

Logic II

Helmut Schwichtenberg

MATHEMATISCHES INSTITUT DER LMU, SOMMERSEMESTER 2023

Contents

Preface	v
Introduction	vii
Chapter 1. Logic	1
1.1. Natural deduction	2
1.2. Derivation terms, normalization	9
Chapter 2. Computable functionals	13
2.1. Information systems	13
2.2. Partial continuous functionals	20
2.3. Terms	26
2.4. Denotational semantics	34
Chapter 3. A theory TCF of computable functionals	35
3.1. Formulas and their computational content	35
3.2. Axioms of TCF	41
3.3. Equality and extensionality	47
Chapter 4. Computational content of proofs	55
4.1. Realizers	55
4.2. Extracted terms, soundness	60
4.3. Extensionality of extracted terms	62
Chapter 5. Applications	63
5.1. Exact real arithmetic	63
5.2. Algorithms on stream-represented real numbers	70
5.3. Lookahead	74
Appendix A. Atomic coherent information systems	77
Appendix B. Unification	85
Appendix C. Denotational semantics	89
C.1. Ideals as values of terms	89
C.2. Preservation of values	95

Appendix D. Realizers for axioms	99
Appendix E. Extensionality of extracted terms	105
Bibliography	111
Index	113

Preface

This is the script for the course “Mathematical Logic” at LMU, held in Sommersemester 2023. It is mainly based on the textbook “Proofs and Computations” in the references.

München, 12. April 2023

Helmut Schwichtenberg

Introduction

Mathematics differs from all other sciences by the fact that it provides proofs for its claims. In this course we study what proofs are, and what we can do with them apart from assuring us of the truth of what they state.

Present technology makes it feasible to generate even large formal proofs with “proof assistants”. These proof objects can be checked for logical correctness, independently of the used proof assistant: one only needs to check that the logical rules have been applied correctly.

Our underlying “minimal” logic is “constructive” in the sense that a “computationally relevant” (c.r.) predicate (for instance an existential statement $\exists_x A$ or a disjunction $A \vee B$) can only be proved by providing an example. Then from a proof of a c.r. statement we can extract a term which can be seen as a program representing the computational content of the proof. We will develop the theoretical concepts used to formulate and prove this “Soundness theorem”.

Usage of a proof assistant makes it possible to automate this extraction process, and also to automatically generate a formal proof that the extracted term “realizes” the original statement. Several applications will be given.

The following list describes some relevant issues. Possibly cryptic notions used above or in these descriptions will be defined in the course.

- (1) **Concrete vs. abstract.** To study extraction of programs from proofs it seems best to start with concrete rather than abstract mathematics.
- (2) **Types.** Mathematical objects have “types”. Examples are “base types” $\mathbb{N}$, $\mathbb{Q}$ of natural or rational numbers, and the “function type” $\mathbb{N} \rightarrow \mathbb{N}$ of unary functions f on $\mathbb{N}$. The iteration operator I with $I(f, n, m) := f^{(n)}(m)$ has type $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$.
- (3) **Approximations.** Infinite objects are viewed as given by their (finite) approximations. Examples are real numbers, or functions $f: \mathbb{N} \rightarrow \mathbb{N}$.
- (4) **Model.** Objects of type τ are the “partial continuous functionals” of Scott (1982) and Ershov (1977). Continuity of a function f here means that for every approximation V of the value $f(x)$ there is

an approximation U of the argument x such that $f[U]$ has more information than V .

- (5) **Constants** of type τ are given by “computation rules”. Example: $\text{Double}: \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$\text{Double}(0) = 0, \quad \text{Double}(S(n)) = S(S(\text{Double}(n))).$$

- (6) **Predicates** of arity $\vec{\tau}$ (a list of types) are either variables or constants. A predicate variable is a place holder for a formula with some object variables bound (for example $\{n, m \mid n \leq m\}$). A predicate constant is inductively or coinductively defined. This means that it is the least or greatest fixed point of some “clauses”. An example of an inductive predicate is Even , the least fixed point of the clauses

$$\text{Even}(0), \quad \forall n(\text{Even}(n) \rightarrow \text{Even}(S(S(n)))).$$

An example of a coinductive predicate is ${}^{\text{co}}\text{Even}$, the greatest fixed point of the single clause

$$\forall n({}^{\text{co}}\text{Even}(n) \rightarrow \exists m(n = S(S(m)) \wedge {}^{\text{co}}\text{Even}(m))).$$

The logical connectives $\exists$, $\forall$ and $\wedge$ are viewed as inductively defined predicates with parameters.

- (7) **Proof trees.** To generate proofs we think of them as trees with “holes” to be filled. Each hole carries a formula (called “goal”) and a “context” consisting of object and assumption variables. The task then is to “refine” the proof by filling a hole with a logical rule, which may bind one of the object or assumption variables. This process is well supported by an interactive proof assistant. Guidance by a previous proof idea is of course essential here.

We shall use the proof assistant Minlog <http://www.minlog-system.de> (under development at LMU), which is suitable for the tasks described above. It is implemented in the programming language Scheme (cf. Abelson et al. (1996))

CHAPTER 1

Logic

The main subject of Mathematical Logic is mathematical proof. In this introductory chapter we deal with the basics of formalizing such proofs and, via normalization, analysing their structure. The system we pick for the representation of proofs is Gentzen's natural deduction from (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 proceed anyway. Second, formal proofs in natural deduction are closely related (via the so-called 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 unmanagable tree-like structures), but also opens up a route to applying the computational techniques which underpin lambda calculus.

An underlying theme of this chapter is to bring out the constructive content of logic, particularly in regard to the relationship between minimal and classical logic. For us the latter is most appropriately viewed as a subsystem of the former. This approach will reveal some interesting aspects of proofs, e.g., that it is possible and useful to distinguish between existential proofs that actually construct witnessing objects, and others that don't.

As an example for a non-constructive existence proof, consider the following proposition.

There are irrational numbers a, b such that a^b is rational.

This can be proved as follows, by cases.

Case $\sqrt{2}^{\sqrt{2}}$ is rational. Choose $a = \sqrt{2}$ and $b = \sqrt{2}$. Then a, b are irrational and by assumption a^b is rational.

Case $\sqrt{2}^{\sqrt{2}}$ is irrational. Choose $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Then by assumption a, b are irrational and

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \left(\sqrt{2}\right)^2 = 2$$

is rational.

As long as we have not decided whether $\sqrt{2}^{\sqrt{2}}$ is rational, we do not know which numbers a, b we must take. Hence we have an example of an existence proof which does not provide an instance.

1.1. Natural deduction

The rules of natural deduction come in pairs: we have an introduction and an elimination rule for the logical connectives $\rightarrow$ and $\forall$. The resulting system is called *minimal logic*; it was introduced by Kolmogorov (1932), Gentzen (1935) and Johansson (1937). Notice that no negation is yet present. If we go on and require *ex-falso-quodlibet* for the nullary propositional symbol $\perp$ (“falsum”) we can embed *intuitionistic logic* with negation as $A \rightarrow \perp$. To embed classical logic, we need to go further and add as an axiom schema the principle of *indirect proof*, also called *stability* ($\forall \vec{x}(\neg\neg R\vec{x} \rightarrow R\vec{x})$ for relation symbols R).

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

$$u: A \quad \text{assumption.}$$

The other rules of natural deduction split into introduction rules (I-rules for short) and elimination rules (E-rules) for the logical connectives $\rightarrow$ and $\forall$. For implication $\rightarrow$ there is an introduction rule $\rightarrow^+$ and an elimination rule $\rightarrow^-$ also called *modus ponens*. The left premise $A \rightarrow B$ in $\rightarrow^-$ is called the *major* (or *main*) premise, and the right premise A the *minor* (or *side*) premise. Note that with an application of the $\rightarrow^+$ -rule *all* assumptions above it marked with $u: A$ 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 uncanceled assumptions $v: A$ of the same formula A , which may get cancelled at a later stage.

$$\frac{\begin{array}{c} [u: A] \\ | M \\ B \end{array}}{A \rightarrow B} \rightarrow^+ u \quad \frac{\begin{array}{c} | M \\ A \rightarrow B \end{array} \quad \begin{array}{c} | N \\ A \end{array}}{B} \rightarrow^-$$

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 must not contain any open assumption having x as a free

variable.

$$\frac{\begin{array}{c} | M \\ A \\ \hline \forall_x A \end{array}}{\forall^+ x} \quad \frac{\begin{array}{c} | M \\ \forall_x A(x) \quad t \\ \hline A(t) \end{array}}{\forall^-}$$

As an example we derive $(A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$:

$$\frac{\frac{\frac{u: A \rightarrow B \rightarrow C}{B \rightarrow C} \quad \frac{w: A}{B}}{\frac{C}{A \rightarrow C} \rightarrow^+ w} \quad \frac{v: A \rightarrow B \quad w: A}{B}}{\frac{(A \rightarrow B) \rightarrow A \rightarrow C \rightarrow^+ v}{(A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C} \rightarrow^+ u}$$

Since in many cases the rule used is determined by the conclusion, we may suppress in such cases the name of the rule. As an example involving the universal quantifier we derive $\forall_x(A \rightarrow B) \rightarrow A \rightarrow \forall_x B$, under the assumption that x is not free in A (written $x \notin \text{FV}(A)$):

$$\frac{\frac{\frac{u: \forall_x(A \rightarrow B) \quad x}{A \rightarrow B} \quad v: A}{\frac{B}{\forall_x B} \forall^+ x} \quad \frac{A \rightarrow \forall_x B \rightarrow^+ v}{\forall_x(A \rightarrow B) \rightarrow A \rightarrow \forall_x B} \rightarrow^+ u$$

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

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 $A_1, \dots, A_n$, if there is a derivation of B with free assumptions among $A_1, \dots, A_n$. 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 $A_1, \dots, A_n \in \Gamma$.

1.1.2. Properties of negation. Recall that negation was defined by $\neg A := (A \rightarrow \perp)$. The following can easily be derived.

$$\begin{aligned} A &\rightarrow \neg\neg A, \\ \neg\neg\neg A &\rightarrow \neg A. \end{aligned}$$

However, $\neg\neg A \rightarrow A$ is in general *not* derivable.

LEMMA. *The following are derivable.*

$$\begin{aligned}
& (A \rightarrow B) \rightarrow \neg B \rightarrow \neg A, \\
& \neg(A \rightarrow B) \rightarrow \neg B, \\
& \neg\neg(A \rightarrow B) \rightarrow \neg\neg A \rightarrow \neg\neg B, \\
& (\perp \rightarrow B) \rightarrow (\neg\neg A \rightarrow \neg\neg B) \rightarrow \neg\neg(A \rightarrow B), \\
& \neg\neg\forall_x A \rightarrow \forall_x \neg\neg A.
\end{aligned}$$

Derivations are left as an exercise.

We have not yet considered the logical connectives “or” (disjunction) and “exists”. They are special from a constructive point of view, since a proposition with such a connective should have “computational content”. For this reason we do not consider them as logical concepts, but rather as (special cases of) inductive definitions which will be dealt with later in this course.

However, there is a substitute, the so-called weak (or “classical”) form of disjunction and existence. We write them with a tilde to mark the difference from the strong form of these connectives. They are defined by

$$A \tilde{\vee} B := \neg A \rightarrow \neg B \rightarrow \perp, \quad \tilde{\exists}_x A := \neg\forall_x \neg A.$$

Since $\tilde{\exists}_x \tilde{\exists}_y A$ unfolds into an awkward formula we extend the $\tilde{\exists}$ -terminology to lists of variables:

$$\tilde{\exists}_{x_1, \dots, x_n} A := \forall_{x_1, \dots, x_n} (A \rightarrow \perp) \rightarrow \perp.$$

Moreover let

$$\tilde{\exists}_{x_1, \dots, x_n} (A_1 \tilde{\wedge} \dots \tilde{\wedge} A_m) := \forall_{x_1, \dots, x_n} (A_1 \rightarrow \dots \rightarrow A_m \rightarrow \perp) \rightarrow \perp.$$

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

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

$$\forall_{\vec{x}} (\perp \rightarrow R\vec{x})$$

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

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

$$\forall_{\vec{x}} (\neg\neg R\vec{x} \rightarrow R\vec{x})$$

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

It is easy to see that intuitionistically (i.e., from Efq) we can derive $\perp \rightarrow A$ for an *arbitrary* formula A , using the introduction rules for the connectives.

THEOREM (Stability, or principle of indirect proof).

- (a) $\vdash (\neg\neg B \rightarrow B) \rightarrow \neg\neg(A \rightarrow B) \rightarrow A \rightarrow B$.
- (b) $\vdash (\neg\neg A \rightarrow A) \rightarrow \neg\neg\forall_x A \rightarrow A$.
- (c) $\vdash_c \neg\neg A \rightarrow A$.

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

$$\frac{\frac{u: \neg\neg B \rightarrow B}{B} \quad \frac{\frac{v: \neg\neg(A \rightarrow B)}{\frac{\frac{u_1: \neg B}{\frac{\frac{u_2: A \rightarrow B}{B} \quad w: A}{B}}{\perp}} \neg(A \rightarrow B)}{\rightarrow^+ u_2}}{\rightarrow^+ u_1}}{\rightarrow^+ u}$$

(b)

$$\frac{\frac{u: \neg\neg A \rightarrow A}{A} \quad \frac{\frac{v: \neg\neg\forall_x A}{\frac{\frac{u_1: \neg A}{\frac{\frac{u_2: \forall_x A}{A} \quad x}{A}}{\perp}} \neg\forall_x A}{\rightarrow^+ u_2}}{\rightarrow^+ u_1}}{\rightarrow^+ u}$$

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

$$\frac{v: (\perp \rightarrow \perp) \rightarrow \perp \quad \frac{u: \perp}{\perp \rightarrow \perp} \rightarrow^+ u}{\perp}$$

In the cases $A \rightarrow B$ and $\forall_x A$ use (a) and (b), 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. *The following are derivable.*

- (1) $(\tilde{\exists}_x A \rightarrow B) \rightarrow \forall_x (A \rightarrow B)$ if $x \notin \text{FV}(B)$,
- (2) $(\neg\neg B \rightarrow B) \rightarrow \forall_x (A \rightarrow B) \rightarrow \tilde{\exists}_x A \rightarrow B$ if $x \notin \text{FV}(B)$,
- (3) $(\perp \rightarrow B[x:=c]) \rightarrow (A \rightarrow \tilde{\exists}_x B) \rightarrow \tilde{\exists}_x (A \rightarrow B)$ if $x \notin \text{FV}(A)$,
- (4) $\tilde{\exists}_x (A \rightarrow B) \rightarrow A \rightarrow \tilde{\exists}_x B$ if $x \notin \text{FV}(A)$.

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) $(\neg\neg B \rightarrow B) \rightarrow \tilde{\exists}_x (A \rightarrow B) \rightarrow \forall_x A \rightarrow B$ if $x \notin \text{FV}(B)$,
- (6) $\forall_x (\neg\neg A \rightarrow A) \rightarrow (\forall_x A \rightarrow B) \rightarrow \tilde{\exists}_x (A \rightarrow B)$ if $x \notin \text{FV}(B)$.

PROOF. (1)

$$\frac{\frac{\frac{u_1: \forall_x \neg A \quad x}{\neg A} \quad A}{\tilde{\exists}_x A \rightarrow B} \quad \frac{\frac{\perp}{\neg \forall_x \neg A} \rightarrow^+ u_1}{B}}{B}$$

(2)

$$\frac{\frac{\frac{\frac{\frac{\forall_x (A \rightarrow B) \quad x}{A \rightarrow B} \quad u_1: A}{B} \quad u_2: \neg B}{\neg \forall_x \neg A} \quad \frac{\frac{\perp}{\neg A} \rightarrow^+ u_1}{\forall_x \neg A}}{\frac{\perp}{\neg \neg B} \rightarrow^+ u_2} \quad \frac{\neg \neg B \rightarrow B}{B}}{B}$$

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

$$\frac{\frac{\frac{\frac{\forall_x \neg (A \rightarrow B) \quad c}{\neg (A \rightarrow B_0)} \quad \frac{\frac{\perp \rightarrow B_0}{B_0} \quad \perp}{A \rightarrow B_0} \rightarrow^+ u_2}{\perp} \quad \frac{\frac{\frac{\frac{\forall_x \neg (A \rightarrow B) \quad x \quad u_1: B}{\neg (A \rightarrow B)} \quad A \rightarrow B}{\tilde{\exists}_x B} \quad u_2: A}{\tilde{\exists}_x B} \quad \frac{\frac{\perp}{\neg B} \rightarrow^+ u_1}{\forall_x \neg B}}{\perp}}{\perp}$$

(4)

$$\begin{array}{c}
\frac{\frac{\forall_x \neg B \quad x}{\neg B} \quad \frac{\frac{u_1: A \rightarrow B \quad A}{B}}{B}}{\frac{\frac{\perp}{\neg(A \rightarrow B)} \rightarrow^+ u_1}{\forall_x \neg(A \rightarrow B)}} \\
\frac{\tilde{\exists}_x(A \rightarrow B)}{\perp}
\end{array}$$

(5)

$$\begin{array}{c}
\frac{\frac{\frac{\frac{u_2: \neg B \quad \frac{\frac{u_1: A \rightarrow B \quad \frac{\forall_x A \quad x}{A}}{B}}{B}}{\frac{\perp}{\neg(A \rightarrow B)} \rightarrow^+ u_1}}{\forall_x \neg(A \rightarrow B)}}{\tilde{\exists}_x(A \rightarrow B)} \\
\frac{\frac{\neg \neg B \rightarrow B}{B} \quad \frac{\frac{\perp}{\neg \neg B} \rightarrow^+ u_2}{B}}{B}
\end{array}$$

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

$$\begin{array}{c}
\frac{\frac{\frac{\forall_y(\perp \rightarrow Ay) \quad y}{\perp \rightarrow Ay} \quad \frac{u_1: \neg Ax \quad u_2: Ax}{\perp}}{Ay}}{\frac{\forall_x Ax \rightarrow B}{\forall_y Ay}} \\
\frac{\frac{\frac{\forall_x \neg(Ax \rightarrow B) \quad x}{\neg(Ax \rightarrow B)} \quad \frac{\frac{\forall_x Ax \rightarrow B}{B} \quad \frac{B}{Ax \rightarrow B} \rightarrow^+ u_2}}{\frac{\perp}{\neg \neg Ax} \rightarrow^+ u_1}}
\end{array}$$

Using this derivation M we obtain

$$\begin{array}{c}
\frac{\frac{\frac{\forall_x(\neg \neg Ax \rightarrow Ax) \quad x}{\neg \neg Ax \rightarrow Ax} \quad \neg \neg Ax}{Ax}}{\frac{\forall_x Ax \rightarrow B}{\forall_x Ax}} \\
\frac{\frac{\frac{\forall_x \neg(Ax \rightarrow B) \quad x}{\neg(Ax \rightarrow B)} \quad \frac{\frac{B}{Ax \rightarrow B}}{B}}{\perp}
\end{array}$$

Since clearly $\vdash (\neg \neg A \rightarrow A) \rightarrow \perp \rightarrow A$ the claim follows. $\square$

REMARK. An immediate consequence of (6) is the classical derivability of the “drinker formula” $\tilde{\exists}_x(Px \rightarrow \forall_x Px)$, 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.

$$\begin{aligned} \vdash_c (\tilde{\exists}_x A \rightarrow B) &\leftrightarrow \forall_x (A \rightarrow B) \quad \text{if } x \notin \text{FV}(B), \\ \vdash_i (A \rightarrow \tilde{\exists}_x B) &\leftrightarrow \tilde{\exists}_x (A \rightarrow B) \quad \text{if } x \notin \text{FV}(A), \\ \vdash_c \tilde{\exists}_x (A \rightarrow B) &\leftrightarrow (\forall_x A \rightarrow B) \quad \text{if } x \notin \text{FV}(B). \end{aligned}$$

There is a similar lemma on weak disjunction:

LEMMA. *The following are derivable.*

- (7) $(A \tilde{\vee} B \rightarrow C) \rightarrow (A \rightarrow C),$
 $(A \tilde{\vee} B \rightarrow C) \rightarrow (B \rightarrow C),$
- (8) $(\neg\neg C \rightarrow C) \rightarrow (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow A \tilde{\vee} B \rightarrow C,$
- (9) $(\perp \rightarrow B) \rightarrow (A \rightarrow B \tilde{\vee} C) \rightarrow (A \rightarrow B) \tilde{\vee} (A \rightarrow C),$
- (10) $(A \rightarrow B) \tilde{\vee} (A \rightarrow C) \rightarrow A \rightarrow B \tilde{\vee} C,$
- (11) $(\neg\neg C \rightarrow C) \rightarrow (A \rightarrow C) \tilde{\vee} (B \rightarrow C) \rightarrow A \rightarrow B \rightarrow C,$
- (12) $(\perp \rightarrow C) \rightarrow (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow C) \tilde{\vee} (B \rightarrow C).$

PROOF. We only consider (8) and (12); the rest is left as an exercise.

