, 1]$ is in $[-1, 1]$ again. Following Berger and Seisenberger (2010) we begin with an informal proof of Theorem 5.2.2 below. The computational content of this proof will be the desired algorithm.

THEOREM 5.2.2. *The average of two real numbers x, y in ${}^{\mathrm{co}}I$ is in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y \in {}^{\mathrm{co}}I}\Big(\frac{x+y}{2} \in {}^{\mathrm{co}}I\Big).$$

Consider two sets of averages, the second one with a "carry" $i \in \mathbb{Z}$

$$P := \{\, \frac{x+y}{2} \mid x, y \in {}^{\mathrm{co}}I \,\}, \qquad Q := \{\, \frac{x+y+i}{4} \mid x, y \in {}^{\mathrm{co}}I, i \in \mathrm{Sd}_2 \,\},$$

where Sd_2 is a (formally inductive) predicate expressing that the integer i is an extended signed digit, i.e., $|i| \le 2$. It suffices to show that Q satisfies

$$x \in Q \to \exists_{d,x'}(d \in \mathrm{Sd} \land x' \in Q \land x = \frac{x'+d}{2}),$$

for then by the greatest-fixed-point axiom for ${}^{\mathrm{co}}I$ we have $Q \subseteq {}^{\mathrm{co}}I$. Since we also have $P \subseteq Q$ we then obtain $P \subseteq {}^{\mathrm{co}}I$, which is our claim.

LEMMA 5.2.3. $x, y \in {}^{\mathrm{co}}I \to \exists_{i,x',y'}(i \in \mathrm{Sd}_2 \land x', y' \in {}^{\mathrm{co}}I \land \frac{x+y}{2} = \frac{x'+y'+i}{4})$.

PROOF. By (20). The extracted term has type $\mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D}) \to \mathbb{D}_2 \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D})$; it maps $(\mathrm{C}_d(u), \mathrm{C}_e(v))$ into $(d+e, u, v)$. □

The main part of the proof consists in showing

$$\begin{aligned} &i \in \mathrm{Sd}_2 \to x, y \in {}^{\mathrm{co}}I \to \\ &\exists_{j,d,x',y'}(j \in \mathrm{Sd}_2 \land d \in \mathrm{Sd} \land x', y' \in {}^{\mathrm{co}}I \land \frac{x+y+i}{4} = \frac{\frac{x'+y'+j}{4} + d}{2}). \end{aligned} \tag{21}$$

PROOF. By (20) we can write $x = \frac{x'+d}{2}$ and $y = \frac{y'+e}{2}$ with $d, e \in \mathrm{Sd}$ and $x', y' \in {}^{\mathrm{co}}I$. Then $\frac{x+y+i}{4} = \frac{x'+y'+d+e+2i}{8}$. Since $|d+e+2i| \le 6$ we can write

$d+e+2i = j+4k$ with $|j| \le 2$ and $|k| \le 1$. Therefore

$$\frac{x+y+i}{4} = \frac{x'+y'+j+4k}{8} = \frac{\frac{x'+y'+j}{4}+k}{2}.$$

Hence we have $q\colon \mathbb{D}_2 \to \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D}) \to \mathbb{D}_2 \times \mathbb{D} \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D})$ defined by $q(i, \mathrm{C}_d(u), \mathrm{C}_e(v)) := (J(d+e+2i), K(d+e+2i), u, v)$, where $J, K\colon \mathbb{Z} \to \mathbb{Z}$ with $\forall_i(|J(i)| \le 2)$, $\forall_i(|i| \le 6 \to |K(i)| \le 1))$ and $\forall_i(i = J(i) + 4K(i))$. □

By coinduction from (21) we obtain

LEMMA 5.2.4. $\exists_{i,x,y}(i \in \mathrm{Sd}_2 \land x, y \in {}^{\mathrm{co}}I \land z = \frac{x+y+i}{4}) \to z \in {}^{\mathrm{co}}I$.