(8)

$$\frac{\frac{\frac{\neg\neg C \rightarrow C}{C} \quad \frac{\frac{\frac{\frac{\neg A \rightarrow \neg B \rightarrow \perp}{\neg B \rightarrow \perp} \quad \frac{\frac{\perp}{\neg A} \rightarrow^+ u_2}}{\neg A \rightarrow \neg B \rightarrow \perp} \quad \frac{\frac{\frac{A \rightarrow C \quad u_2 : A}{C}}{u_1 : \neg C} \quad \frac{\frac{\frac{\perp}{\neg B} \rightarrow^+ u_3}{\perp}}{u_1 : \neg C} \quad \frac{\frac{B \rightarrow C \quad u_3 : B}{C}}{u_1 : \neg C}}{\frac{\perp}{\neg\neg C} \rightarrow^+ u_1}}{\neg\neg C \rightarrow C} \quad \frac{\perp}{\neg\neg C} \rightarrow^+ u_1}{C}$$

(12)

$$\frac{\frac{\frac{\frac{\perp \rightarrow C}{\perp} \quad \frac{\frac{\frac{A \rightarrow B \rightarrow C \quad u_1 : A}{B \rightarrow C} \quad \frac{u_2 : B}{C}}{A \rightarrow C} \rightarrow^+ u_1}{\neg(A \rightarrow C)} \quad \frac{\perp}{\perp}}{\frac{\perp}{\perp} \rightarrow C} \quad \frac{\frac{C}{B \rightarrow C} \rightarrow^+ u_2}{\neg(B \rightarrow C)} \quad \frac{\perp}{\perp}}{\perp}$$

The general idea here is to view $\tilde{\vee}$ as a finitary version of $\tilde{\exists}$. □

COROLLARY.

$$\begin{aligned} &\vdash (A \tilde{\vee} B \rightarrow C) \rightarrow (A \rightarrow C), \quad \vdash (A \tilde{\vee} B \rightarrow C) \rightarrow (B \rightarrow C), \\ &\vdash_c (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow A \tilde{\vee} B \rightarrow C, \\ &\vdash_i (A \rightarrow B \tilde{\vee} C) \leftrightarrow (A \rightarrow B) \tilde{\vee} (A \rightarrow C), \\ &\vdash_c (A \rightarrow C) \tilde{\vee} (B \rightarrow C) \leftrightarrow (A \rightarrow B \rightarrow C). \end{aligned}$$

1.2. Derivation terms, normalization

A derivation in normal form does not make “detours”, or more precisely, it cannot occur that an elimination rule immediately follows an introduction rule. We use “conversions” to remove such “local maxima” of complexity, thus reducing any given derivation to normal form.

1.2.1. The Curry-Howard correspondence. Since natural deduction derivations can be notationally cumbersome, it will be convenient to represent them 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 (1972) 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 $\rightarrow, \forall$ -rules in Table 1 where for clarity we have written the corresponding derivations to the left. This can be extended to the rules for $\vee, \wedge$ and $\exists$, but we will not do this here.

Every derivation term carries a formula as its type. However, we shall usually leave these formulas implicit and write derivation terms without them.

Every derivation term can be written uniquely in one of the forms

$$u\vec{M} \mid \lambda_v M \mid (\lambda_v M)N\vec{L},$$

where u is an assumption variable or assumption constant, v is an assumption variable or object variable, and M, N, L are derivation terms or object terms.

Notice that the final form $(\lambda_v M)N\vec{L}$ of a derivation term is not “normal”: $(\lambda_v M)N\vec{L}$ is called a β -*redex* (for “reducible expression”). It can be reduced by a “conversion”, which removes a “detour” in a derivation.

Derivation	Term
$u : A$	u^A
$\frac{[u : A] \quad M \quad \frac{B}{A \rightarrow B}}{\rightarrow^+ u}$	$(\lambda_{u^A} M^B)^{A \rightarrow B}$
$\frac{ M \quad \frac{A \rightarrow B}{B} \quad N \quad A}{\rightarrow^-}$	$(M^{A \rightarrow B} N^A)^B$
$\frac{ M \quad \frac{A}{\forall_x A}}{\forall^+ x} \quad (\text{with var.cond.})$	$(\lambda_x M^A)^{\forall_x A} \quad (\text{with var.cond.})$
$\frac{ M \quad \frac{\forall_x A(x) \quad t}{A(t)}}{\forall^-}$	$(M^{\forall_x A(x)} t)^{A(t)}$

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

1.2.2. Normalization. A *conversion* removes an elimination immediately following an introduction. We consider the following conversions, for derivations written in tree notation and also as derivation terms.

$\rightarrow$ -conversion.

$$\frac{[u : A] \quad | M \quad \frac{B}{A \rightarrow B}}{\rightarrow^+ u} \quad \frac{| N \quad A}{\rightarrow^-} \quad \mapsto_\beta \quad \frac{| N \quad A \quad | M \quad B}{\rightarrow^-}$$

or written as derivation terms

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

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

$\forall$ -conversion.

$$\frac{\frac{\frac{| M}{A(x)} \quad \forall^+ x}{\forall_x A(x)} \quad t}{A(t)} \quad \forall^- \quad \mapsto_\beta \quad \frac{| M'}{A(t)}$$

or written as derivation terms

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

The *closure* of the conversion relation $\mapsto_\beta$ is defined by

- (a) If $M \mapsto_\beta M'$, then $M \mapsto M'$.
- (b) If $M \mapsto M'$, then also $MN \mapsto M'N$, $NM \mapsto NM'$, $\lambda_v M \mapsto \lambda_v M'$ (*inner reductions*).

Therefore $M \mapsto N$ means that M *reduces in one step to* N , i.e., N is obtained from M by replacement of (an occurrence of) a redex M' of M by a conversum M'' of M' , i.e., by a single conversion.

EXAMPLE. Consider assumption variables

$$\begin{array}{lll} x: A \rightarrow (B \rightarrow A) \rightarrow A & u: A & u': A \\ y: A \rightarrow B \rightarrow A & v: B \rightarrow A & v': B \\ z: A & & \end{array}$$

Then we have derivation terms

$$\begin{aligned} S &:= \lambda_x \lambda_y \lambda_z (xz(yz))^A : (A \rightarrow (B \rightarrow A) \rightarrow A) \rightarrow (A \rightarrow B \rightarrow A) \rightarrow A \rightarrow A \\ K &:= \lambda_u \lambda_v u : A \rightarrow (B \rightarrow A) \rightarrow A \\ K' &:= \lambda_{u'} \lambda_{v'} u' : A \rightarrow B \rightarrow A \end{aligned}$$

By the one step reduction relation we obtain

$$\begin{aligned} SKK' &\mapsto (\lambda_x \lambda_y \lambda_z (xz(yz))) (\lambda_u \lambda_v u) (\lambda_{u'} \lambda_{v'} u') \mapsto \\ &(\lambda_y \lambda_z ((\lambda_u \lambda_v u) z(yz))) (\lambda_{u'} \lambda_{v'} u') \mapsto \\ &(\lambda_y \lambda_z ((\lambda_v z)(yz))) (\lambda_{u'} \lambda_{v'} u') \mapsto \\ &(\lambda_y \lambda_z z) (\lambda_{u'} \lambda_{v'} u') \mapsto \lambda_z z. \end{aligned}$$

A term M is *in normal form*, or M is *normal*, if M does not contain a redex. A *reduction sequence* is a (finite or infinite) sequence $M_0 \mapsto M_1 \mapsto M_2 \dots$ such that $M_i \mapsto M_{i+1}$, for all i . Finite reduction sequences are partially ordered under the initial part relation; the collection of finite reduction sequences starting from a term M forms a tree, the *reduction tree* of M . The

branches of this tree may be identified with the collection of all infinite and all terminating finite reduction sequences. A term is *strongly normalizing* if its reduction tree is finite.

THEOREM 1.2.1. *Every derivation term is strongly normalizing, and the final element of each reduction sequence is uniquely determined.*

For time reasons the proof is omitted. It can be found for instance in Troelstra and van Dalen (1988).

CHAPTER 2

Computable functionals

The objects studied in mathematics have types, which in many cases are function types, possibly of a higher type. We are particularly interested in computable such objects. Their domains are partial continuous functionals (Scott, Ersov) which we study first, via Scott's information systems.

2.1. 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).

2.1.1. Ideals.

DEFINITION. An *information system* is a structure $(A, \text{Con}, \vdash)$ 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 $\text{Con} \times A$ (the *entailment* relation), which satisfy

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

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

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

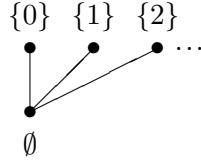
DEFINITION. The *ideals* (also called *objects*) of an information system $\mathbf{A} = (A, \text{Con}, \vdash)$ are defined to be those subsets x of A which satisfy

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

We write $x \in |\mathbf{A}|$ to mean that x is an ideal of $\mathbf{A}$.

EXAMPLES. The *deductive closure* $\overline{U} := \{a \in A \mid U \vdash a\}$ of $U \in \text{Con}$ is an ideal.

Every countable set A can be turned into a “flat” information system by letting the set of tokens be A , $\text{Con} := \{\emptyset\} \cup \{\{a\} \mid a \in A\}$ 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

$$\begin{aligned} \text{Con} &:= \{ \{ (a_i, b_i) \mid i < k \} \mid \forall_{i,j < k} (a_i = a_j \rightarrow b_i = b_j) \}, \\ U \vdash (a, b) &:= (a, b) \in U. \end{aligned}$$

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 $\mathbf{A} = (A, \text{Con}, \vdash)$ the structure $(|\mathbf{A}|, \subseteq, \overline{})$ is a “domain” (also called Scott-Ershov domain, or “bounded complete algebraic cpo”), whose set of “compact elements” can be represented as $|\mathbf{A}|_c = \{ \overline{U} \mid U \in \text{Con} \}$. 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 (cf. Abramsky and Jung (1994)), and hence not even define these notions here.

2.1.2. Function spaces. We define the “function space” $\mathbf{A} \rightarrow \mathbf{B}$ between two information systems $\mathbf{A}$ and $\mathbf{B}$.

DEFINITION. Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ be information systems. Define $\mathbf{A} \rightarrow \mathbf{B} = (C, \text{Con}, \vdash)$ by

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

$$\{(U_i, b_i) \mid i \in I\} \in \text{Con} := \forall_{J \subseteq I} \left(\bigcup_{j \in J} U_j \in \text{Con}_A \rightarrow \{b_j \mid j \in J\} \in \text{Con}_B \right).$$

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

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

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 $(WU' \subseteq WU, \text{ hence also } WU \vdash_B WU')$. 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 $\mathbf{A}$ and $\mathbf{B}$ are information systems, then so is $\mathbf{A} \rightarrow \mathbf{B}$ defined as above.*

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

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

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

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

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

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

Since

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

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

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

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

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

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. $\square$

We shall now give an alternative characterization of the ideals in $\mathbf{A} \rightarrow \mathbf{B}$, as “approximable maps”. The basic idea for approximable maps is the desire to study “information respecting” maps from $\mathbf{A}$ into $\mathbf{B}$. 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 $U \in \text{Con}_A$, then we know that at least the token b appears in the value.

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

- (a) if $(U, b_1), \dots, (U, b_n) \in r$, then $\{b_1, \dots, b_n\} \in \text{Con}_B$;
- (b) if $(U, b_1), \dots, (U, b_n) \in r$ and $\{b_1, \dots, b_n\} \vdash_B b$, 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 $\mathbf{A}$ and $\mathbf{B}$ be information systems. Then the ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are exactly the approximable maps from $\mathbf{A}$ to $\mathbf{B}$.*

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

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

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

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

by the definition of the entailment relation $\vdash$ in $\mathbf{A} \rightarrow \mathbf{B}$, 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 $r \subseteq \text{Con}_A \times B$ is an approximable map. We must show that $r \in |\mathbf{A} \rightarrow \mathbf{B}|$.

Consistency of r . Suppose $(U_1, b_1), \dots, (U_n, b_n) \in r$ and $U = \bigcup \{U_i \mid i \in I\} \in \text{Con}_A$ for some $I \subseteq \{1, \dots, n\}$. We must show that $\{b_i \mid i \in I\} \in \text{Con}_B$. 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 $\{b_i \mid i \in I\} \in \text{Con}_B$ by axiom (a).

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

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

We must show $(U, b) \in r$. By definition of $\vdash$ for $\mathbf{A} \rightarrow \mathbf{B}$ 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). $\square$

2.1.3. Continuous functions. We can also characterize approximable maps in a different way, which is closer to usual characterizations of continuity¹:

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

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

$$f\left(\bigcup_{x \in D} x\right) = \bigcup_{x \in D} f(x).$$

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

PROOF. Let f satisfy PFS, and $D \subseteq |\mathbf{A}|$ 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 $b \in f(\bar{U})$ for some $U \subseteq \bigcup_{x \in D} x$. From the

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

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

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

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

We call a function $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ continuous if it satisfies the conditions in Lemma 2.1.3. Hence continuous maps $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract objects $x \in |\mathbf{A}|$ and $f(x) \in |\mathbf{B}|$: 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 $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ and $g: |\mathbf{B}| \rightarrow |\mathbf{C}|$.

THEOREM 2.1.4. *Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ be information systems. Then the ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are in a natural bijective correspondence with the continuous functions from $|\mathbf{A}|$ to $|\mathbf{B}|$, as follows.*

- (a) *With any approximable map $r \subseteq \text{Con}_A \times B$ we can associate a continuous function $|r|: |\mathbf{A}| \rightarrow |\mathbf{B}|$ 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 $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ we can associate an approximable map $\hat{f} \subseteq \text{Con}_A \times B$ 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 $\mathbf{A} \rightarrow \mathbf{B}$; then by Theorem 2.1.2 we know that r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\mathbf{A}|$.

$|r|(z)$ 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 $\{b_1, \dots, b_n\} \in \text{Con}_B$.

$|r|(z)$ 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 $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ be continuous. It is easy to verify that $\hat{f}$ is indeed an approximable map. Furthermore

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

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

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

hence $r = \widehat{|r|}$. □

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

Equality of two subsets $r, s \subseteq \text{Con}_A \times B$ 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 $U \in \text{Con}_A$.

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

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

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

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

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.

2.2. Partial continuous functionals

We define types, built from base types by the formation of function types, $\tau \rightarrow \sigma$; types may involve type variables. For every closed type τ we define an information system $\mathbf{C}_\tau$ whose ideals will be the partial continuous functionals of type τ .

2.2.1. Types. Types will be built from “algebras” (given by a list of “constructor types”) by the formation of “function types” $\tau \rightarrow \sigma$. Types may involve type variables $\alpha, \beta, \gamma, \xi, \zeta$. Iterated arrows are understood as associated to the right. For example, $\alpha \rightarrow \beta \rightarrow \gamma$ means $\alpha \rightarrow (\beta \rightarrow \gamma)$, not $(\alpha \rightarrow \beta) \rightarrow \gamma$.

DEFINITION. *Constructor types* κ have the form

$$\vec{\alpha} \rightarrow (\xi)_{i < n} \rightarrow \xi$$

with all type variables α_i distinct from each other and from ξ . 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 is *recursive* if it has a recursive argument type. 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.

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

Algebra forms with parameters are

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

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

DEFINITION (Type).

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

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

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

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. $\mathbb{R} := (\mathbb{N} \rightarrow \mathbb{Q}) \times (\mathbb{P} \rightarrow \mathbb{N})$ 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.

2.2.2. The information system $\mathbf{C}_\tau$. For every closed type τ we define the information system $\mathbf{C}_\tau = (C_\tau, \text{Con}_\tau, \vdash_\tau)$. The ideals $x \in |\mathbf{C}_\tau|$ are the *partial continuous functionals* of type τ . Since we will have $\mathbf{C}_{\tau \rightarrow \sigma} = \mathbf{C}_\tau \rightarrow \mathbf{C}_\sigma$, the partial continuous functionals of type $\tau \rightarrow \sigma$ will correspond to the continuous functions from $|\mathbf{C}_\tau|$ to $|\mathbf{C}_\sigma|$.