Theorem 5.2.2 is an immediate consequence of Lemmata 5.2.3 and 5.2.4. Its proof gives a function $\mathrm{Av}\colon \mathbb{D}_2 \times \mathbb{S}(\mathbb{D}) \times \mathbb{S}(\mathbb{D}) \to \mathbb{S}(\mathbb{D})$ defined corecursively by $\mathrm{Av}(i, \mathrm{C}_d(v), \mathrm{C}_e(w)) = \mathrm{C}_{K(d+e+2i)}(\mathrm{Av}(J(d+e+2i), v, w))$. More precisely, Lemma 5.2.3 computes the first "carry" $i \in \mathrm{Sd}_2$ and the tails of the inputs. Then Av is called repeatedly, computing the average step by step. For the first n digits in the Sd-representation of $\frac{x+y}{2}$ we need the first $n+1$ digits in the Sd-representations of x and y.

5.2.4. Division for signed digit streams. We finally prove that $[-1, 1]$ is closed under division, w.r.t. the representation of reals as signed digit streams. Using ideas from Ciaffaglione and Gianantonio (2006), we will reduce this problem to the one for the average function.

For division $\frac{x}{y}$ we clearly need a restriction on the denominator y to stay in the interval $[-1, 1]$, and must assume that $|y|$ is strictly positive. For simplicity we assume $\frac{1}{4} \le y$; this can easily be extended to the case $2^{-p} \le y$ (using induction on p and Lemma 5.2.6 below). The main idea of the algorithm and also of the proof consists in three representations of $\frac{x}{y}$:

$$\frac{x}{y} = \frac{1+\frac{x_1}{y}}{2} = \frac{0+\frac{x_0}{y}}{2} = \frac{-1+\frac{x_{-1}}{y}}{2}$$

where

$$x_1 = 4\frac{x+\frac{-y}{2}}{2}, \quad x_0 = 2x, \quad x_{-1} = 4\frac{x+\frac{y}{2}}{2}.$$

Depending on what we know about x we choose one of these representations of $\frac{x}{y}$ to obtain its first digit. This will give us a corecursive definition of $\frac{x}{y}$.

LEMMA 5.2.5 (CoINegToCoIPlusOne, CoIPosToCoIMinusOne). *${}^{\mathrm{co}}I$ is closed under shifting a real $x \le 0$ ($x \ge 0$) by $+1$ (-1):*

$$\forall_{x \in {}^{\mathrm{co}}I}(x \le 0 \to x+1 \in {}^{\mathrm{co}}I),$$
$$\forall_{x \in {}^{\mathrm{co}}I}(0 \le x \to x-1 \in {}^{\mathrm{co}}I).$$

PROOF. We only consider the first claim; the second is proved similarly. The proof uses coinduction on ${}^{\mathrm{co}}I$ with $P := \{\, x \mid \exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x = y+1)\,\}$ the competitor predicate. It suffices to prove the step formula

$$\forall_x\Big(\exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x = y+1) \to \exists_{d\in\mathrm{Sd},x'\in{}^{\mathrm{co}}I\cup P}\Big(|x| \le 1 \land x = \frac{d+x'}{2}\Big)\Big).$$

Let x, y be given with $y \in {}^{\mathrm{co}}I$, $y \le 0$ and $x = y+1$. From $y \in {}^{\mathrm{co}}I$ we know $|y| \le 1$ and with $y \le 0$ also $|x| \le 1$. We need $d \in \mathrm{Sd}$ and x' such that

$$(x' \in {}^{\mathrm{co}}I \vee \exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x' = y+1)) \land x = \frac{d+x'}{2}.$$

Again from $y \in {}^{\mathrm{co}}I$ we obtain $e \in \mathrm{Sd}$ and $z \in {}^{\mathrm{co}}I$ with $y = \frac{e+z}{2}$. We now distinguish cases on $e \in \mathrm{Sd}$.

Case $e = 1$. Then $0 \ge y = \frac{1+z}{2} \ge \frac{1}{2} - \frac{1}{2} = 0$ and hence $x = y+1 = 1$. Picking $d = 1$ and $x' = 1$ gives the claim. Here we need a lemma `CoIOne` stating that 1 is in ${}^{\mathrm{co}}I$. The proof of this lemma (by coinduction) is omitted.

Case $e = 0$. Pick $d = 1$ and $x' = z+1$. Then $z = 2y \le 0$ and hence the r.h.s. of the disjunction above holds with z for y. Also $x = 2y = \frac{1+2y+1}{2}$.