DEFINITION (Information system of a closed type τ). We simultaneously define C_ι , $C_{\tau \rightarrow \sigma}$, Con_ι and $\text{Con}_{\tau \rightarrow \sigma}$.

- (a) The *tokens* $a \in C_\iota$ are the type correct constructor expressions $Ca_1^* \dots a_n^*$ where a_i^* is an *extended token*, i.e., a token or the special symbol $*$ which carries no information.
- (b) The tokens in $C_{\tau \rightarrow \sigma}$ are the pairs (U, b) with $U \in \text{Con}_\tau$ and $b \in C_\sigma$.
- (c) A finite set U of tokens in C_ι is *consistent* (i.e., $U \in \text{Con}_\iota$) if all its elements start with the same constructor C , say of arity $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \iota$, and all $U_i \in \text{Con}_{\tau_i}$ for $i = 1, \dots, n$, where U_i consists of all (proper) tokens at the i -th argument position of some token in $U = \{Ca_1^*, \dots, Ca_m^*\}$.
- (d) $\{(U_i, b_i) \mid i \in I\} \in \text{Con}_{\tau \rightarrow \sigma}$ is defined to mean

$$\forall J \subseteq I \left(\bigcup_{j \in J} U_j \in \text{Con}_\tau \rightarrow \{b_j \mid j \in J\} \in \text{Con}_\sigma \right).$$

Building on this definition, we define $U \vdash_\tau a$ for $U \in \text{Con}_\tau$ and $a \in C_\tau$.

- (e) $\{Ca_1^*, \dots, Ca_m^*\} \vdash_\iota C'a^*$ is defined to mean $C = C'$, $m \geq 1$ and $U_i \vdash a_i^*$, with U_i as in (c) above (and $U \vdash *$ taken to be true).
- (f) $W \vdash_{\tau \rightarrow \sigma} (U, b)$ is defined to mean $WU \vdash_\sigma b$, where application WU of $W = \{(U_i, b_i) \mid i \in I\} \in \text{Con}_{\tau \rightarrow \sigma}$ to $U \in \text{Con}_\tau$ is defined to be $\{b_i \mid U \vdash_\tau U_i\}$; recall that $U \vdash V$ abbreviates $\forall_{a \in V} (U \vdash a)$.

Note that the present definition is by recursion on the *height* of the syntactic expressions involved:

$$\begin{aligned}
|\alpha| &:= 0, \\
|\iota(\tau_1 \dots \tau_n)| &:= 1 + \max\{|\tau_i| \mid i = 1, \dots, n\}, \\
|\tau \rightarrow \sigma| &:= \max\{1 + |\tau|, 1 + |\sigma|\}, \\
|Ca_1^* \dots a_n^*| &:= 1 + \max\{|a_i^*| \mid i = 1, \dots, n\}, \\
|*| &:= 0, \\
|(U, b)| &:= \max\{1 + |U|, 1 + |b|\}, \\
|\{a_i \mid i \in I\}| &:= \max\{1 + |a_i| \mid i \in I\}, \\
|U \vdash a| &:= \max\{1 + |U|, 1 + |a|\}.
\end{aligned}$$

It is easy to see that $(C_\tau, \text{Con}_\tau, \vdash_\tau)$ is an information system. Observe that all the notions involved are computable: $a \in C_\tau$, $U \in \text{Con}_\tau$ and $U \vdash_\tau a$.

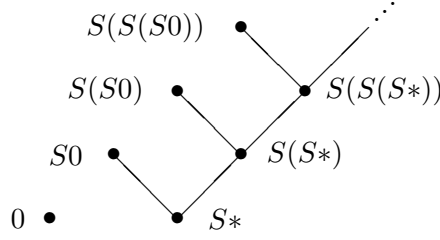
DEFINITION (Partial continuous functionals). For every closed type τ let $\mathbf{C}_\tau$ be the information system $(C_\tau, \text{Con}_\tau, \vdash_\tau)$. The set $|\mathbf{C}_\tau|$ of ideals in $\mathbf{C}_\tau$ is the set of *partial continuous functionals* of type τ . A partial continuous functional $x \in |\mathbf{C}_\tau|$ is *computable* if it is recursively enumerable when viewed as a set of tokens.

Notice that $\mathbf{C}_{\tau \rightarrow \sigma} = \mathbf{C}_\tau \rightarrow \mathbf{C}_\sigma$ as defined generally for information systems.

For example, the tokens for the base type $\mathbb{N}$ 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). As another (more typical) example, consider the base type $\mathbb{Y}$ of binary trees with a nullary constructor $-$ and a binary Branch. Then $\{\text{Branch}(-, *), \text{Branch}(*, -)\}$ is consistent, and it entails $\text{Branch}(-, -)$.

Recall that a partial continuous functional is an ideal in $\mathbf{C}_\tau$. We will be interested in “cototal” and “total” ideals. To grasp what this means it helps to look at examples of tokens for some closed base types.

$$\begin{array}{ll}
\mathbb{B} & \mathbf{tt}, \quad \mathbf{ff} \\
\mathbb{N} & S^n 0, \quad S^{n+1} * \\
\mathbb{L}(\mathbb{B}) & \text{Cons}(\mathbf{tt}, \text{Cons}(*, \text{Cons}(\mathbf{ff}, [])))
\end{array}$$

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

$$\begin{aligned} \mathbb{S}(\mathbb{B}) & \quad \text{SCons}(\mathbf{tt}, \text{SCons}(*, \text{SCons}(\mathbf{ff}, *))) =: a \\ \mathbb{L}(\mathbb{S}(\mathbb{B})) & \quad \text{Cons}(a, \text{Cons}(a, [])). \end{aligned}$$

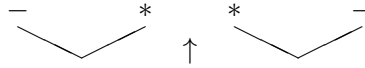
Generally these tokens can be described as terms built from the constructors involved in the used base types and possibly the symbol $*$.

DEFINITION (Cototal and total ideals of a closed base type). Let ι be a closed base type. Its tokens can be seen as constructor terms with some argument positions occupied by $*$. An ideal x in $\mathbf{C}_\iota$ is *cototal* if for each of its tokens $t(*)$ with a distinguished occurrence of $*$ there is another token of the form $t(C\vec{*})$ in x . Here C is a constructor for the base type of $*$'s position in $t(*)$. We call x *total* if it is cototal and finite.

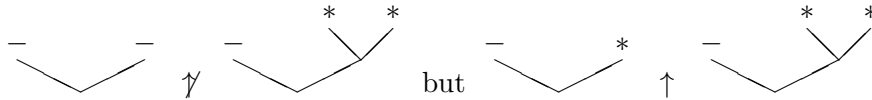
We give some examples, with a more pictorial representation. Tokens in $\mathbf{C}_\mathbb{Y}$:



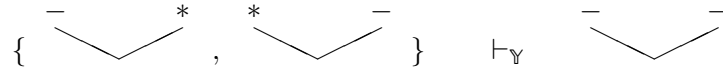
Consistency in $\mathbf{C}_\mathbb{Y}$:



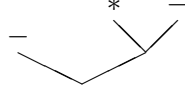
Moreover



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



(6) Closure of



Among these are

- (1) – (4) cototal ideals,
- (5) a total ideal,
- (6) a neither total nor cototal ideal.

2.2.3. Constructors as continuous functions. Let ι be a closed base type. Every constructor C generates the following ideal in the function space determined by the type of the constructor:

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

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

According to the general definition of a continuous function associated to an ideal in a function space the continuous map $|r_C|$ satisfies

$$|r_C|(\vec{x}) = \{ C\vec{a}^* \mid \exists \vec{U} \subseteq \vec{x} (\vec{U} \vdash \vec{a}^*) \}.$$

(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.

LEMMA 2.2.1 (Constructors are injective and have disjoint ranges). *Let ι be a closed base type and C be a constructor of ι . Then*

$$|r_C|(\vec{x}) \subseteq |r_C|(\vec{y}) \leftrightarrow \vec{x} \subseteq \vec{y}.$$

If C_1, C_2 are distinct constructors of ι , then $|r_{C_1}|(\vec{x}) \neq |r_{C_2}|(\vec{y})$, 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

$$\begin{aligned} |r_C|(\bar{\emptyset}, y) &= \bar{\emptyset} = |r_C|(x, \bar{\emptyset}), \\ |r_{C_1}|(\bar{\emptyset}) &= \bar{\emptyset} = |r_{C_2}|(\bar{\emptyset}) \end{aligned}$$

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

LEMMA (Ideals of a closed base type). *Every non-empty ideal in the information system associated to a closed base type has the form $|r_C|(\vec{x})$ with a constructor C and ideals $\vec{x}$.*

PROOF. Let z be a non-empty ideal and $Ca_0^*b_0^* \in z$, where for simplicity we assume that C is a binary constructor. Let $x := \{a \mid Ca^* \in z\}$ and $y := \{b \mid C^*b \in z\}$; clearly x, y are ideals. We claim that $z = |r_C|(x, y)$. For $\supseteq$ consider Ca^*b^* with $a^* \in x \cup \{*\}$ and $b^* \in y \cup \{*\}$. Then by definition $\{Ca^*, C^*b^*\} \subseteq z$, hence $Ca^*b^* \in z$ by deductive closure. Conversely, notice that an arbitrary element of z must have the form Ca^*b^* , because of consistency. Then $\{Ca^*, C^*b^*\} \subseteq z$ again by deductive closure. Hence $a^* \in x \cup \{*\}$ and $b^* \in y \cup \{*\}$, and therefore $Ca^*b^* \in |r_C|(x, y)$. $\square$

It is in this proof that we need entailment to be a relation between finite sets of tokens and single tokens, not just a binary relation between tokens. Information systems with the latter property are called *atomic*. For comparison, Appendix A contains the development of the first part of the present chapter, based on atomic coherent information systems.

The information systems $\mathbf{C}_\tau$ enjoy the pleasant property of *coherence*, which amounts to the possibility to locate inconsistencies in two-element sets of data objects. Generally, an information system $\mathbf{A} = (A, \text{Con}, \vdash)$ is *coherent* if it satisfies: $U \subseteq A$ is consistent if and only if all of its two-element subsets are.

LEMMA 2.2.2. *Let $\mathbf{A}$ and $\mathbf{B}$ be information systems. If $\mathbf{B}$ is coherent, then so is $\mathbf{A} \rightarrow \mathbf{B}$.*

PROOF. Exercise. $\square$

2.3. Terms

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^+ .

2.3.1. 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:

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

The set $FV(M)$ of free variables of a term M is defined by

$$\begin{aligned} FV(x) &:= \{x\}, & FV(C) &:= FV(D) := \emptyset, \\ FV(\lambda_x M) &:= FV(M) \setminus \{x\}, & FV(MN) &:= FV(M) \cup FV(N). \end{aligned}$$

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

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

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

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

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

$$(13) \quad D\vec{P}_i(\vec{y}_i) = M_i \quad (i = 1, \dots, n \text{ where } n \geq 0)$$

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-unifiable² (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 $\text{ar}(D)$. A substitution instance of a left hand side of (13) 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 $C\vec{P}$.

²A detailed treatment of unification is in Appendix B

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 $\perp_\iota$ of type ι with no computation rules. The boolean connectives `andb`, `impb` and `orb` are defined by

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

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

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

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

and similarly for $\mathbb{Y}$ with constructors `0` (leaf) and `C` (branch)

$$\begin{array}{ll} (0 =_{\mathbb{Y}} 0) = \mathbf{tt}, & (Cts =_{\mathbb{Y}} 0) = \mathbf{ff}, \\ (0 =_{\mathbb{Y}} Cts) = \mathbf{ff}, & (Cts =_{\mathbb{Y}} Ct's') = (t =_{\mathbb{Y}} t' \text{ andb } s =_{\mathbb{Y}} s'). \end{array}$$

For $\mathbb{L}(\iota)$ with ι a closed base type we define $=_{\mathbb{L}(\iota)}: \mathbb{L}(\iota) \rightarrow \mathbb{L}(\iota) \rightarrow \mathbb{B}$ by

$$\begin{array}{ll} ([] =_{\mathbb{L}(\iota)} []) = \mathbf{tt}, & (a::\ell = []) =_{\mathbb{L}(\iota)} \mathbf{ff}, \\ ([] =_{\mathbb{L}(\iota)} a::\ell) = \mathbf{ff}, & (a::\ell =_{\mathbb{L}(\iota)} a'::\ell') = (a =_\iota a' \text{ andb } \ell =_{\mathbb{L}(\iota)} \ell'). \end{array}$$

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

$$\begin{array}{ll} \text{Double}(0) & := 0, \\ \text{Double}(S(n)) & := S(S(\text{Double}(n))). \end{array}$$

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

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

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 base type of streams defined by

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

with a single constructor $C: \alpha \rightarrow \mathbb{S}(\alpha) \rightarrow \mathbb{S}(\alpha)$. We write $C_a u$ or $a :: u$ for Cau . This base type 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 $\{C_a^n(*) \mid n \geq 1\}$. We define the function Map of type $(\tau \rightarrow \sigma) \rightarrow \mathbb{S}(\tau) \rightarrow \mathbb{S}(\sigma)$ mapping its function argument $h: \tau \rightarrow \sigma$ over a stream u of type $\mathbb{S}(\tau)$ by the computation rule

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

2.3.2. Recursion operators. Important examples of such constants D are the (structural) higher type *recursion operators* $\mathcal{R}_l^\tau$ 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 $\mathbb{N} \rightarrow \tau \rightarrow (\mathbb{N} \rightarrow \tau \rightarrow \tau) \rightarrow \tau$. 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

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

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

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

Here ι is the type of the recursion argument, and each $(\rho_{i\nu}(\iota \times \tau))_{\nu < n_i} \rightarrow \tau$ 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 $\mathcal{R}_\iota^\tau$). Let

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

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

$$\mathcal{R}(C_i \vec{x}) \vec{f} := f_i \vec{x} (\mathcal{R} x_0^R \vec{f}) \dots (\mathcal{R} x_{n-1}^R \vec{f}).$$

EXAMPLES.

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

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 $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu < n_i} \rightarrow \xi)_{i < k}$ and result type τ the type of the cases operator $\mathcal{C}_\iota^\tau$ is

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

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

$$\mathcal{C}_\mathbb{N}^\tau: \mathbb{N} \rightarrow \tau \rightarrow (\mathbb{N} \rightarrow \tau) \rightarrow \tau.$$

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

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

However, it is easier to define the predecessor function by the computation rules $P0 := 0$ and $P(Sn) := n$.

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 is $[\mathbf{if} \ m \ 0 \ \lambda_n n]$ (which is often written in the form $[\mathbf{case} \ m \ \mathbf{of} \ 0 \mid \lambda_n n]$).

2.3.3. 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

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

The product of these k constructor types is isomorphic to

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

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

$$\iota \rightarrow \left(\sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota \times \tau) \rightarrow \tau \right) \rightarrow \tau.$$

This way of subsuming the types of all constructors of an algebra into a single type suggests the following definition of the *destructor* $\mathcal{D}_\iota$ of an algebra ι .

DEFINITION (Destructor). For the algebra $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu < n_i} \rightarrow \xi)_{i < k}$ we define the type of the destructor $\mathcal{D}_\iota$ to be

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

The computation rules for $\mathcal{D}_\iota$ disassemble a constructor-built pattern into its parts. For instance, the computation rules for $\mathcal{D}_\mathbb{N}$ of type $\mathbb{N} \rightarrow \text{ysum}(\mathbb{N})$ (or $\mathbb{U} + \mathbb{N}$) are

$$\begin{aligned} \mathcal{D}_\mathbb{N}(0) &:= \text{DummyL}, \\ \mathcal{D}_\mathbb{N}(S(n)) &:= \text{Inr}(n). \end{aligned}$$

The computation rules for $\mathcal{D}_{\mathbb{P}}$ of type $\mathbb{P} \rightarrow \text{uysum}(\mathbb{P} + \mathbb{P})$ (or $\mathbb{U} + (\mathbb{P} + \mathbb{P})$) are

$$\begin{aligned}\mathcal{D}_{\mathbb{P}}(1) &:= \text{DummyL}, \\ \mathcal{D}_{\mathbb{P}}(S_0(p)) &:= \text{Inr}(\text{InL}^{\mathbb{P} \rightarrow \mathbb{P} + \mathbb{P}}(p)), \\ \mathcal{D}_{\mathbb{P}}(S_1(p)) &:= \text{Inr}(\text{InR}^{\mathbb{P} \rightarrow \mathbb{P} + \mathbb{P}}(p)).\end{aligned}$$

The corecursion operator ${}^{\text{co}}\mathcal{R}_{\iota}^{\tau}$ is used to construct a map from τ to ι by “corecursion” on the structure of ι . Its type is

$$(14) \quad \tau \rightarrow \left(\tau \rightarrow \sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota + \tau) \right) \rightarrow \iota.$$

EXAMPLES (Types and computation rules of corecursion operators).

$$\begin{aligned}{}^{\text{co}}\mathcal{R}_{\mathbb{B}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + \mathbb{U}) \rightarrow \mathbb{B}, \\ {}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + (\mathbb{N} + \tau)) \rightarrow \mathbb{N}, \\ {}^{\text{co}}\mathcal{R}_{\mathbb{P}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + ((\mathbb{P} + \tau) + (\mathbb{P} + \tau))) \rightarrow \mathbb{P}, \\ {}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + (\mathbb{Y} + \tau) \times (\mathbb{Y} + \tau)) \rightarrow \mathbb{Y}, \\ {}^{\text{co}}\mathcal{R}_{\mathbb{L}(\rho)}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + \rho \times (\mathbb{L}(\rho) + \tau)) \rightarrow \mathbb{L}(\rho), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau} &: \tau \rightarrow (\tau \rightarrow \rho \times (\mathbb{S}(\rho) + \tau)) \rightarrow \mathbb{S}(\rho).\end{aligned}$$

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

$$\begin{aligned}{}^{\text{co}}\mathcal{R}_{\mathbb{B}}^{\tau} x f &:= [\lambda_{\text{tt}}, \lambda_{\text{ff}}](f x), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau} x f &:= [\lambda_0, \lambda_y(S([\text{id}^{\mathbb{N} \rightarrow \mathbb{N}}, P_{\mathbb{N}}]y))](f x), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{P}}^{\tau} x f &:= [\lambda_1, \lambda_y(S_0([\text{id}, P_{\mathbb{P}}]y)), \lambda_y(S_1([\text{id}, P_{\mathbb{P}}]y))](f x), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau} x f &:= [\lambda_0, \lambda_{y_0, y_1}(C([\text{id}, P_{\mathbb{Y}}]y_0)([\text{id}, P_{\mathbb{Y}}]y_1))](f x), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{L}(\rho)}^{\tau} x f &:= [\lambda_{\text{[]}}, \lambda_{y_0, y_1}(y_0 :: [\text{id}, P_{\mathbb{L}(\rho)}]y_1)](f x), \\ {}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau} x f &:= (f x)_0 :: [\text{id}, P_{\mathbb{S}(\rho)}](f x)_1\end{aligned}$$

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

DEFINITION. The (single) computation rule for ${}^{\text{co}}\mathcal{R}_{\iota}^{\tau}$ of type (14) is

$${}^{\text{co}}\mathcal{R}_{\iota}^{\tau} x f := [g_0, \dots, g_{k-1}](f x)$$

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

$$g_i := \lambda_{\bar{x}}(C_i(N_\nu)_{\nu < n_i}) \quad \text{with } x_\nu : \rho_{i\nu}(\iota + \tau),$$

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

and $P := \lambda_x({}^{\text{co}}\mathcal{R}_\iota^\tau x f)$ 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

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

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

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

Recall that Map of type $(\tau \rightarrow \sigma) \rightarrow \mathbb{S}(\tau) \rightarrow \mathbb{S}(\sigma)$ maps its function argument $h : \tau \rightarrow \sigma$ over a stream u of type $\mathbb{S}(\tau)$ (see Section 2.3.1, page 29). 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 $t_R := {}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^\tau x_0 f_0$ with τ, x_0 arbitrary (for instance $\tau := \mathbb{U}, x_0 := \text{Dummy}^{\mathbb{U}}$) and

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

For the leftmost path we similarly 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.

2.4. 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 2.3. 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 $\llbracket \lambda_{\vec{x}} M \rrbracket$, where M is a term with free variables among $\vec{x}$. Since this value is a token set, we can define inductively the relation $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$.

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

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

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

DEFINITION (Inductive, of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$).

$$\frac{U_i \vdash a}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket}(V), \quad \frac{(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}(A).$$

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

$$\frac{\vec{V} \vdash a^*}{(\vec{U}, \vec{V}, \mathbf{C}a^*) \in \llbracket \lambda_{\vec{x}} \mathbf{C} \rrbracket}(\mathbf{C}), \quad \frac{(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} M \rrbracket \quad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, a) \in \llbracket \lambda_{\vec{x}} D \rrbracket}(D)$$

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. Some of these proofs are given in Appendix C. First of all, one can prove that $\llbracket \lambda_{\vec{x}} M \rrbracket$ 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 $\rightarrow$ 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 $P\vec{t}$.

In a mathematical theory involving implication we need a notion of equality of formulas. The reason is that in the rule

$$\frac{\begin{array}{c} | M \qquad | N \\ A \rightarrow B \quad A \end{array}}{B} \rightarrow -$$

the two occurrences of A in the premises have to be “equal”. In our case equality of formulas boils down to equality of terms: $P\vec{t}$ and $P\vec{s}$ are equal if $\vec{t}$ and $\vec{s}$ are componentwise equal. Now for equality of terms we can make use of the fact that our defined constants D come with a list of computation rules. These rules are not required to terminate (corecursion is an example), but we can and do consider two terms to be equal if they have a common reduct. This is reasonable because by Section C.2 the denotation of a term is preserved under reduction.

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) $\forall_{n \in \mathbb{N}}(\ell \in T_{\mathbb{L}(\mathbb{N})} \rightarrow n :: \ell \in T_{\mathbb{L}(\mathbb{N})})$. An example for the latter is ${}^{\text{co}}T_{\mathbb{L}(\mathbb{N})}$ defined by a closure axiom saying that every $\ell \in {}^{\text{co}}T_{\mathbb{L}(\mathbb{N})}$ is of the form $n :: \ell'$ with $n \in \mathbb{N}$ and $\ell' \in {}^{\text{co}}T_{\mathbb{L}(\mathbb{N})}$ 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 $t \in {}^{\text{co}}T_{\mathbb{L}(\mathbb{N})}$ 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 closed base types considered in Section 2.2.2. A solution for a problem posed by the formula $A \rightarrow B$ 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 $X^{\text{nc}}, Y^{\text{nc}}, Z^{\text{nc}}, W^{\text{nc}} \dots$, and use $X, Y, Z, W \dots$ for both.

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

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

with all predicate variables $Y_i^c, Z_i^{\text{nc}}, W_i^{\text{nc}}$ occurring exactly once and distinct from each other and from X . By $\tilde{X}$ 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. 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 $I^c := \mu_{X^c} \tilde{K}$ and $I^{\text{nc}} := \mu_{X^{\text{nc}}} \tilde{K}$ with $\tilde{K}$ not empty *predicate forms* (and use I for both), and similarly with ${}^{\text{co}}I$ for I and ν for μ .

EXAMPLES. Recall that $\text{Nil}^{\mathbb{L}(\alpha)}$ and $\text{Cons}^{\alpha \rightarrow \mathbb{L}(\alpha) \rightarrow \mathbb{L}(\alpha)}$ 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

$$\begin{aligned} K_0 &:= X([], []), \\ K_1 &:= \forall_{x, x'}(Y(x, x') \rightarrow \forall_{\ell, \ell'}(X(\ell, \ell') \rightarrow X(x :: \ell, x' :: \ell')))) \end{aligned}$$

are clauses and both *similarity* $\sim_{\mathbb{L}}(Y)$ defined by $\mu_X(K_0, K_1)$ and *bisimilarity* $\approx_{\mathbb{L}}(Y)$ 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

$$\begin{aligned} K_0 &:= ([] \in X), \\ K_1 &:= \forall_x(x \in Y \rightarrow \forall_{\ell}(\ell \in X \rightarrow x :: \ell \in X)) \end{aligned}$$

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* ${}^{\text{co}}T_{\mathbb{L}}(Y)$ (or ${}^{\text{co}}T_{\mathbb{L},Y}$) 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

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

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

DEFINITION (Predicates and formulas).

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

with $I/{}^{\text{co}}I$ a predicate form. $(I/{}^{\text{co}}I)(\vec{\rho}, \vec{P})$ 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

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

We call a predicate or formula C *non-computational* (n.c., or *Harrop*) if its final predicate $\text{fp}(C)$ is of the form X^{nc} or I^{nc} , else *computationally relevant* (c.r.). We require that all predicate substitutions involved in $(I/{}^{\text{co}}I)(\vec{\rho}, \vec{P})$ 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 ${}^{\text{co}}I(\vec{\rho}, \vec{P})$ *coinductive*.

The terms $\vec{t}$ are those introduced in Section 2.3.1, 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 $C\vec{t}$ for $\{ \vec{y} \mid C\vec{t}\vec{y} \}$.

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 .

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

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

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

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

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

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 a algebra form (for instance $\mathbb{L}$), the ρ_i are closed algebras 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

$$\begin{aligned} K_0 &:= X^c(0, 0), \\ K_1 &:= \forall_{n, n'}(X^c(n, n') \rightarrow X^c(Sn, Sn')). \end{aligned}$$

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

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

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

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

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

$$\begin{aligned} K_0 &:= \forall_{x, y}(Yxy \rightarrow Xxy), \\ K_1 &:= \forall_{x, y, z}(Yxy \rightarrow Xyz \rightarrow Xxz) \end{aligned}$$

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

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

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 $\text{TC}(\mathbb{N}, <)$ is its transitive closure. The constructor types of $\tau(\text{TC})$ are $\beta \rightarrow \xi$ and $\beta \rightarrow \xi \rightarrow \xi$, hence $\tau(\text{TC}(\mathbb{N}, <))$ is the type of non-empty lists of natural numbers.

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

$$\text{Eq} := \mu_{X^{\text{nc}}}(\forall_x X^{\text{nc}}xx) \quad (\text{D for “inductively defined”}).$$

We will use the abbreviation

$$(x \equiv y) := \text{Eq}(x, y).$$

The missing logical connectives disjunction, conjunction and existence can now be defined inductively. Disjunction is a special case of union

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

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

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

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

$$\begin{aligned} \tau(\text{CupD}) &= \mu_{\xi}(\beta_0 \rightarrow \xi, \beta_1 \rightarrow \xi) = \beta_0 + \beta_1, \\ \tau(\text{CupL}) &= \mu_{\xi}(\beta \rightarrow \xi, \xi) = \beta + \mathbb{U} = \mathbf{ysumu}(\beta), \\ \tau(\text{CupR}) &= \mu_{\xi}(\xi, \beta \rightarrow \xi) = \mathbb{U} + \beta = \mathbf{uysum}(\beta), \\ \tau(\text{CupU}) &= \mu_{\xi}(\xi, \xi) = \mathbb{B}. \end{aligned}$$

We use the abbreviations

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

In case of nullary predicates we use

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

Since the “decoration” is determined by the c.r./n.c. status of the two parameter predicates we usually leave it out in $\vee^d, \vee^l, \vee^r, \vee^u$ 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 $\vee^{\text{nc}}$.

Similarly conjunction is a special case of intersection

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

and we obtain the variants

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

Then by definition

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

We use the abbreviations

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

In case of nullary predicates we use

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

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

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

Then by definition

$$\tau(\text{Ex}) = \mu_\xi(\beta \rightarrow \xi) = \mathbb{I}(\beta).$$

We use the abbreviation

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

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 $\rightarrow$ 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).

3.2.1. Axioms for inductive predicates. 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

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

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

$$(15) \quad I_i^+ : \forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu < n_i} \rightarrow I\vec{t}_i),$$

$$(16) \quad I^- : (\forall_{\vec{x}_i}((A_{i\nu}(I \cap X))_{\nu < n_i} \rightarrow X\vec{t}_i))_{i < k} \rightarrow I \subseteq X$$

($I \cap X$ was inductively defined above). (16) 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 $\forall_{\vec{x}_i}(A_{i\nu}(I \cap X))_{\nu < n_i} \rightarrow X\vec{t}_i$ rather than $\forall_{\vec{x}_i}(A_{i\nu}(X))_{\nu < n_i} \rightarrow X\vec{t}_i$. 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 $Y \cap Z := \text{Cap}_{Y,Z}$, since there are no recursive calls in this particular inductive definition and hence $\cap$ does not occur in

$$\text{Cap}_{Y,Z}^- : \forall_{\vec{x}}(Y\vec{x} \rightarrow Z\vec{x} \rightarrow X\vec{x}) \rightarrow \text{Cap}_{Y,Z} \subseteq X.$$

(iii) The elimination axiom (16) could equivalently be written as

$$I^- : \forall_{\vec{x}}(I\vec{x} \rightarrow (\forall_{\vec{x}_i}((A_{i\nu}(I \cap X))_{\nu < n_i} \rightarrow X\vec{t}_i))_{i < k} \rightarrow X\vec{x})$$

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 $\mathcal{R}_l^\tau$ 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 (16) in the present notes.

EXAMPLES. 1. For Even the introduction axioms are

$$0 \in \text{Even}, \quad \forall_n(n \in \text{Even} \rightarrow S(Sn) \in \text{Even})$$

and the elimination axiom is Even^- :

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

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

$$\begin{aligned} &\forall_{x,y}(x \prec y \rightarrow \text{TC}_\prec(x,y)), \\ &\forall_{x,y,z}(x \prec z \rightarrow \text{TC}_\prec(z,y) \rightarrow \text{TC}_\prec(x,y)) \end{aligned}$$

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

$$\begin{aligned} & \forall_{x,y}(x \prec y \rightarrow Xxy) \rightarrow \forall_{x,y,z}(x \prec z \rightarrow \text{TC}_{\prec}(z, y) \rightarrow Xzy \rightarrow Xxy) \rightarrow \\ & \text{TC}_{\prec} \subseteq X. \end{aligned}$$

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

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

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

$$\begin{aligned} & A \vee B \rightarrow (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C, \\ & A \wedge B \rightarrow (A \rightarrow B \rightarrow C) \rightarrow C, \\ & \exists_x A \rightarrow \forall_x(A \rightarrow B) \rightarrow B \quad (x \notin \text{FV}(B)). \end{aligned}$$

3.2.2. Axioms for coinductive predicates. Consider the algebra $\mathbb{Y}$ of binary trees. We can represent the set of total ideals by an inductively defined predicate $T_{\mathbb{Y}}$. The clauses are

$$\begin{aligned} & (T_{\mathbb{Y}})_0^+ : - \in T_{\mathbb{Y}}, \\ & (T_{\mathbb{Y}})_1^+ : \forall_{t_1, t_2}(t_1, t_2 \in T_{\mathbb{Y}} \rightarrow \text{Ct}_1 t_2 \in T_{\mathbb{Y}}) \end{aligned}$$

and the elimination axiom is

$$T_{\mathbb{Y}}^- : - \in X \rightarrow \forall_{t_1, t_2}(t_1, t_2 \in T_{\mathbb{Y}} \cap X \rightarrow \text{Ct}_1 t_2 \in X) \rightarrow T_{\mathbb{Y}} \subseteq X.$$

We now want to represent the set of *cototal* binary trees (cf. Section 2.2.2) by a coinductively defined predicate ${}^{\text{co}}T_{\mathbb{Y}}$. Note that the conjunction of the two clauses of $T_{\mathbb{Y}}$ is equivalent to

$$\forall_t((t \equiv -) \vee \exists_{t_1, t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \text{Ct}_1 t_2) \rightarrow t \in T_{\mathbb{Y}}).$$

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

$$\forall_t(t \in T_{\mathbb{Y}} \leftrightarrow (t \equiv -) \vee \exists_{t_1, t_2}(t_1, t_2 \in T_{\mathbb{Y}} \wedge t \equiv \text{Ct}_1 t_2))$$

(called inversion property). How can we formally represent cototality of a binary tree? The idea is to define ${}^{\text{co}}T_{\mathbb{Y}}$ 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

$${}^{\text{co}}T_{\mathbb{Y}}^- : \forall_t(t \in {}^{\text{co}}T_{\mathbb{Y}} \rightarrow (t \equiv -) \vee \exists_{t_1, t_2}(t_1, t_2 \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t \equiv \text{Ct}_1 t_2)).$$

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

$$\begin{aligned} {}^{\text{co}}T_Y^+ : \forall t(t \in X \rightarrow (t \equiv -) \vee \exists_{t_1, t_2}(t_1, t_2 \in {}^{\text{co}}T_Y \cup X \wedge t \equiv Ct_1t_2)) \rightarrow \\ X \subseteq {}^{\text{co}}T_Y. \end{aligned}$$

${}^{\text{co}}T_Y^+$ expresses that every competitor X satisfying the closure property is below ${}^{\text{co}}T_Y$.

To understand the general axioms for coinductive predicates note that the conjunction of the k clauses (15) of an inductive predicate I is equivalent to

$$\forall \vec{x}(\bigvee_{i < k} \exists \vec{x}_i(\bigwedge_{\nu < n_i} A_{i\nu}(I) \wedge \vec{x} \equiv \vec{t}_i) \rightarrow I\vec{x}).$$

DEFINITION (Closure and greatest-fixed-point axioms). For an inductive predicate $\mu_X(\forall \vec{x}_i((A_{i\nu}(X))_{\nu < n_i} \rightarrow X\vec{t}_i)_{i < k} =: I$ we define its dual ${}^{\text{co}}I$ (with ν for μ) by the *closure axiom* ${}^{\text{co}}I^-$ and the *greatest-fixed-point axiom* ${}^{\text{co}}I^+$:

$$(17) \quad {}^{\text{co}}I^- : \forall \vec{x}({}^{\text{co}}I\vec{x} \rightarrow \bigvee_{i < k} \exists \vec{x}_i(\bigwedge_{\nu < n_i} A_{i\nu}({}^{\text{co}}I) \wedge \vec{x} \equiv \vec{t}_i)),$$

$$(18) \quad {}^{\text{co}}I^+ : \forall \vec{x}(X\vec{x} \rightarrow \bigvee_{i < k} \exists \vec{x}_i(\bigwedge_{\nu < n_i} A_{i\nu}({}^{\text{co}}I \cup X) \wedge \vec{x} \equiv \vec{t}_i)) \rightarrow X \subseteq {}^{\text{co}}I.$$

(${}^{\text{co}}I \cup X$ was inductively defined above). The axiom expresses that every “competitor” X satisfying the closure axiom is contained in ${}^{\text{co}}I$. We take all substitution instances of ${}^{\text{co}}I^+$, ${}^{\text{co}}I^-$ (w.r.t. substitutions for type and predicate variables) as axioms.

Again we have used a “strengthened” form of the “step formula”, with $A_{i\nu}({}^{\text{co}}I \cup X)$ 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 (18) could be written as

$$\forall \vec{x}(X\vec{x} \rightarrow \forall \vec{x}(X\vec{x} \rightarrow \bigvee_{i < k} \exists \vec{x}_i(\bigwedge_{\nu < n_i} A_{i\nu}({}^{\text{co}}I \cup X) \wedge \vec{x} \equiv \vec{t}_i)) \rightarrow {}^{\text{co}}I\vec{x}).$$

Then its type will be the type of the corecursion operator ${}^{\text{co}}\mathcal{R}_\ell^\tau$ taken as the “computational content” (cf. Section 4.1) of the greatest-fixed-point axiom for ${}^{\text{co}}I$. Therefore in the implementation of TCF this form is used. However, for readability we prefer the form (18) in the present notes.

REMARK. Instead of Leibniz equality $\equiv$ in (17) and (18) we could also use a different equality relation, for instance the n.c. variant $\dot{=}^{\text{nc}}$ of pointwise equality to be introduced in Section 3.3. This leads to a new variant of ${}^{\text{co}}I$.

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

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

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

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

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

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

For n.c. inductive or coinductive predicates the axioms are formed as in the c.r. case, using $\vee^{\text{nc}}$ for the closure axiom of ${}^{\text{co}}I^{\text{nc}}$. But there is an important restriction: for I^{nc} with more than one clause the elimination axiom $(I^{\text{nc}})^-$ 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. $I \subseteq {}^{\text{co}}I$, $I^{\text{nc}} \subseteq {}^{\text{co}}I^{\text{nc}}$.

PROOF. The least-fixed-point axiom (16) for I (i.e., I^-) is equivalent to

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

It suffices that its premise holds with ${}^{\text{co}}I$ for X . This follows from the greatest-fixed-point axiom (18) (i.e., ${}^{\text{co}}I^+$), with the competitor predicate

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

This means that we have to show the premise of (18) with this X , i.e.,

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

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

REMARK. In case of an inductive predicate with non-recursive clauses only also the reverse inclusions ${}^{\text{co}}I \subseteq I$, ${}^{\text{co}}I^{\text{nc}} \subseteq I^{\text{nc}}$. Hence it is not necessary to consider ${}^{\text{co}}I$. Examples are the inductively defined logical connectives $\vee$, $\wedge$, $\exists$ and Leibniz equality.