Case $e = -1$. Pick $d = 1$ and $x' = z$. Then the l.h.s. of the disjunction above holds, and $\frac{d+x'}{2} = \frac{1+z}{2} = \frac{-1+z}{2} + 1 = y+1 = x$. □

We have formalized this proof M in the Minlog[1] proof assistant. Minlog has a tool to extract from the proof M the term $\mathrm{et}(M)$ representing the computational content of the proof M. This term is displayed as

```
[u](CoRec ai=>ai)u
 ([u0][case (DesYprod u0)
     (s pair u1 -> [case s
       (SdR -> SdR pair InL cCoIOne)
       (SdM -> SdR pair InR u1)
       (SdL -> SdR pair InL u1)])])
```

Here `[u]` means lambda abstraction λ_u, and `(CoRec ai=>ai)` is the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}}$ defined above, where τ is $\mathbb{I}$ again. The type of ${}^{\mathrm{co}}\mathcal{R}^{\mathbb{I}}_{\mathbb{I}}$ is $\mathbb{I} \to (\mathbb{I} \to \mathbb{D} \times (\mathbb{I} + \mathbb{I})) \to \mathbb{I}$. The first argument of the corecursion operator is the abstracted variable `u` of type $\mathbb{I}$, and the second `([u0][case ...])` is the "step" function, which first destructs its argument `u0` into a pair of a signed digit `s` and another stream `u1`, and then distinguishes cases on `s`. In the `SdR` case the returned pair of type $\mathbb{D} \times (\mathbb{I} + \mathbb{I})$ has `SdR` again as its left component, and as right component the InL (left embedding into a sum

[1]The development described here resides in the dev branch of Minlog, in the directory `minlog/examples/analysis`, files `sddiv.scm` and `graydiv.scm`

type) of a certain stream denoted `cCoIOne`. This is the computational content (hence the "`c`") of `CoIOne`, essentially an infinite sequence of the digit SdR (written $\vec{1}$).

The algorithm represented by $\mathrm{et}(M)$ can be understood as follows. If $s = \mathrm{SdR}$, then y must be non-negative. Hence $y = 0$, and a stream-representation of $y + 1$ is $\vec{1}$. Here we do not need a corecursive call, and hence the result is $\langle \mathrm{SdR}, \mathrm{InL}(\vec{1}\,)\rangle$. The same happens in case $s = \mathrm{SdL}$. Here $y + 1$ can be determined easily by changing the first digit from SdL to SdR and leaving the tail as it is. Hence the result is $\langle \mathrm{SdR}, \mathrm{InL}(u')\rangle$. Only in case $s = \mathrm{SdM}$ we have a corecursive call. Here we change the first digit from SdM to SdR, which however amounts to adding $\frac{1}{2}$ only. Therefore the procedure continues with the tail. Hence the result is $\langle \mathrm{SdR}, \mathrm{InR}(u')\rangle$. Using the computation rule for ${}^{\mathrm{co}}\mathcal{R}^{\mathbb{I}}_{\mathbb{I}}$ we can now describe the computational content as a function $f\colon \mathbb{I} \to \mathbb{I}$ defined by

$$\begin{aligned} f(\mathrm{SdR} :: u) &:= [\mathrm{SdR}, \mathrm{SdR}, \dots], \\ f(\mathrm{SdM} :: u) &:= \mathrm{SdR} :: f(u), \\ f(\mathrm{SdL} :: u) &:= \mathrm{SdR} :: u. \end{aligned}$$

A similar argument for the second part of the lemma gives $g\colon \mathbb{I} \to \mathbb{I}$ with

$$\begin{aligned} g(\mathrm{SdR} :: u) &:= \mathrm{SdL} :: u, \\ g(\mathrm{SdM} :: u) &:= \mathrm{SdL} :: g(u), \\ g(\mathrm{SdL} :: u) &:= [\mathrm{SdL}, \mathrm{SdL}, \dots]. \end{aligned}$$

LEMMA 5.2.6. *For x in ${}^{\mathrm{co}}I$ with $|x| \le \frac{1}{2}$ we have $2x$ in ${}^{\mathrm{co}}I$:*

$$\forall_{x \in {}^{\mathrm{co}}I}\Big(|x| \le \frac{1}{2} \to 2x \in {}^{\mathrm{co}}I\Big).$$

PROOF. Let $x \in {}^{\mathrm{co}}I$ be given. From the closure axiom for ${}^{\mathrm{co}}I$ we obtain $d \in \mathrm{Sd}$ and $x' \in {}^{\mathrm{co}}I$ such that $x = \frac{d+x'}{2}$. We distinguish cases on $d \in \mathrm{Sd}$.