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

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

For $I \subseteq I^{\text{nc}}$ we use the elimination axiom (16) with I^{nc} as competitor predicate:

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

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

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

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

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

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

LEMMA 3.2.3 (Compatibility of Eq). $\forall_{x,y}(x \equiv y \rightarrow A(x) \rightarrow A(y))$.

PROOF. By the elimination axiom with $X := \{x, y \mid A(x) \rightarrow A(y)\}$. $\square$

Using compatibility of $\equiv$ one easily proves symmetry and transitivity. Define *falsity* by $\mathbf{F} := (\text{ff} \equiv \text{tt})$.

THEOREM 3.2.4 (Ex-falso-quodlibet). *For every formula A we can derive $\mathbf{F} \rightarrow A$ from assumptions $\text{Ef}_Y: \forall_{\vec{x}}(\mathbf{F} \rightarrow Y\vec{x})$ for predicate variables Y strictly positive in A , and $\text{Ef}_I: \forall_{\vec{x}}(\mathbf{F} \rightarrow I\vec{x})$ for inductive predicates I without a nullary clause.*

PROOF. We first show $\text{Ef}_{\text{Eq}}: \mathbf{F} \rightarrow x^\tau \equiv y^\tau$. To see this, we first obtain $\mathcal{R}_{\mathbb{B}}^\tau \text{ff}xy \equiv \mathcal{R}_{\mathbb{B}}^\tau \text{ff}xy$ from the introduction axiom. Then from $\text{ff} \equiv \text{tt}$ we get $\mathcal{R}_{\mathbb{B}}^\tau \text{tt}xy \equiv \mathcal{R}_{\mathbb{B}}^\tau \text{ff}xy$ by compatibility. Now $\mathcal{R}_{\mathbb{B}}^\tau \text{tt}xy$ converts to x and $\mathcal{R}_{\mathbb{B}}^\tau \text{ff}xy$ converts to y . Hence $x^\tau \equiv y^\tau$, 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 ${}^{\text{co}}I\vec{s}$.* Use Lemma 3.2.1. The cases $Y\vec{s}$, $A \rightarrow B$ and $\forall_x A$ are obvious. $\square$

A crucial use of the equality predicate Eq is that it allows us to lift a boolean term $t^{\mathbb{B}}$ to a formula, using $\text{atom}(t^{\mathbb{B}}) := (t^{\mathbb{B}} \equiv \text{tt})$. This opens up a convenient way to deal with equality on closed base types. 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 $(St =_{\mathbb{N}} Ss) \equiv \mathbf{tt}$, 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 $St =_{\mathbb{N}} Ss \rightarrow t =_{\mathbb{N}} s$ explicitly.

3.2.3. Similarity and bisimilarity for base types. We take the algebra $\mathbb{Y}$ of binary trees as an example. In previous examples we have seen that the set of total binary trees can be represented by an inductive predicate $T_{\mathbb{Y}}$ (page 43), and that the set of cototal binary trees can be represented by a coinductive predicate ${}^{\text{co}}T_{\mathbb{Y}}$ (page 43). As candidates for equality we define binary versions of $T_{\mathbb{Y}}$ and ${}^{\text{co}}T_{\mathbb{Y}}$, called similarity $\sim_{\mathbb{Y}}$ and bisimilarity $\approx_{\mathbb{Y}}$. The introduction axioms for $\sim_{\mathbb{Y}}$ are

$$\begin{aligned} (\sim_{\mathbb{Y}})_0^+ &: - \sim_{\mathbb{Y}} -, \\ (\sim_{\mathbb{Y}})_1^+ &: \forall_{t_1, t'_1} (t_1 \sim_{\mathbb{Y}} t'_1 \rightarrow \forall_{t_2, t'_2} (t_2 \sim_{\mathbb{Y}} t'_2 \rightarrow Ct_1 t_2 \sim_{\mathbb{Y}} Ct'_1 t'_2)) \end{aligned}$$

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

$$X(-, -) \rightarrow$$

$$\forall_{t_1, t_2} ((t_1 \sim_{\mathbb{Y}} t_2 \wedge Xt_1 t_2) \rightarrow \forall_{t'_1, t'_2} ((t'_1 \sim_{\mathbb{Y}} t'_2 \wedge Xt'_1 t'_2) \rightarrow X(Ct_1 t_2, Ct'_1 t'_2))) \rightarrow \sim_{\mathbb{Y}} \subseteq X.$$

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

$$\begin{aligned} \forall_{t, t'} (t \approx_{\mathbb{Y}} t' \rightarrow ((t \equiv -) \wedge (t' \equiv -)) \vee \\ \exists_{t_1, t_2, t'_1, t'_2} (t_1 \approx_{\mathbb{Y}} t'_1 \wedge t_2 \approx_{\mathbb{Y}} t'_2 \wedge t \equiv Ct_1 t_2 \wedge t' \equiv Ct'_1 t'_2)) \end{aligned}$$

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

$$\begin{aligned} \forall_{t, t'} (Xtt' \rightarrow ((t \equiv -) \wedge (t' \equiv -)) \vee \\ \exists_{t_1, t_2, t'_1, t'_2} ((t_1 \approx_{\mathbb{Y}} t'_1 \vee Xt_1 t'_1) \wedge (t_2 \approx_{\mathbb{Y}} t'_2 \vee Xt_2 t'_2) \wedge \\ t \equiv Ct_1 t_2 \wedge t' \equiv Ct'_1 t'_2)) \rightarrow \end{aligned}$$

$$X \subseteq \approx_{\mathbb{Y}}.$$

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

3.3. Equality and extensionality

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 $\tau(C)$ (page 37) to *cotype* $\varphi(C)$. Using ideas from Gandy (1953, 1956) and Takeuti (1953) we then define pointwise equality $\dot{=}_{\varphi}$, which is relative to a cotype φ . Extensionality Ext_{φ} relative to φ is defined by $(x \in \text{Ext}_{\varphi}) := (x \dot{=}_{\varphi} x)$.

We now aim at using $\sim_{\mathbb{Y}}$ and $\approx_{\mathbb{Y}}$ for a characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\text{co}}T_{\mathbb{Y}}$. This is useful because it gives us a tool (induction, coinduction) to prove equalities $t \equiv t'$, which otherwise would be difficult. Generally we will need another axiom:

AXIOM (Bisimilarity). *For every closed base type bisimilarity implies Leibniz equality.*

This axiom is justified by the fact that it holds in our intended model. As an example we prove this for $\mathbb{Y}$. Let a range over tokens for $\mathbb{Y}$. By induction on the height $|a^*|$ of extended tokens a^* we prove that for all ideals t, t' and extended tokens a^* from $a^* \in t$ we can infer $a^* \in t'$. It suffices to consider the case $Ca_1^*a_2^*$. From $t \approx_{\mathbb{Y}} t'$ we obtain by the closure property t_1, t_2, t'_1, t'_2 with

$$t_1 \approx t'_1 \wedge t_2 \approx t'_2 \wedge t \equiv Ct_1t_2 \wedge t' \equiv Ct'_1t'_2.$$

Then $a_i^* \in t_i$ (for $i = 1, 2$), and by IH $a_i^* \in t'_i$. Thus $Ca_1^*a_2^* \in t'$.

For simplicity the following lemma and its corollaries are formulated for $\mathbb{Y}$. However, they hold for arbitrary closed base types as well.

LEMMA 3.3.1 (Characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\text{co}}T_{\mathbb{Y}}$).

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

PROOF. (b). “ $\rightarrow$ ”. By the Bisimilarity Axiom it remains to prove $t \approx_{\mathbb{Y}} t' \rightarrow t \in {}^{\text{co}}T_{\mathbb{Y}}$. We use ${}^{\text{co}}T_{\mathbb{Y}}^+$ with competitor $X := \{t \mid \exists_{t'} (t \approx_{\mathbb{Y}} t')\}$. It suffices to prove the premise. Fix t, t' with $t \approx_{\mathbb{Y}} t'$. The goal is

$$(t \equiv -) \vee \exists_{t_1, t_2} ((t_1 \in {}^{\text{co}}T_{\mathbb{Y}} \vee \exists_{t'_1} (t_1 \approx t'_1)) \wedge (t_2 \in {}^{\text{co}}T_{\mathbb{Y}} \vee \exists_{t'_2} (t_2 \approx t'_2)) \wedge t \equiv Ct_1t_2).$$

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

$$(t \equiv - \wedge t' \equiv -) \vee \exists_{t_1, t_2, t'_1, t'_2} (t_1 \approx t'_1 \wedge t_2 \approx t'_2 \wedge t \equiv Ct_1t_2 \wedge t' \equiv Ct'_1t'_2).$$

In the first case we have $t \equiv -$ and are done. In the second case we have t_1, t_2, t'_1, t'_2 with $t_1 \approx t'_1$, $t_2 \approx t'_2$ and $t \equiv Ct_1t_2$, and are done as well.

“ $\leftarrow$ ”. We prove $t \in {}^{\text{co}}T_{\mathbb{Y}} \rightarrow t \equiv t' \rightarrow t \approx_{\mathbb{Y}} t'$ by the greatest-fixed-point property $\approx_{\mathbb{Y}}^+$ with competitor $X := \{t, t' \mid t \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t \equiv t'\}$. It suffices to

prove the premise. Fix t, t' with $t \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t \equiv t'$. The goal is

$$\begin{aligned} (t \equiv - \wedge t' \equiv -) \vee \exists_{t_1, t_2, t'_1, t'_2} ((t_1 \approx t'_1 \vee (t_1 \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t_1 \equiv t'_1)) \wedge \\ (t_2 \approx t'_2 \vee (t_2 \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t_2 \equiv t'_2)) \wedge \\ t \equiv Ct_1t_2 \wedge t' \equiv Ct'_1t'_2)). \end{aligned}$$

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

$$(t \equiv -) \vee \exists_{t_1, t_2} (t_1 \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t_2 \in {}^{\text{co}}T_{\mathbb{Y}} \wedge t \equiv Ct_1t_2).$$

In the first case we have $t \equiv -$ and are done, since $t \equiv t'$. In the second case we have $t_1, t_2 \in {}^{\text{co}}T_{\mathbb{Y}}$ with $t \equiv Ct_1t_2$. Then we are done as well with $t'_1 := t_1$ and $t'_2 := t_2$, since again $t \equiv t'$.

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

COROLLARY 3.3.2.

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

PROOF. Immediate from Lemma 3.3.1. $\square$

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 ${}^{\text{co}}T_{\mathbb{Y}}$.

PROOF. Immediate from Lemma 3.3.1. $\square$

REMARK. For closed base types like $\mathbb{Y}$ we can also relate $\sim_{\mathbb{Y}}$ to the binary boolean-valued function $=_{\mathbb{Y}}: \mathbb{Y} \rightarrow \mathbb{Y} \rightarrow \mathbb{B}$ defined in Section 2.3.1. One easily proves that

$$\begin{aligned} \forall_t (t \in T_{\mathbb{Y}} \rightarrow t = t), \\ \forall_t (t \in T_{\mathbb{Y}} \rightarrow \forall_{t'} (t' \in T_{\mathbb{Y}} \rightarrow t = t' \rightarrow t \sim_{\mathbb{Y}} t')). \end{aligned}$$

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

As example for a base type with parameters consider $\mathbb{L}(\alpha)$. Similarity and bisimilarity for $\mathbb{L}(\alpha)$ requires a binary predicate variable Y of arity (α, α) . As for $\mathbb{Y}$ we can define similarity and bisimilarity for $\mathbb{L}(\alpha)$ as binary versions of totality and cototality. These (co)inductive predicates have Y as parameter. This can be generalized 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 $\vec{\alpha} \rightarrow (\xi)_{i < n} \rightarrow \xi$ be a constructor type. Take $(\mu/\nu)_Z(\vec{K})$, where the clause for the constructor type above is

$$Y_1 u_1 u'_1 \rightarrow \cdots \rightarrow Y_n u_n u'_n \rightarrow Z v_1 v'_1 \rightarrow \cdots \rightarrow Z v_m v'_m \rightarrow Z(C\vec{u}\vec{v}, C\vec{u}'\vec{v}')$$

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

We now introduce cotypes of c.r. formulas, as refinement of the assignment given in Section 3.1 (page 37) of the type $\tau(C)$ to a c.r. predicate or formula C . We mark base types ι as ${}^{\text{co}}\iota$ when they arise from a coinductive predicate $\nu(\vec{K})$ rather than an inductive predicate $\mu(\vec{K})$. This is a *syntactic* procedure only: the model $|\mathcal{C}_\iota|$ with ι determined by the clauses $\vec{K}$ is not changed. It is just that attention switches from total ideals to cototal ones. This switch is necessary when we define totality and pointwise equality of realizers for formulas ${}^{\text{co}}I\vec{t}$. We then must use cototality ${}^{\text{co}}T_\iota$ and bisimilarity $\approx_\iota$ rather than totality T_ι and similarity $\sim_\iota$.

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 .

$$\begin{aligned} \varphi(X^c) &:= \xi, \\ \varphi(\{\vec{x} \mid A\}) &:= \varphi(A), \\ \varphi(I(\vec{\rho}, \vec{P})) &:= \iota_I(\varphi(\vec{P}^c)), \\ \varphi({}^{\text{co}}I(\vec{\rho}, \vec{P})) &:= \begin{cases} \iota_I(\varphi(\vec{P}^c)) & \text{for } \iota_I \text{ non-recursive} \\ {}^{\text{co}}\iota_I(\varphi(\vec{P}^c)) & \text{otherwise,} \end{cases} \\ \varphi(P\vec{t}) &:= \varphi(P), \\ \varphi(A \rightarrow B) &:= \begin{cases} \varphi(A) \rightarrow \varphi(B) & \text{if } A \text{ is c.r.} \\ \varphi(B) & \text{if } A \text{ is n.c.,} \end{cases} \\ \varphi(\forall_x A) &:= \varphi(A) \end{aligned}$$

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

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

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

$$\begin{aligned}\varphi(\sim_{\mathbb{L}}(\sim_{\mathbb{N}})) &= \mathbb{L}(\mathbb{N}), \\ \varphi(\sim_{\mathbb{L}}(\approx_{\mathbb{N}})) &= \mathbb{L}({}^{\text{co}}\mathbb{N}), \\ \varphi(\approx_{\mathbb{L}}(\sim_{\mathbb{N}})) &= {}^{\text{co}}\mathbb{L}(\mathbb{N}), \\ \varphi(\approx_{\mathbb{L}}(\approx_{\mathbb{N}})) &= {}^{\text{co}}\mathbb{L}({}^{\text{co}}\mathbb{N}).\end{aligned}$$

DEFINITION (Totality T_{φ} and pointwise equality $\dot{=}_{\varphi}$ w.r.t. a cotype φ). By recursion on $\text{lev}(\varphi)$ with a subordinate recursion on the height $|\varphi|$ we define *totality* T_{φ} w.r.t. a cotype φ by

$$\begin{aligned}(x \in T_{\alpha}) &:= (x \in X) \quad \text{with } X \text{ uniquely assigned to } \alpha, \\ (x \in T_{\iota(\vec{\varphi})}) &:= (x \in T) \quad \text{with } T := T_{\iota}(\vec{\varphi}), \\ (x \in T_{\text{co}\iota(\vec{\varphi})}) &:= (x \in {}^{\text{co}}T) \quad \text{with } {}^{\text{co}}T := {}^{\text{co}}T_{\iota}(\vec{\varphi}), \\ (f \in T_{\varphi \rightarrow \psi}) &:= \forall_x (x \in T_{\varphi} \rightarrow fx \in T_{\psi})\end{aligned}$$

and *pointwise equality* $\dot{=}_{\varphi}$ w.r.t. a cotype φ by

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

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

REMARK. Relations like $\dot{=}_{\varphi}$ depending on a cotype φ which satisfy the final condition above are called *logical relations*.

DEFINITION. Extensionality w.r.t. a cotype φ is defined by

$$(x \in \text{Ext}_{\varphi}) := (x \dot{=}_{\varphi} x).$$

EXAMPLE (A non-extensional functional). Define f, g of type $\mathbb{N} \rightarrow \mathbb{N}$ by the computation rules $fn = 0$ and $g0 = 0$, $g(Sn) = gn$. Then $f \perp_{\mathbb{N}} = 0$ because of the computation rules for f . For $g \perp_{\mathbb{N}}$ no computation rule fits, but because of the inductive definition of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ in Section 2.4, page 89 $\llbracket g \perp_{\mathbb{N}} \rrbracket$ is the empty ideal $\llbracket \perp_{\mathbb{N}} \rrbracket$. Hence $f \dot{=}_{\mathbb{N}} g$, i.e., $\forall_{n,m} (n \dot{=}_{\mathbb{N}} m \rightarrow fn \dot{=}_{\mathbb{N}} gm)$, since $n \dot{=}_{\mathbb{N}} m$ implies $n \in T_{\mathbb{N}}$ and $n \equiv m$. Therefore the functional F defined by $Fh = h \perp_{\mathbb{N}}$ maps the pointwise equal f, g to different values.

By Lemma 3.3.1 (page 48) we know the equivalence of $\text{Ext}_{\mathbb{Y}}$ and $T_{\mathbb{Y}}$; this also holds for arbitrary closed base cotypes. Since extensionality (i.e., the diagonalization of $\dot{=}$) can be reasonably defined at higher types as well (again

by diagonalization of $\dot{=}$ 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 $\varphi \rightarrow \psi$ be a closed cotype of level 1. The following are equivalent.

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

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

For arbitrary closed cotypes the relation $\dot{=}_\varphi$ is a “partial equivalence relation”, which means the following.

LEMMA 3.3.5. *For every closed cotype φ the relation $\dot{=}_\varphi$ is an equivalence relation on Ext_φ .*

PROOF. By induction on the height $|\varphi|$ of φ . *Case $\iota(\vec{\varphi})/\text{co}\iota(\vec{\varphi})$.* 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 $\varphi \rightarrow \psi$. We first prove symmetry of $\dot{=}_{\varphi \rightarrow \psi}$. Let $f \dot{=}_{\varphi \rightarrow \psi} g$. The goal is $g \dot{=}_{\varphi \rightarrow \psi} f$. Assume $x \dot{=}_\varphi y$. The goal now is $gx \dot{=}_\psi fy$. From $x \dot{=}_\varphi y$ we obtain $y \dot{=}_\varphi x$ by symmetry of $\dot{=}_\varphi$, hence $fy \dot{=}_\psi gx$ from $f \dot{=}_{\varphi \rightarrow \psi} g$, hence $gx \dot{=}_\psi fy$ by symmetry of $\dot{=}_\psi$.