Case $d = 1$. Then $x = \frac{1+x'}{2}$ and hence $2x = 1 + x'$. Since $|x| \le \frac{1}{2}$ we have $x' \le 0$. Now the first part of Lemma 5.2.5 gives the claim.

Case $d = 0$. Then $x = \frac{0+x'}{2}$ and hence $2x = x' \in {}^{\mathrm{co}}I$.

Case $d = -1$. Then $x = \frac{-1+x'}{2}$ and hence $2x = -1 + x'$. Since $|x| \le \frac{1}{2}$ we have $0 \le x'$. Now the second part of Lemma 5.2.5 gives the claim. □

Using arguments similar to those in the remark after Lemma 5.2.5 we can see that the corresponding algorithm can be written as a function

Double: $\mathbb{I} \to \mathbb{I}$ with

$$\begin{aligned}\mathrm{Double}(\mathrm{SdR} :: u) &:= f(u),\\ \mathrm{Double}(\mathrm{SdM} :: u) &:= u,\\ \mathrm{Double}(\mathrm{SdL} :: u) &:= g(u)\end{aligned}$$

where f and g are the functions from this remark.

LEMMA 5.2.7. *For x, y in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$ ($x \le 0$) we have $2x - y$ ($2x + y$) in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to 0 \le x \to 4\frac{x + \frac{-y}{2}}{2} \in {}^{\mathrm{co}}I\Big),$$
$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to x \le 0 \to 4\frac{x + \frac{y}{2}}{2} \in {}^{\mathrm{co}}I\Big).$$

PROOF. We essentially use the function $\mathrm{Av}\colon \mathbb{I} \to \mathbb{I} \to \mathbb{I}$ extracted from the proof of Theorem 5.2.2. In the formulas above instead of $2x \pm y$ we have written $4\frac{x+(\pm y/2)}{2}$ to make Theorem 5.2.2 applicable. We also use two lemmas stating that ${}^{\mathrm{co}}I$ is closed under $x \mapsto \frac{x}{2}$ and $x \mapsto -x$. To prove the first claim, let x, y in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$. Then clearly $\frac{-y}{2} \in {}^{\mathrm{co}}I$, and $\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}I$ by Theorem 5.2.2. We have

$$\Big|\frac{x + \frac{-y}{2}}{2}\Big| = \Big|\frac{2x - y}{4}\Big| \le \Big|\frac{2y - y}{4}\Big| = \Big|\frac{y}{4}\Big| \le \frac{1}{4}. \tag{22}$$

Hence we can apply Lemma 5.2.6 twice and obtain $4\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}I$. The proof of the second claim is similar. □

The computational content of the proofs is $\mathrm{AuxL}, \mathrm{AuxR}\colon \mathbb{I} \to \mathbb{I} \to \mathbb{I}$:

$$\begin{aligned}\mathrm{AuxL}(u, v) &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(n(v))))),\\ \mathrm{AuxR}(u, v) &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(v)))).\end{aligned}$$

Here $h, n\colon \mathbb{I} \to \mathbb{I}$ represent the computational content of the two lemmas used in the proof; both are proved by coinduction. The function h prepends SdM to the stream, and n negates all digits:

$$h(u) := \mathrm{SdM} :: u, \qquad \begin{aligned} n(\mathrm{SdR} :: u) &:= \mathrm{SdL} :: n(u),\\ n(\mathrm{SdM} :: u) &:= \mathrm{SdM} :: n(u),\\ n(\mathrm{SdL} :: u) &:= \mathrm{SdR} :: n(u).\end{aligned}$$