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

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

$$\forall_{\vec{x}, \vec{y}} (\vec{x} \dot{=}_{\vec{\chi}} \vec{y} \rightarrow t(\vec{x}) \dot{=}_\varphi t(\vec{y})).$$

PROOF. This is proved by induction on t . *Case x .* Immediate. *Case c .* By assumption $c \dot{=}_\varphi c$. *Case $\lambda_x t(x, \vec{x})$.* Let $\vec{x} \dot{=}_{\vec{\chi}} \vec{y}$. The goal is

$\lambda_x t(x, \vec{x}) \dot{=}_{\varphi \rightarrow \psi} \lambda_x t(x, \vec{y})$, which by definition means

$$\forall_{x,y} (x \dot{=}_{\varphi} y \rightarrow t(x, \vec{x}) \dot{=}_{\psi} t(y, \vec{y})).$$

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

$$\forall_{x,y} (x \dot{=}_{\varphi} y \rightarrow t(\vec{x})x \dot{=}_{\psi} t(\vec{y})y).$$

Again by IH we have $s(\vec{x}) \dot{=}_{\varphi} s(\vec{y})$. Hence $t(\vec{x})s(\vec{x}) \dot{=}_{\psi} t(\vec{y})s(\vec{y})$. $\square$

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

$$\forall_{\vec{x}} (\vec{x} \in \text{Ext}_{\vec{\chi}} \rightarrow t(\vec{x}) \in \text{Ext}_{\varphi}).$$

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

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 $\text{et}(M)$ 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 $I^{\mathbf{r}} / {}^{\text{co}}I^{\mathbf{r}}$ for (co)inductive predicates $I / {}^{\text{co}}I$. A formula or predicate C is called **r-free** if it does not contain any of these $X^{\mathbf{r}}$, $I^{\mathbf{r}}$ or ${}^{\text{co}}I^{\mathbf{r}}$. A derivation M is called **r-free** if it contains **r-free** formulas only.

DEFINITION ($C^{\mathbf{r}}$ for **r-free** predicates and formulas C). For every **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

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

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 $C_i: \kappa_i(\iota_I)$. 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 ${}^{\text{co}}I^{\mathbf{r}}$ by

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

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

$$\begin{aligned} z \mathbf{r} P\vec{t} &:= P^{\mathbf{r}}\vec{t}z, \\ z \mathbf{r} (A \rightarrow B) &:= \begin{cases} \forall_w(w \mathbf{r} A \rightarrow zw \mathbf{r} B) & \text{if } A \text{ is c.r.} \\ A \rightarrow z \mathbf{r} B & \text{if } A \text{ is n.c.} \end{cases} \\ z \mathbf{r} \forall_x A &:= \forall_x(z \mathbf{r} A). \end{aligned}$$

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 $K_1(X) := \forall_n(n \in X \rightarrow S(Sn) \in X)$. The associated algebra form is $\mu_\xi(\kappa_0(\xi), \kappa_1(\xi))$ with $\kappa_0(\xi) := \xi$ and $\kappa_1(\xi) := \xi \rightarrow \xi$, i.e., the algebra $\mathbb{N}$ with constructors $C_0 := 0$ and $C_1 := S$. 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 $\forall_{n,m}(X^{\mathbf{r}}nm \rightarrow X^{\mathbf{r}}(S(Sn), Sm))$ we obtain

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

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*

$$(19) \quad \text{InvAll}_A: \forall_z(z \mathbf{r} A \rightarrow A).$$

$$(20) \quad \text{InvEx}_A: A \rightarrow \exists_z(z \mathbf{r} A).$$

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

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

- (a) $T_l^{\mathbf{r}}xy$,
- (b) $x \sim_l^{\text{nc}} y$,
- (c) $x \in T_l^{\text{nc}} \wedge x \equiv y$.

PROOF. (a) $\leftrightarrow$ (b). Both $T_l^{\mathbf{r}}xy$ and $x \sim_l^{\text{nc}} y$ satisfy the same clauses. Use the respective elimination axiom in each of the two directions.

(b) $\leftrightarrow$ (c). Use Lemma 3.3.1, page 48. $\square$

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

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

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

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

$$\begin{aligned} & \forall_{n,m} (Xnm \rightarrow (n \equiv 0 \wedge m \equiv 0) \vee \\ & \quad \exists_{n',m'} ((n' \approx_{\mathbb{N}} m' \vee Xn'm') \wedge n \equiv Sn' \wedge m \equiv Sm')) \rightarrow X \subseteq \approx_{\mathbb{N}} \end{aligned}$$

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

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

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

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

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 .

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

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

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

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

$$\begin{aligned} & \forall_{n,m} (Xnm \rightarrow (n \equiv 0 \wedge m \equiv 0) \vee \\ & \quad \exists_{n',m'} (n', m' \in ({}^{\text{co}}T_{\mathbb{N}}^r \cup X) \wedge n \equiv Sn' \wedge m \equiv Sm')) \rightarrow X \subseteq {}^{\text{co}}T_{\mathbb{N}}^r \end{aligned}$$

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

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

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

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

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 .

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.

Recall that for the sum type $\rho + \sigma$ we had the constructors $(\text{InL}_{\rho\sigma})^{\rho \rightarrow \rho + \sigma}$ and $(\text{InR}_{\rho\sigma})^{\sigma \rightarrow \rho + \sigma}$. 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 $\sigma \rightarrow \mathbb{U} + \sigma$. Similarly $\rho + \mathbb{U}$ is viewed as a unary algebra form, with constructors Inl of type $\rho \rightarrow \rho + \mathbb{U}$ and DummyR of type $\rho + \mathbb{U}$.

LEMMA 4.1.3 (Realizers for $\vee$). *$z \mathbf{r} (A \vee B)$ is equivalent to*

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

PROOF. As an example we consider the case A n.c. and B c.r. Recall $\text{OrR}_{X^{\text{nc}}, Y^{\text{c}}} := \mu_Z (X^{\text{nc}} \rightarrow Z, Y^{\text{c}} \rightarrow Z)$. Then

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

Now substituting X^{nc} by A and Y^{r} by $\{y \mid y \mathbf{r} B\}$ in the introduction axioms gives

$$\begin{aligned} (\text{OrR}_{A, \{y \mid y \mathbf{r} B\}}^{\mathbf{r}})_0^+ : A \rightarrow \text{DummyL} \mathbf{r} (A \vee B), \\ (\text{OrR}_{A, \{y \mid y \mathbf{r} B\}}^{\mathbf{r}})_1^+ : \forall_y (y \mathbf{r} B \rightarrow \text{Inr}(y) \mathbf{r} (A \vee B)). \end{aligned}$$

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

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

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

Substitute Z by $\{z \mid (A \wedge z \equiv \text{DummyL}) \vee^{\text{nc}} \exists_y (y \mathbf{r} B \wedge z \equiv \text{Inr}(y))\}$. Then with A for X^{nc} and $\{y \mid y \mathbf{r} B\}$ for Y^{r} the two premises become provable and we obtain

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

Similarly we have

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

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

PROOF. *Case* A, B c.r. Recall $\text{AndD}_{X^c, Y^c} := \mu_{Z^c}(X^c \rightarrow Y^c \rightarrow Z^c)$. Then

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

Now substituting X^r by $\{x \mid x \mathbf{r} A\}$ and Y^r by $\{y \mid y \mathbf{r} B\}$ in the introduction axiom gives

$$(\text{AndD}_{\{x \mid x \mathbf{r} A\}, \{y \mid y \mathbf{r} B\}}^r)_0^+ : \forall_x((x \mathbf{r} A) \rightarrow \forall_y(y \mathbf{r} B \rightarrow \langle x, y \rangle \mathbf{r} (A \wedge B))).$$

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

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

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

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

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

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

Now substituting X^r by $\{z \mid z \mathbf{r} A\}$ and Y^{nc} by B in the introduction axiom gives

$$(\text{AndL}_{\{z \mid z \mathbf{r} A\}, B}^r)_0^+ : \forall_z((z \mathbf{r} A) \rightarrow B \rightarrow z \mathbf{r} (A \wedge B)).$$

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

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

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

$$\forall_z(z \mathbf{r} (A \wedge B) \rightarrow (z \mathbf{r} A) \wedge B). \quad \square$$

LEMMA 4.1.5 (Realizers for $\exists$). $z \mathbf{r} \exists_x A \leftrightarrow \exists_x(z \mathbf{r} A)$ for A c.r.

PROOF. Recall $\text{Ex}_Y := \mu_X(\forall_x(x \in Y \rightarrow X))$. Then

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

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

$$(\text{Ex}_{\{x, z \mid z \mathbf{r} A\}}^r)_0^+ : \forall_{x,z}(z \mathbf{r} A \rightarrow z \mathbf{r} \exists_x A)$$

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

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

which is equivalent to

$$\forall_z(z \in \text{Ex}_{Y^r}^r \rightarrow \forall_z(\exists_x Y^r xz \rightarrow z \in X) \rightarrow z \in X).$$

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 $\forall_z(z \mathbf{r} \exists_x A \rightarrow \exists_x(z \mathbf{r} A))$ from $(\text{Ex}_{\{x, z \mid z \mathbf{r} A\}}^{\mathbf{r}})^{-}$. $\square$

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 ${}^{\text{co}}I^{\mathbf{r}}$. We define its *extracted term* $\text{et}(M)$, 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 $\text{et}(M)$ by

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

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

From the Soundness Theorem 4.2.1 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 $I^{\pm}$, ${}^{\text{co}}I^{\pm}$ realizes the respective axiom¹. Proofs of these facts are automatically generated in Minlog.

¹In Appendix D such proofs for some (co)inductive predicates are written out.

THEOREM 4.2.1 (Soundness). *Let M be an $\mathbf{r}$ -free derivation of a formula A from assumptions $u_i: C_i$ ($i < n$). Then we can derive*

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

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 $u: A$. Subcase A c.r.* Then $\text{et}(u) = z_u$. *Subcase A n.c.* Immediate.

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

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

$$\forall_z (z \mathbf{r} A \rightarrow \text{et}(M)(z) \mathbf{r} B)$$

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

Subcase A n.c. Then the goal is

$$A \rightarrow \text{et}(\lambda_u M) \mathbf{r} B.$$

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

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

Subcase A c.r. By induction hypothesis we have a derivation of B from $z \mathbf{r} A$. Using the invariance axiom $A \rightarrow \exists_z (z \mathbf{r} A)$ we obtain the required derivation of B from A as follows.

$$\frac{\frac{A \rightarrow \exists_z (z \mathbf{r} A) \quad A}{\exists_z (z \mathbf{r} A)} \quad \frac{[z \mathbf{r} A] \quad B}{B} \text{IH}}{B} \exists^-$$

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

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

$$\forall_z (z \mathbf{r} A \rightarrow \text{et}(M)z \mathbf{r} B)$$

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

and hence of

$$A \rightarrow \text{et}(M) \mathbf{r} B$$

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

Case $(M^{A \rightarrow B} N^A)^B$ with B n.c. The goal is to find a derivation of B .

Subcase A c.r. By induction hypothesis we have derivations of $A \rightarrow B$ and of $\text{et}(N) \mathbf{r} A$. Now using the invariance axiom $\forall_z(z \mathbf{r} A \rightarrow A)$ we obtain the required derivation of B by $\rightarrow^-$ from the derivation of $A \rightarrow B$ and

$$\frac{\frac{\forall_z(z \mathbf{r} A \rightarrow A) \quad \text{et}(N)}{\text{et}(N) \mathbf{r} A \rightarrow A} \quad | \text{IH}}{\text{et}(N) \mathbf{r} A} A$$

Subcase A n.c. By induction hypothesis we have derivations of $A \rightarrow B$ 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 $\text{et}(\lambda_x M) \mathbf{r} \forall_x A$. By definition $\text{et}(\lambda_x M) = \text{et}(M)$. Hence we must derive

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

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 $\text{et}(Mt) \mathbf{r} A(t)$. By definition $\text{et}(Mt) = \text{et}(M)$, and by induction hypothesis we can derive

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

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 $\forall^-$. $\square$

4.3. Extensionality of extracted terms

Let I be an inductive predicate and ι_I its associated algebra. One can show² that

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

Since the term $\text{et}(M)$ 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 $\text{et}(M)$ is extensional w.r.t. the cotype of A .

²In Appendix E the proofs for some (co)inductive predicates are written out.

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 $d_0, d_1, d_2 \dots$ represents the real number

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

We are interested in formally verified algorithms on real numbers in an arbitrary format. To this end we generate such algorithms as extracted terms $\text{et}(M)$ 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 ${}^{\text{co}}I$ coinductively defined in such a way that the computational content of $x \in {}^{\text{co}}I$ 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 $M: \mathbb{Z}^+ \rightarrow \mathbb{N}$ such that $(a_n)_n$ is a *Cauchy sequence* with modulus M , that is

$$|a_n - a_m| \leq \frac{1}{2^p} \quad \text{for } n, m \geq M(p)$$

and M is weakly increasing (that is $M(p) \leq M(q)$ for $p \leq q$). 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 Archimedean 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 $\max\{|a_n| \mid n \leq n_0\} + \frac{1}{2} \leq 2^{p_x}$. Then $|a_n| \leq 2^{p_x}$ for all n . $\square$

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

$$-\frac{1}{2^p} \leq a_{M(p)} \quad \text{for all } p \in \mathbb{Z}^+.$$

It is *p-positive* (written $x \in_p \mathbb{R}^+$, or $x \in \mathbb{R}^+$ if p is not needed) if

$$\frac{1}{2^p} \leq a_{M(p+1)}.$$

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) $\forall_p \exists_{n_0} \forall_{n \geq n_0} (-\frac{1}{2^p} \leq a_n)$.

PROOF. (a) implies (b). For $n \geq M(p+1)$ we have

$$\begin{aligned} -\frac{1}{2^p} &\leq -\frac{1}{2^{p+1}} + a_{M(p+1)} \\ &= -\frac{1}{2^{p+1}} + (a_{M(p+1)} - a_n) + a_n \\ &\leq -\frac{1}{2^{p+1}} + \frac{1}{2^{p+1}} + a_n. \end{aligned}$$

(b) implies (a). Let $q \in \mathbb{Z}^+$ and $n \geq n_0, M(p)$ with n_0 provided by (b) (for q). Then

$$\begin{aligned} -\frac{1}{2^p} - \frac{1}{2^q} &\leq -\frac{1}{2^p} + a_n \\ &= -\frac{1}{2^p} + (a_n - a_{M(p)}) + a_{M(p)} \\ &\leq -\frac{1}{2^p} + \frac{1}{2^p} + a_{M(p)}. \end{aligned}$$

The claim follows, because this holds for every q . $\square$

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. $\square$

LEMMA 5.1.9 (RealPosChar). *For a real $x := ((a_n)_n, M)$ with $x \in_p \mathbb{R}^+$ we have*

$$\frac{1}{2^{p+1}} \leq a_n \quad \text{for } M(p+1) \leq n.$$

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 $x \in_{q+1} \mathbb{R}^+$. $\square$

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 $x = y$. $\square$

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	$L(p)$
$x + y$	$a_n + b_n$	$\max(M(p+1), N(p+1))$
$-x$	$-a_n$	$M(p)$
$ x $	$ a_n $	$M(p)$
$x \cdot y$	$a_n \cdot b_n$	$\max(M(p+1+p_y), N(p+1+p_x))$
$\frac{1}{x}$ for $ x \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 $n, m \geq \max(M(p+1+p_y), N(p+1+p_x))$.

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

$$\begin{aligned}
 \left| \frac{1}{a_n} - \frac{1}{a_m} \right| &= \frac{|a_m - a_n|}{|a_n a_m|} \\
 &\leq 2^{2(q+1)} |a_m - a_n| \quad \text{for } n, m \geq M(q+1) \\
 &\leq \frac{1}{2^p} \quad \text{for } n, m \geq M(2(q+1)+p).
 \end{aligned}$$

The claim now follows from the assumption that M is weakly increasing. $\square$

LEMMA 5.1.12. *For reals x, y, z*

$$\begin{aligned}
 x + (y + z) &= (x + y) + z & x \cdot (y \cdot z) &= (x \cdot y) \cdot z \\
 x + 0 &= x & x \cdot 1 &= x \\
 x + (-x) &= 0 & 0 < |x| \rightarrow x \cdot \frac{1}{x} &= 1 \\
 x + y &= y + x & x \cdot y &= y \cdot x \\
 x \cdot (y + z) &= x \cdot y + x \cdot z
 \end{aligned}$$

PROOF. For $0 < |x| \rightarrow x \cdot \frac{1}{x} = 1$ 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. $\square$

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 $x = y \rightarrow x + z = y + z$. But this follows immediately from Lemma 5.1.3 (RealEqChar): the n_0 for the conclusion can be the same as for the premise. $\square$

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 $\frac{1}{2^{p+1}} \leq |a_n|$. $\square$

LEMMA 5.1.15. *For reals x, y ,*

- (a) $x, y \in \mathbb{R}^{0+} \rightarrow x + y, x \cdot y \in \mathbb{R}^{0+}$,
- (b) $x, y \in \mathbb{R}^+ \rightarrow x + y, x \cdot y \in \mathbb{R}^+$,
- (c) $x \in \mathbb{R}^{0+} \rightarrow -x \in \mathbb{R}^{0+} \rightarrow x = 0$.

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 $|a_n| \leq \frac{1}{2^p}$. $\square$

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 ,*

$$\begin{array}{ll}
x \leq x & x \not\leq x \\
x \leq y \rightarrow y \leq x \rightarrow x = y & x < y \rightarrow y < z \rightarrow x < z \\
x \leq y \rightarrow y \leq z \rightarrow x \leq z & x < y \rightarrow x + z < y + z \\
x \leq y \rightarrow x + z \leq y + z & x < y \rightarrow 0 < z \rightarrow x \cdot z < y \cdot z \\
x \leq y \rightarrow 0 \leq z \rightarrow x \cdot z \leq y \cdot z &
\end{array}$$

PROOF. From Section 5.1.4. $\square$

Here we have left out information on witnesses p for the statements proving a $<$ -formula. Such estimates can easily be given explicitly. Here are two examples.

LEMMA 5.1.18 (RealPosPlus). $0 \leq x \rightarrow 0 <_p y \rightarrow 0 <_{p+3} x + y$.