THEOREM 5.2.8. *For x, y in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$ and $|x| \le y$ we have $\frac{x}{y}$ in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to \frac{x}{y} \in {}^{\mathrm{co}}I\Big).$$

PROOF. The proof uses coinduction on ${}^{\mathrm{co}}I$ with $P := \{\, z \mid \exists_{x,y\in{}^{\mathrm{co}}I}(|x| \le y \land \frac{1}{4} \le y \land z = \frac{x}{y})\,\}$. It suffices to prove the step formula

$$\forall_z\Big(\exists_{x,y\in{}^{\mathrm{co}}I}\Big(|x| \le y \land \frac{1}{4} \le y \land z{=}\frac{x}{y}\Big) \to \exists_{d\in\mathrm{Sd},z'\in{}^{\mathrm{co}}I\cup P}\Big(|z| \le 1 \land z{=}\frac{d+z'}{2}\Big)\Big).$$

Let x, y, z be given with $x, y \in {}^{\mathrm{co}}I$, $|x| \le y$, $\frac{1}{4} \le y$ and $z = \frac{x}{y}$. From $|x| \le y$ we have $z \le 1$. By a trifold application of the closure axiom to x we obtain $d_1, d_2, d_3 \in \mathrm{Sd}$ and $\tilde{x} \in {}^{\mathrm{co}}I$ such that $x = \frac{4d_1+2d_2+d_3+\tilde{x}}{8}$ or $x = d_1d_2d_3\tilde{x}$ for short. We now distinguish three cases.

If $x = 1d_2d_3\tilde{x}$, $x = 01d_3\tilde{x}$ or $x = 001\tilde{x}$, then $0 \le x$. Pick $d = 1$ and $z' = \frac{x'}{y}$ with $x' = 4\frac{x+\frac{-y}{2}}{2}$. Then $x' \in {}^{\mathrm{co}}I$ by Lemma 5.2.7. From (22) we also obtain $|x'| \le y$ and hence $z' \in P$. One can easily check that $z = \frac{1+z'}{2}$.

If $x = \bar{1}d_2d_3\tilde{x}$, $x = 0\bar{1}d_3\tilde{x}$ or $x = 00\bar{1}\tilde{x}$, then $x \le 0$. Pick $d = -1$ and $z' = \frac{x'}{y}$ with $x' = 4\frac{x+\frac{y}{2}}{2}$. We can then proceed as in the first case.

The final case is $x = 000\tilde{x}$. Then $|x| \le \frac{1}{8}$; pick $d = 0$ and $z' = \frac{x'}{y}$ with $x' = 2x$. We obtain $|x'| \le \frac{1}{4} \le y$ and with Lemma 5.2.6 also $x' \in {}^{\mathrm{co}}I$. Therefore $z' \in P$, and $z = \frac{z'}{2}$ is easily checked. □

The term Minlog extracts is displayed in Figure 2.

```
[u,u0](CoRec ai=>ai)u
 ([u1][case (cCoIClosure u1)
   (s pair u2 -> [case s
     (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
     (SdM -> [case (cCoIClosure u2)
       (s0 pair u3 -> [case s0
        (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
        (SdM -> [case (cCoIClosure u3)
         (s1 pair u4 -> [case s1
          (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
          (SdM -> SdM pair InR(cCoIToCoIDouble u1))
          (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
        (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
     (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
```

FIGURE 1. Extracted term for Theorem 5.2.8.

The three occurrences of `cCoIClosure` correspond to the trifold application of the closure axioms to x. The seven cases $x = 1d_2d_3\tilde{x}$, $x = 01d_3\tilde{x}$, $x =$

$001\tilde{x}$, $x = 000\tilde{x}$, $x = \bar{1}d_2d_3\tilde{x}$, $x = 0\bar{1}d_3\tilde{x}$ and $x = 00\bar{1}\tilde{x}$ are clearly visible. In the first three cases `SdR` corresponds to picking $d = 1$ and usage of the AuxR function from Lemma 5.2.7, and similarly in the last three cases `SdL` corresponds to picking $d = -1$ and usage of AuxL. In the middle case `SdM` corresponds to $d = 0$; there we use the computational content of Lemma 5.2.6. We can describe the extracted term by a function $\mathrm{Div}\colon \mathbb{I} \to \mathbb{I} \to \mathbb{I}$ corecursively defined by

$$\mathrm{Div}(u,v) := \begin{cases} \mathrm{SdR} :: \mathrm{Div}(\mathrm{AuxR}(u,v),v) & \text{if } u = 1\tilde{u} \vee u = 01\tilde{u} \vee u = 001\tilde{u}, \\ \mathrm{SdM} :: \mathrm{Div}(\mathrm{Double}(u),v) & \text{if } u = 000\tilde{u}, \\ \mathrm{SdL} :: \mathrm{Div}(\mathrm{AuxL}(u,v),v) & \text{if } u = \bar{1}\tilde{u} \vee u = 0\bar{1}\tilde{u} \vee u = 00\bar{1}\tilde{u}. \end{cases}$$

We use this description of the extracted term to see how far we have to look into u and v to determine the first n entries of $\mathrm{Div}(u,v)$. To this end we write the above equation as

$$\mathrm{Div}(u,v) = d(u) :: \mathrm{Div}(G(u,v),v),$$

where $d(u)$ depends on the first three digits of u, and $G(u,v)$ is one of $\mathrm{AuxR}(u,v)$, $\mathrm{Double}(u)$ or $\mathrm{AuxL}(u,v)$, according to the present case. Recall

$$\begin{aligned} \mathrm{AuxL}(u,v) &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u,h(n(v))))), \\ \mathrm{AuxR}(u,v) &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u,h(v)))). \end{aligned}$$

By the equations for n, h, Av and Double we see that the first n entries of

$n(u)$	need the first n entries of u,
$h(u)$	need the first $n-1$ entries of u,
$\mathrm{Av}(u,v)$	need the first $n+1$ entries of u and v,
$\mathrm{Double}(u)$	need the first $n+1$ entries of u.

Hence $\mathrm{AuxR}(u,v)$, $\mathrm{AuxL}(u,v)$ and $G(u,v)$ all need at most the first $n+3$ entries of u and $n+2$ entries of v. Iterating the above equation for G gives for $\mathrm{Div}(u,v)$ the representation

$$d(u) :: d(G(u,v)) :: d(G(G(u,v),v)) :: d(G(G(G(u,v),v),v),v) \ldots$$

Therefore the first n entries of $\mathrm{Div}(u,v)$ depend on at most the first $3n$ entries of u and the first $3n-1$ entries of v.

Bibliography

Ulrich Berger and Monika Seisenberger. Proofs, programs, processes. In F. Ferreira et al., editors, *Proceedings CiE 2010*, volume 6158 of *LNCS*, pages 39–48. Springer Verlag, Berlin, Heidelberg, New York, 2010.

Alberto Ciaffaglione and Pietro Di Gianantonio. A certified, corecursive implementation of exact real numbers. *Theoretical Computer Science*, 351:39–51, 2006.

Haskell B. Curry. Grundlagen der kombinatorischen Logik. *Amer. J. Math.*, 52:509–536, 789–834, 1930.

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.

Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

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

David Hilbert. Über das Unendliche. *Mathematische Annalen*, 95:161–190, 1925.

William A. Howard. The formulae–as–types notion of construction. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

Ingebrigt Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 4:119–136, 1937.

Andrey N. Kolmogorov. Zur Deutung der intuitionistischen Logik. *Math. Zeitschr.*, 35:58–65, 1932.

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

Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

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

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.

Index

$\tilde{\exists}, \tilde{\forall}$, 9
$\bot$, 4
$\tilde{\wedge}$, 9

algebra, 28
  associated with $(I^c/^{\mathrm{co}}I^c)(\vec{\rho}, \vec{P})$, 48
  closed, 28
  non-recursive, 61
algebra form, 27
  explicit, 27
  non-recursive, 27
application, 22, 25
approximable map, 23
ApproxSplit, 89
arity, 45
assumption, 5
  cancelled, 5
  closed, 5
  open, 5

binary tree, 28
bisimilarity, 33, 46, 59, 61
  axiom, 59
  relative, 61
boolean, 28