PROOF. From $0 \leq x$ we have $\forall q \exists n_0 \forall n \geq n_0 (-\frac{1}{2^q} \leq a_n)$. From $0 <_p y$ we have some n_1 such that $\forall n \geq n_1 (\frac{1}{2^{p+1}} \leq b_n)$. Pick n_0 for $p+2$. Then $n_0, n_1 \leq n$ implies $0 \leq a_n + \frac{1}{2^{p+2}}$ and $\frac{1}{2^{p+2}} \leq b_n - \frac{1}{2^{p+2}}$, hence $\frac{1}{2^{p+2}} \leq a_n + b_n$. Now Lemma 5.1.9 (RealPosChar) gives $0 <_{p+3} x + y$. $\square$

LEMMA 5.1.19. $x \leq y \rightarrow y <_p z \rightarrow x <_{p+5} z$.

PROOF. This follows from Lemma 5.1.18 (RealPosPlus). $\square$

As is to be expected in view of the existential and universal character of the predicates $<$ and $\leq$ on the reals, we have:

LEMMA 5.1.20 (LeIsNotGt). $x \leq y \leftrightarrow y \not< x$.

PROOF. $\rightarrow$. Assume $x \leq y$ and $y < x$. By Lemma 5.1.19 we obtain $x < x$, a contradiction.

$\leftarrow$. It clearly suffices to show $0 \not< z \rightarrow z \leq 0$, for a real z given by $(c_n)_n$. Assume $0 \not< z$. We must show $\forall p \exists n_0 \forall n \geq n_0 (c_n \leq \frac{1}{2^p})$. Let p be given. By assumption $0 \not< z$, hence $\neg \exists q (\frac{1}{2^q} \leq c_{M(q+1)})$. For $q := p+1$ this implies $c_{M(p+2)} < \frac{1}{2^{p+1}}$, hence $c_n \leq c_{M(p+2)} + \frac{1}{2^{p+2}} < \frac{1}{2^p}$ for $M(p+2) \leq n$. $\square$

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 $z \leq y$ or $x \leq z$.*

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) $\frac{1}{2^p} \leq b_n - a_n$ 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 $l \geq m$:

$$a_l \leq a_n + \frac{1}{2^{p+2}} \leq a_n + \frac{b_n - a_n}{4} = \frac{a_n + b_n}{2} - \frac{b_n - a_n}{4} \leq c_m - \frac{1}{2^{p+2}} \leq c_l. \quad \square$$

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 $x, y \leq z \rightarrow \max(x, y) \leq z$. 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 $m := \max(M(p+1), N(p+1))$.

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 ${}^{\text{co}}I$. We model infinite sequences of signed digits (streams) as objects in the algebra $\mathbb{S}(\text{sd}) := \mu_{\xi}(\text{C}: \text{sd} \rightarrow \xi \rightarrow \xi)$, where $\text{sd} := \mu_{\xi}(\text{SdR}: \xi, \text{SdM}: \xi, \text{SdL}: \xi)$ is a 3-element algebra. Such streams will appear as realizers of an inductive predicate I defined by the single clause

$$(21) \quad \forall_{d, x', x} \left(d \in \text{Sd} \rightarrow x' \in I \rightarrow x = \frac{x' + d}{2} \rightarrow x \in I \right).$$

Here (and later) x, y, z range over real numbers in $[-1, 1]$ and d over integers. Sd is a (formally inductive) predicate expressing that its integer argument d is a signed digit, i.e., $|d| \leq 1$. We have chosen (21) rather than the simpler

$$(22) \quad \forall_{d,x} \left(d \in \text{Sd} \rightarrow x \in I \rightarrow \frac{x+d}{2} \in I \right),$$

since we want I to be compatible with the defined equality $=$ on real numbers

$$(23) \quad \forall_{x,y} (x = y \rightarrow x \in I \rightarrow y \in I),$$

which easily follows from (21) (with reflexivity, symmetry and transitivity of $=$). Using (23) we then obtain (22) from (21) as a lemma.

The dual ${}^{\text{co}}I$ of I is defined by its closure axiom

$$x \in {}^{\text{co}}I \rightarrow \exists_{d,x',y} \left(d \in \text{Sd} \wedge x' \in {}^{\text{co}}I \wedge y = \frac{x'+d}{2} \wedge x = y \right).$$

Similar to what was done above it can be simplified to

$$(24) \quad x \in {}^{\text{co}}I \rightarrow \exists_{d,x'} \left(d \in \text{Sd} \wedge x' \in {}^{\text{co}}I \wedge x = \frac{x'+d}{2} \right).$$

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 69, 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. $|x| \leq 1 \rightarrow x \in {}^{\text{co}}I$.

PROOF. We use coinduction with competitor predicate $\{x \mid |x| \leq 1\}$. It suffices to prove

$$|x| \leq 1 \rightarrow \exists_{d,x'} \left(d \in \text{Sd} \wedge (x' \in {}^{\text{co}}I \vee |x'| \leq 1) \wedge x = \frac{x'+d}{2} \right).$$

Compare x with $[-\frac{1}{2}, 0]$. If $x \leq 0$, return SdL and continue with $2x + 1$. If $-\frac{1}{2} \leq x$, compare x with $[0, \frac{1}{2}]$. If $x \leq \frac{1}{2}$, return SdM and continue with $2x$. If $0 \leq x$, return SdR and continue with $2x - 1$. $\square$

More precisely, we have proven (with f of type $\mathbb{N} \rightarrow \mathbb{Q}$)

$$f \in T \rightarrow M \in T \rightarrow x \equiv (f, M) \rightarrow x \in \text{Real} \rightarrow |x| \leq 1 \rightarrow x \in {}^{\text{co}}I$$

of cotype $(\mathbb{N} \rightarrow \mathbb{Q}) \rightarrow (\mathbb{P} \rightarrow \mathbb{N}) \rightarrow {}^{\text{co}}\mathcal{S}(\text{sd})$. Here $x \in \text{Real}$ is an n.c. formula expressing that x is a real number in format (i). The term extracted from

this proof has type $(\mathbb{N} \rightarrow \mathbb{Q}) \rightarrow (\mathbb{P} \rightarrow \mathbb{N}) \rightarrow \mathbb{S}(\text{sd})$ and is built by the corecursion operator with step function

$$x \mapsto \begin{cases} (\text{SdL}, 2x + 1) & \text{if } g(-\frac{1}{2}, 0, x) = \mathbf{tt} \\ (\text{SdM}, 2x) & \text{if } g(-\frac{1}{2}, 0, x) = \mathbf{ff} \text{ and } g(0, \frac{1}{2}, x) = \mathbf{tt} \\ (\text{SdR}, 2x - 1) & \text{if } g(0, \frac{1}{2}, x) = \mathbf{ff}. \end{cases}$$

Here g is a function of type $\mathbb{Q} \rightarrow \mathbb{Q} \rightarrow \mathbb{R} \rightarrow \mathbb{B}$ 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:

$$x, y \in {}^{\text{co}}I \rightarrow \frac{x + y}{2} \in {}^{\text{co}}I.$$

Following Berger and Seisenberger (2010) we begin with an informal proof. The computational content of this proof will be the desired algorithm.

Consider two sets of averages, the second one with a “carry” $i \in \mathbb{Z}$

$$P := \left\{ \frac{x + y}{2} \mid x, y \in {}^{\text{co}}I \right\}, \quad Q := \left\{ \frac{x + y + i}{4} \mid x, y \in {}^{\text{co}}I, i \in \text{Sd}_2 \right\},$$

where Sd_2 is a (formally inductive) predicate expressing that the integer i is an extended signed digit, i.e., $|i| \leq 2$. It suffices to show that Q satisfies

$$x \in Q \rightarrow \exists_{d, x'} (d \in \text{Sd} \wedge x' \in Q \wedge x = \frac{x' + d}{2}),$$

for then by the greatest-fixed-point axiom for ${}^{\text{co}}I$ we have $Q \subseteq {}^{\text{co}}I$. Since we also have $P \subseteq Q$ we then obtain $P \subseteq {}^{\text{co}}I$, which is our claim.

Below we will need functions $J, K: \mathbb{Z} \rightarrow \mathbb{Z}$ such that

$$(25) \quad \forall_i (i = J(i) + 4K(i)) \quad (\text{with } \forall_i (|J(i)| \leq 2), \forall_i (|i| \leq 6 \rightarrow |K(i)| \leq 1)).$$

We use the following notation for streams u of type $\mathbb{S}(\text{sd})$:

$$(u)_n := n\text{-th entry in } u,$$

$$Tu := \text{rest of } u \text{ after removal of its first entry.}$$

Clearly $(T^n u)_m = (u)_{n+m}$, where T^n denotes the n -th iteration of T .

LEMMA 5.2.2 (CoIAvToAvc).

$$x, y \in {}^{\text{co}}I \rightarrow \exists_{i, x', y'} (i \in \text{Sd}_2 \wedge x', y' \in {}^{\text{co}}I \wedge \frac{x + y}{2} = \frac{x' + y' + i}{4}).$$

PROOF. By the closure axiom (24) for ${}^{\text{co}}I$ we can write $x = \frac{x' + d}{2}$ and $y = \frac{y' + e}{2}$ for some $d, e \in \text{Sd}$ and $x', y' \in {}^{\text{co}}I$. Then

$$\frac{x + y}{2} = \frac{x' + y' + d + e}{4}. \quad \square$$

The computational content of this proof is a function

$$\begin{aligned} \text{Av}_{\text{init}} &: \mathbb{S}(\text{sd}) \rightarrow \mathbb{S}(\text{sd}) \rightarrow \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd}) \\ \text{Av}_{\text{init}}(u, v) &:= \langle (u)_0 + (v)_0, Tu, Tv \rangle. \end{aligned}$$

LEMMA 5.2.3 (CoIAvcSatCoICl).

$$i \in \text{Sd}_2 \wedge x, y \in {}^{\text{co}}I \rightarrow \exists_{d \in \text{Sd}} \exists_{j \in \text{Sd}_2} \exists_{x', y' \in {}^{\text{co}}I} \left(\frac{x+y+i}{4} = \frac{\frac{x'+y'+j}{4} + d}{2} \right).$$

PROOF. By the closure axiom (24) for ${}^{\text{co}}I$ we can write $x = \frac{x'+d}{2}$ and $y = \frac{y'+e}{2}$ for some $d, e \in \text{Sd}$ and $x', y' \in {}^{\text{co}}I$. Then

$$\frac{x+y+i}{4} = \frac{x'+y'+d+e+2i}{8}.$$

Since $|d+e+2i| \leq 6$ we can write $d+e+2i = j+4k$ with $|j| \leq 2$ and $|k| \leq 1$ by the JK-property (25). Therefore

$$\frac{x+y+i}{4} = \frac{x'+y'+j+4k}{8} = \frac{\frac{x'+y'+j}{4} + k}{2}. \quad \square$$

The computational content of this proof is a function

$$\begin{aligned} \text{Av}_{\text{step}} &: \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd}) \rightarrow \text{sd} \times \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd}) \\ \text{Av}_{\text{step}} \langle i, u, v \rangle &:= \langle K((u)_0 + (v)_0 + 2i), J((u)_0 + (v)_0 + 2i), Tu, Tv \rangle. \end{aligned}$$

By coinduction from Lemma 5.2.3 we obtain

$$\text{LEMMA 5.2.4. } \forall_z (\exists_{i,x,y} (i \in \text{Sd}_2 \wedge x, y \in {}^{\text{co}}I \wedge z = \frac{x+y+i}{4}) \rightarrow z \in {}^{\text{co}}I).$$

THEOREM 5.2.5. *The average of two real numbers x, y in ${}^{\text{co}}I$ is in ${}^{\text{co}}I$:*

$$x, y \in {}^{\text{co}}I \rightarrow \frac{x+y}{2} \in {}^{\text{co}}I.$$

PROOF. Immediate from Lemma 5.2.2 and Lemma 5.2.4. $\square$

The extracted term is short (19 lines in `minlog/examples/analysis/`, file `sdavaux.scm`), and involves the corecursion operator. It represents the function

$$\text{cCoIAverage}: \mathbb{S}(\text{sd}) \rightarrow \mathbb{S}(\text{sd}) \rightarrow \mathbb{S}(\text{sd})$$

defined using corecursion, as follows.

- (i) From $u, v \in \mathbb{S}(\text{sd})$ form an initial triple $\text{Av}_{\text{init}}(u, v) \in \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd})$.
- (ii) Iterate $\text{Av}_{\text{step}}: \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd}) \rightarrow \text{sd} \times \text{sd}_2 \times \mathbb{S}(\text{sd}) \times \mathbb{S}(\text{sd})$ starting with $\text{Av}_{\text{init}}(u, v)$.
- (iii) Return the stream of the generated $d \in \text{sd}$.

5.3. Lookahead

Recall that real numbers in $[-1.1]$ can be represented in the form

$$\sum_{n=0}^{\infty} \frac{d_n}{2^{n+1}} \quad (d_n \in \{-1, 0, 1\}).$$

We are interested in algorithms for arithmetical functions on real numbers operating on such stream representations $(d_n)_n$. In particular we want to verify lookahead properties of such algorithms.

To this end we inductively define a binary predicate L of arity $(\mathbb{R}, \mathbb{N})$. The intended property of L is

a realizer of $x \in L_n$ is a list $(d_m)_{m < n}$ of signed digits which approximates x with error bound $\frac{1}{2^n}$, i.e., such that

$$\left| \sum_{m=0}^{n-1} \frac{d_m}{2^{m+1}} - x \right| \leq \frac{1}{2^n}.$$

Here we have written $x \in L_n$ for $L(x, n)$. Let x, y, z range 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., $|d| \leq 1$. L is defined by the two clauses

$$L_0^+ : \forall x (|x| \leq 1 \rightarrow x \in L_0),$$

$$L_1^+ : \forall_{x,y,n,d} \left(d \in \text{Sd} \rightarrow x \in L_n \rightarrow y = \frac{x+d}{2} \rightarrow y \in L_{n+1} \right).$$

L_1^+ says that if we know the first n digits of a representation of x and that $y = \frac{x+d}{2}$, then we know the first $n+1$ digits of a representation of y .

The induction or least-fixed-point (lfp) axiom for L is

$$\begin{aligned} & \forall_x (|x| \leq 1 \rightarrow Y(x, 0)) \rightarrow \\ & \forall_{x,y,n,d} \left(d \in \text{Sd} \rightarrow x \in L_n \rightarrow Y(x, n) \rightarrow y = \frac{x+d}{2} \rightarrow Y(y, n+1) \right) \rightarrow L \subseteq Y. \end{aligned}$$

The algebra associated with L is $\mathbb{L} := \mu_{\xi}(U : \xi, C : \text{sd} \rightarrow \xi \rightarrow \xi)$, i.e., finite lists of signed digits $-1, 0, 1$. We write u, v for objects of type $\mathbb{L}$, and $d :: u$ for Cdu .

We will need some properties of L .

LEMMA 5.3.1 (**LCompat**). $x = y \rightarrow x \in L_n \rightarrow y \in L_n$.

PROOF. Use the lfp-axiom for L . □

The term extracted from this proof is the identity on $\mathbb{L}$; it will be omitted in the sequel.

LEMMA 5.3.2 (**LToBd**). $x \in L_n \rightarrow |x| \leq 1$.

PROOF. Use induction on n . $\square$

Parallel to the (simplified) closure axiom (24) for ${}^{\text{co}}I$ (page 71) we have

LEMMA 5.3.3 (LClosure). $x \in L_{n+1} \rightarrow \exists_{d,x'} (d \in \text{Sd} \wedge x' \in L_n \wedge x = \frac{x'+d}{2})$.

PROOF. Use the lfp-axiom for L . $\square$

The extracted term **cLClosure** has type $\mathbb{L} \rightarrow \text{sd} \times \mathbb{L}$. It satisfies

$$\text{cLClosure}(d :: u) = \langle d, u \rangle.$$

We now proceed as before, but with L instead of ${}^{\text{co}}I$. As in Section 5.2.3 (page 72) we show

$$\left\{ \frac{x+y}{2} \mid x, y \in L_{n+1} \right\} \subseteq \left\{ \frac{x+y+i}{4} \mid x, y \in L_n, i \in \text{Sd}_2 \right\} \subseteq L_n,$$

where this time the final conclusion is proved by induction.

LEMMA 5.3.4 (LAvToAvc).

$$x, y \in L_{n+1} \rightarrow \exists_{i,x',y'} \left(i \in \text{Sd}_2 \wedge x', y' \in L_n \wedge \frac{x+y}{2} = \frac{x'+y'+i}{4} \right).$$

PROOF. By Lemma 5.3.3 we can write $x = \frac{x'+d}{2}$ and $y = \frac{y'+e}{2}$ for some $d, e \in \text{Sd}$ and $x', y' \in L_n$. Then

$$\frac{x+y}{2} = \frac{x'+y'+d+e}{4}. \quad \square$$

The extracted term **cLAvToAvc** has type $\mathbb{L} \rightarrow \mathbb{L} \rightarrow \text{sd}_2 \times \mathbb{L} \times \mathbb{L}$ and satisfies

$$\text{cLAvToAvc}(d :: u, e :: v) = \langle d+e, u, v \rangle.$$

LEMMA 5.3.5 (LAvcSatLC1).

$$i \in \text{Sd}_2 \wedge x, y \in L_{n+1} \rightarrow \exists_{d \in \text{Sd}} \exists_{j \in \text{Sd}_2} \exists_{x', y' \in L_n} \left(\frac{x+y+i}{4} = \frac{\frac{x'+y'+j}{4} + d}{2} \right).$$

PROOF. Let $i \in \text{Sd}_2$ and $x, y \in L_{n+1}$. By Lemma 5.3.3 we can write $x = \frac{x'+d}{2}$ and $y = \frac{y'+e}{2}$ for some $d, e \in \text{Sd}$ and $x', y' \in L_n$. Then

$$\frac{x+y+i}{4} = \frac{x'+y'+d+e+2i}{8}.$$

Since $|d+e+2i| \leq 6$ we can write $d+e+2i = j+4k$ with $|j| \leq 2$ and $|k| \leq 1$ by the JK-property (25). Therefore

$$\frac{x+y+i}{4} = \frac{x'+y'+j+4k}{8} = \frac{\frac{x'+y'+j}{4} + k}{2}. \quad \square$$

The extracted term **cLAvcSatLC1** has type $\text{sd}_2 \times \mathbb{L} \times \mathbb{L} \rightarrow \text{sd} \times \text{sd}_2 \times \mathbb{L} \times \mathbb{L}$ and satisfies

$$\text{cLAvcSatLC1}\langle i, d :: u, e :: v \rangle = \langle K(d + e + 2i), J(d + e + 2i), u, v \rangle.$$

By induction from Lemma 5.3.5 we obtain

LEMMA 5.3.6 (LAvcToL). $n \in T_{\mathbb{N}} \rightarrow i \in \text{Sd}_2 \wedge x, y \in L_n \rightarrow \frac{x+y+i}{4} \in L_n$.