case-construct, 40
$\mathcal{C}$-operator, 39
Cauchy modulus, 83
Cauchy sequence, 83
clause, 46
  nullary, 46
closure
  transitive, 48
closure axiom, 33, 34, 54
compatibility
  of extracted terms, 80
  of terms, 63
comprehension term, 47
computation rule, 36
conclusion, 5
conjunction, 7, 8
  decoration of, 50
  inductive definition, 50
consistent, 21
constructor
  as continuous function, 35
constructor pattern, 36
constructor type, 27
conversion
  D-, 37
corecursion, 40
  operator, 41
cototal, 30
cototality
  relative, 46
cotype, 59
  of a formula, 60
Curry, 16
Curry-Howard correspondence, 3, 16

decoration
  of conjunction, 50
  of disjunction, 50
  of existence, 51
derivable, 5
  classically, 10
  intuitionistically, 10
derivation, 4, 28
  r-free, 65
  n.c. part, 51
  term, 16
destructor, 40

disjunction, 7, 8
  decoration of, 50
  inductive definition, 49
drinker formula, 13
duplication, 39

E-rule, 5
EfEqD, 57
effectivity principle, 20
elimination axiom, 51
entailment, 21
equality, 48
  decidable, 37
  Leibniz, 49, 57
  pointwise, 62
  w.r.t. a cotype, 61
ex-falso-quodlibet, 4, 9, 57
existence
  decoration of, 51
  inductive definition, 51
existential quantifier, 7, 8
extensionality, 26
  of extracted terms, 80
  of terms, 64
  w.r.t. a cotype, 62

falsity **F**, 57
falsum $\bot$, 4
finite support principle, 19, 25
formal neighborhood, 21
formula, 47
  r-free, 65
  computationally relevant (c.r.), 47
  non-computational (n.c.), 47
function
  continuous, 25
  monotone, 24
  strict, 35
functional
  computable, 30

Gandy, 59
Gentzen, 3
  translation, 15
greatest-fixed-point axiom, 54

Harrop
  formula, 47
  predicate, 47
height
  of a base type, 48
  of a cotype, 62
Howard, 16

I-rule, 5
ideal, 21
  cototal, 30
  total, 30
identity, 28
information system, 21
  flat, 21
  of a type τ, 29
intersection
  inductive definition, 50
intuitionistic logic, 4
invariance axiom, 66
inversion, 52

Johansson, 4

Kolmogorov, 3

Larsen, 21
least-fixed-point axiom, 51
Leibniz equality, 49, 56, 57
LeIsNotGt, 89
level
  of a type, 28
list, 28

marker, 5
Markov, 58
Martin-Löf, 16
minimal logic, 5
modus ponens, 6
monotonicity principle, 20

nullary
  argument type, 27
number, 28
  binary, 28
  even, 48
  positive, 28
  unary, 28

parameter
  argument type, 27
predecessor, 40
predicate, 47
  r-free, 65
  coinductive, 47

computationally relevant (c.r.), 47
form, 46
inductive, 47
non-computational (n.c.), 47
one-clause-nc, 56, 75
premise, 5
major, 6
minor, 6
parameter, 46
recursive, 46
product, 28
proof, 4

real
nonnegative, 85
positive, 85
RealApprox, 88
RealBound, 84
RealEqChar, 84
RealNNegChar, 85
RealNNegCompat, 85
RealPosChar, 86
RealPosCompat, 86
RealPosPlus, 89
reals
equal, 84
equivalent, 84
recursion
operator, 38, 39
recursive
argument type, 27
rule, 5

Scott, 21
set of tokens
deductively closed, 21
similarity, 46, 59, 61
relative, 61
soundness theorem
for realizability, 75
stability, 10
stream, 28
substitution
sharp, 47
sum, 28

Takeuti, 59
TCF, 51
term
extracted, 70
of T^+, 36
token, 21
extended, 29
total, 30
totality
relative, 46
transitive closure, 53
tree
binary, 28
type, 28
base, 28
higher, 28
level of, 28
of a formula, 47
of a predicate, 47

union
inductive definition, 49
unit, 28
uysum, 28

variable
assumption, 5
object, 5
variable condition, 6, 8

Winskel, 21

ysumu, 28