PROOF. By induction on n . *Base.* Assume $i \in \text{Sd}_2$ and $x, y \in L_0$. Then $|x|, |y| \leq 1$ by Lemma 5.3.2, hence $\frac{x+y+i}{4} \in L_0$ by the first clause of L . *Step.* Assume $i \in \text{Sd}_2$ and $x, y \in L_{n+1}$. By Lemma 5.3.5 we can write

$$\frac{x + y + i}{4} = \frac{\frac{x' + y' + j}{4} + d}{2}$$

with $x', y' \in L_n, j \in \text{Sd}_2$ and $d \in \text{Sd}$. Now we can apply the second clause of L , using the IH to obtain $x', y' \in L_n$ and then compatibility of L . $\square$

The extracted term **cLAvcToL** has type $\mathbb{N} \rightarrow \text{sd}_2 \times \mathbb{L} \times \mathbb{L} \rightarrow \mathbb{L}$ and is defined by recursion over $\mathbb{N}$. It satisfies

$$\begin{aligned} \text{cLAvcToL}(0, \langle i, u, v \rangle) &= U, \\ \text{cLAvcToL}(n + 1, \langle i, u, v \rangle) &= d :: \text{cLAvcToL}(n, w) \end{aligned}$$

where

$$\langle d, w \rangle = \text{cLAvcSatLC1}\langle i, u, v \rangle.$$

THEOREM 5.3.7 (LAverage). $n \in T_{\mathbb{N}} \rightarrow x, y \in L_{n+1} \rightarrow \frac{x+y}{2} \in L_n$.

PROOF. Immediate by Lemma 5.3.4 and Lemma 5.3.6. $\square$

The extracted term **cLAverage** has type $\mathbb{N} \rightarrow \mathbb{L} \rightarrow \mathbb{L} \rightarrow \mathbb{L}$. It satisfies

$$\text{cLAverage}(n, u, v) = \text{cLAvcToL}(n, \text{cLAvToAvc}(u, v)).$$

The way **cLAverage** operates can be understood as follows.

- From $u, v \in \mathbb{L}$ form the initial triple $\text{cLAvToAvc}(u, v) \in \text{sd}_2 \times \mathbb{L} \times \mathbb{L}$.
- Recall that **cLAvcSatLC1** has type $\text{sd}_2 \times \mathbb{L} \times \mathbb{L} \rightarrow \text{sd} \times \text{sd}_2 \times \mathbb{L} \times \mathbb{L}$. Iterate it n times, starting with $\text{cLAvToAvc}(u, v)$.
- Return the list of generated $d \in \text{sd}$.

The soundness proof is within TCF and can be automatically generated with **add-sound**; see **sdindav.scm** (on the webpage of the course).

With the same method one can also treat other functions in exact real arithmetic. For instance, recently Nils Köpp proved

$$\begin{aligned} x \in L_{n+5} &\rightarrow x^2 \in L_n, \\ x, y \in L_{n+6} &\rightarrow xy \in L_n. \end{aligned}$$

APPENDIX A

Atomic coherent information systems

The objects studied in mathematics have types, which in many cases are function types, possibly of a higher type. We are particularly interested in computable such objects. Their domains are partial continuous functionals (Scott, Ersov) which we study first, via Scott's 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 simplification (based on Schwichtenberg (2006)) of the ones of Scott (1982) and Larsen and Winskel (1991).

DEFINITION. An *atomic coherent information system* (abbreviated *acis*) is a triple $(A, \text{Con}, \geq)$ with A a countable set (the *tokens*, denoted $a, b, \dots$), Con a nonempty set of finite subsets of A (the *consistent* sets or *formal neighborhoods*, denoted $U, V, \dots$), and $\geq$ a transitive and reflexive relation on A (the *entailment relation*) which satisfy

- (a) $\emptyset \in \text{Con}$, and $\{a\} \in \text{Con}$ for every $a \in A$,
- (b) $U \in \text{Con}$ if and only if every two-element subset of U is in Con , and
- (c) if $\{a, b\} \in \text{Con}$ and $b \geq c$, then $\{a, c\} \in \text{Con}$.

We write $U \geq a$ for $\exists b \in U b \geq a$, and $U \geq V$ for $\forall a \in V U \geq a$.

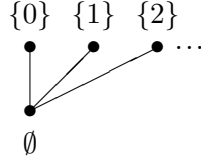
DEFINITION. The *ideals* (also called *objects*) of an acis $\mathbf{A} = (A, \text{Con}, \geq)$ are defined to be those subsets x of A which satisfy

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

We write $x \in |\mathbf{A}|$ to mean that x is an ideal of $\mathbf{A}$.

EXAMPLES. The *deductive closure* $\overline{U} := \{a \in A \mid U \geq a\}$ of $U \in \text{Con}$ is an ideal.

Every countable set A can be turned into a “flat” acis by letting the set of tokens be A , $\text{Con} := \{\emptyset\} \cup \{\{a\} \mid a \in A\}$ and $U \geq a$ 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

$$\begin{aligned} \text{Con} &:= \{ \{ (a_i, b_i) \mid i < k \} \mid \forall i, j < k (a_i = a_j \rightarrow b_i = b_j) \}, \\ U \geq (a, b) &:= (a, b) \in U. \end{aligned}$$

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

REMARK. One can show that for an arbitrary acis $\mathbf{A} = (A, \text{Con}, \geq)$ the structure $(|\mathbf{A}|, \subseteq, \emptyset)$ is a “domain” (also called Scott-Ershov domain, or “bounded complete algebraic cpo”), whose set of “compact elements” can be represented as $|\mathbf{A}|_c = \{\overline{U} \mid U \in \text{Con}\}$. The converse holds as well: every countable domain can be represented as an acis. We will not need this relation to standard (non-constructive) domain theory (cf. Abramsky and Jung (1994)), and hence not even define these notions here.

We define the “function space” $\mathbf{A} \rightarrow \mathbf{B}$ between two acis’s $\mathbf{A}$ and $\mathbf{B}$.

LEMMA. Let $\mathbf{A} = (A, \text{Con}, \geq)$ be an acis. $U \geq V_1, V_2$ implies $V_1 \cup V_2 \in \text{Con}$.

PROOF. Let $b_1 \in V_1, b_2 \in V_2$. Then we have $a_1, a_2 \in U$ such that $a_i \geq b_i$. From $\{a_1, a_2\} \in \text{Con}$ we obtain $\{a_1, b_2\} \in \text{Con}$ by (c), hence $\{b_1, b_2\} \in \text{Con}$ again by (c). $\square$

DEFINITION. Let $\mathbf{A} = (A, \text{Con}_A, \geq_A)$ and $\mathbf{B} = (B, \text{Con}_B, \geq_B)$ be acis’s. Define $\mathbf{A} \rightarrow \mathbf{B} = (C, \text{Con}, \geq)$ by

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

$$\{(U_1, b_1), \dots, (U_n, b_n)\} \in \text{Con} :\Leftrightarrow \forall i, j (U_i \cup U_j \in \text{Con}_A \rightarrow \{b_i, b_j\} \in \text{Con}_B),$$

$$(U, b) \geq (V, c) :\Leftrightarrow V \geq_A U \wedge b \geq_B c.$$

LEMMA. Let $\mathbf{A} = (A, \text{Con}_A, \geq_A)$ and $\mathbf{B} = (B, \text{Con}_B, \geq_B)$ be acis's. Then $\mathbf{A} \rightarrow \mathbf{B}$ is an acis again.

PROOF. Clearly $\geq$ is transitive and reflexive, and the conditions (a) and (b) of an acis hold; it remains to check (c). Let $\{(U_1, b_1), (U_2, b_2)\} \in \text{Con}$ and $(U_2, b_2) \geq (V, c)$, hence $V \geq U_2$ and $b_2 \geq c$. We must show $\{(U_1, b_1), (V, c)\} \in \text{Con}$. Assume $U_1 \cup V \in \text{Con}$; we must show $\{b_1, c\} \in \text{Con}$. Now $U_1 \cup V \in \text{Con}$ and $V \geq U_2$ by the previous lemma imply $U_1 \cup U_2 \in \text{Con}$. But then $\{b_1, b_2\} \in \text{Con}$, hence $\{b_1, c\} \in \text{Con}$ by (c). $\square$

Scott (1982) introduced the notion of an *approximable map* from $\mathbf{A}$ to $\mathbf{B}$. Such a map is given by a relation r between Con_A and B , where $r(U, b)$ intuitively means that whenever we are given the information $U \in \text{Con}_A$ on the argument, then we know that at least the token b appears in the value.

DEFINITION (Approximable map). Let $\mathbf{A}$ and $\mathbf{B}$ be acis's. A relation $r \subseteq \text{Con}_A \times B$ is an *approximable map* from $\mathbf{A}$ to $\mathbf{B}$ (written $r: \mathbf{A} \rightarrow \mathbf{B}$) if and only if

- (a) if $r(U, b_1)$ and $r(U, b_2)$, then $\{b_1, b_2\} \in \text{Con}_B$, and
- (b) if $r(U, b)$, $V \geq_A U$ and $b \geq_B c$, then $r(V, c)$.

THEOREM A.0.1. Let $\mathbf{A}$ and $\mathbf{B}$ be acis's. The ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are exactly the approximable maps from $\mathbf{A}$ to $\mathbf{B}$.

PROOF. We show that $r \in |\mathbf{A} \rightarrow \mathbf{B}|$ satisfies the axioms for approximable maps. (a). Let $r(U, b_1)$ and $r(U, b_2)$. Then $\{b_1, b_2\} \in \text{Con}_B$ by the consistency of r . (b). Let $r(U, b)$, $V \geq_A U$ and $b \geq_B c$. Then $(U, b) \geq (V, c)$ by definition, hence $r(V, c)$ by the deductive closure of r .

For the other direction suppose $r: \mathbf{A} \rightarrow \mathbf{B}$ is an approximable map. We must show that $r \in |\mathbf{A} \rightarrow \mathbf{B}|$. Consistency of r : Suppose $r(U_1, b_1)$, $r(U_2, b_2)$ and $U = U_1 \cup U_2 \in \text{Con}_A$. We must show that $\{b_1, b_2\} \in \text{Con}_B$. Now by definition of approximable maps, from $r(U_i, b_i)$ and $U \geq_A U_i$ we obtain $r(U, b_i)$, and hence $\{b_1, b_2\} \in \text{Con}_B$. Deductive closure of r : Suppose $r(U, b)$ and $(U, b) \geq (V, c)$, i.e., $V \geq_A U \wedge b \geq_B c$. Then $r(V, c)$ by definition of approximable maps. $\square$

We can also characterize approximable maps in a different way, which is closer to usual characterizations of continuity¹:

LEMMA A.0.2. Let $\mathbf{A}$ and $\mathbf{B}$ be acis's and $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ monotone (i.e., $x \subseteq y$ implies $f(x) \subseteq f(y)$). Then the following are equivalent.

- (a) f satisfies the “principle of finite support” PFS: If $b \in f(x)$, then $b \in f(\bar{U})$ for some $U \subseteq x$.

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

- (b) f commutes with directed unions: for every directed $D \subseteq |\mathbf{A}|$ (i.e., for any $x, y \in D$ there is a $z \in D$ such that $x, y \subseteq z$)

$$f\left(\bigcup_{x \in D} x\right) = \bigcup_{x \in D} f(x).$$

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 $|\mathbf{B}|$. 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 $D \subseteq |\mathbf{A}|$ 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

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

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

DEFINITION. We call a function $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ *continuous* if it is monotone and satisfies any of the conditions in Lemma A.0.2.

Hence continuous maps $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract objects $x \in |\mathbf{A}|$ and $f(x) \in |\mathbf{B}|$: 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 $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ and $g: |\mathbf{B}| \rightarrow |\mathbf{C}|$.

THEOREM A.0.3. Let $\mathbf{A} = (A, \text{Con}_A, \geq_A)$, $\mathbf{B} = (B, \text{Con}_B, \geq_B)$ be acis's. Then the ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are in a natural bijective correspondence with the continuous functions from $|\mathbf{A}|$ to $|\mathbf{B}|$, as follows.

- (a) With any approximable map $r \subseteq \text{Con}_A \times B$ we can associate a continuous function $|r|: |\mathbf{A}| \rightarrow |\mathbf{B}|$ 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 $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ we can associate an approximable map $\hat{f} \subseteq \text{Con}_A \times B$ 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 $\mathbf{A} \rightarrow \mathbf{B}$; then by Theorem A.0.1 we know that r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\mathbf{A}|$.

$|r|(z)$ is consistent: Let $b_1, b_2 \in |r|(z)$. Then there are $U_1, U_2 \subseteq z$ such that $(U_i, b_i) \in r$. Hence $U := U_1 \cup U_2 \subseteq z$ and $(U, b_i) \in r$ by axiom (b) of approximable maps. Now from axiom (a) we can conclude that $\{b_1, b_2\} \in \text{Con}_B$.

$|r|(z)$ is deductively closed: Let $b_1, b_2 \in |r|(z)$ and $\{b_1, b_2\} \geq_B b$. 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 A.0.2 above, since by definition $|r|$ is monotone and satisfies PFS.

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

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

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

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

hence $r = \widehat{|r|}$. □

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

Equality of two subsets $r, s \subseteq \text{Con}_A \times B$ 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 $U \in \text{Con}_A$.

LEMMA A.0.4 (Extensionality). *Assume that $\mathbf{A} = (A, \text{Con}_A, \geq_A)$ and $\mathbf{B} = (B, \text{Con}_B, \geq_B)$ are acis's and $r, s \subseteq \text{Con}_A \times B$ approximable maps. Then the following are equivalent.*

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

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

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

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.

We define types, built from base types by the formation of function types, $\tau \rightarrow \sigma$; types may involve type variables. For every closed type τ we define an information system $\mathbf{C}_\tau$ whose ideals will be the partial continuous functionals of type τ .

Types will be built from “algebras” (given by a list of “constructor types”) by the formation of “function types” $\tau \rightarrow \sigma$. Types may involve type variables $\alpha, \beta, \gamma, \xi, \zeta$. Iterated arrows are understood as associated to the right. For example, $\alpha \rightarrow \beta \rightarrow \gamma$ means $\alpha \rightarrow (\beta \rightarrow \gamma)$, not $(\alpha \rightarrow \beta) \rightarrow \gamma$.

DEFINITION. *Constructor types* κ have the form

$$\vec{\alpha} \rightarrow (\xi)_{i < n} \rightarrow \xi$$

with all type variables α_i distinct from each other and from ξ . 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 is *recursive* if it has a recursive argument type. 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.

Later for each constructor type κ of a base type we will introduce a constant for a *constructor* and in case it has n parameter argument types also n constants for *destructors*.

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

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

Algebra forms with parameters are

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

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

DEFINITION (Type).

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

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

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

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. $\mathbb{R} := (\mathbb{N} \rightarrow \mathbb{Q}) \times (\mathbb{P} \rightarrow \mathbb{N})$ 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.

For every closed type τ we define the acis $\mathbf{C}_\tau = (C_\tau, \text{Con}_\tau, \geq_\tau)$. The ideals $x \in |\mathbf{C}_\tau|$ are the *partial continuous functionals* of type τ . Since we

will have $C_{\tau \rightarrow \sigma} = C_\tau \rightarrow C_\sigma$, the partial continuous functionals of type $\tau \rightarrow \sigma$ will correspond to the continuous functions from $|C_\tau|$ to $|C_\sigma|$.

DEFINITION (Acis of a closed type τ). We simultaneously define C_ι and $C_{\tau \rightarrow \sigma}$, then Con_ι and $\text{Con}_{\tau \rightarrow \sigma}$ and finally $\geq_\iota$ and $\geq_{\tau \rightarrow \sigma}$

- (a) The *tokens* $a \in C_\iota$ are the type correct constructor expressions $Ca_1^* \dots a_n^*$ where a_i^* is an *extended token*, i.e., a token or the special symbol $*$ which carries no information.
- (b) The tokens in $C_{\tau \rightarrow \sigma}$ are the pairs (U, b) with $U \in \text{Con}_\tau$ and $b \in C_\sigma$.
- (c) A finite set U of tokens in C_ι is *consistent* (i.e., $U \in \text{Con}_\iota$) if all its elements start with the same constructor C , say of arity $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \iota$, and all $U_i \in \text{Con}_{\tau_i}$ for $i = 1, \dots, n$, where U_i consists of all (proper) tokens at the i -th argument position of some token in $U = \{Ca_1^*, \dots, Ca_m^*\}$.
- (d) $\{(U_i, b_i) \mid i \in I\} \in \text{Con}_{\tau \rightarrow \sigma}$ is defined to mean $\{(U_1, b_1), \dots, (U_n, b_n)\} \in \text{Con}_{\tau \rightarrow \sigma} :\Leftrightarrow \forall_{i,j} (U_i \cup U_j \in \text{Con}_\tau \rightarrow \{b_i, b_j\} \in \text{Con}_\sigma)$,
- (e) $C\vec{b}^* \geq_\iota C'\vec{a}^*$ is defined to mean $C = C'$ and $b_i^* \geq a_i^*$ for all i , with $b \geq *$ taken to be true.
- (f) $(U, b) \geq_{\tau \rightarrow \sigma} (V, c) :\Leftrightarrow V \geq_\tau U \wedge b \geq_\sigma c$.

Note that the present definition is by recursion on the *height* of the syntactic expressions involved:

$$\begin{aligned}
|\alpha| &:= 0, \\
|\iota(\tau_1 \dots \tau_n)| &:= 1 + \max\{|\tau_i| \mid i = 1, \dots, n\}, \\
|\tau \rightarrow \sigma| &:= \max\{1 + |\tau|, 1 + |\sigma|\}, \\
|Ca_1^* \dots a_n^*| &:= 1 + \max\{|a_i^*| \mid i = 1, \dots, n\}, \\
|*| &:= 0, \\
|(U, b)| &:= \max\{1 + |U|, 1 + |b|\}, \\
|\{a_i \mid i \in I\}| &:= \max\{1 + |a_i| \mid i \in I\}, \\
|U \vdash a| &:= \max\{1 + |U|, 1 + |a|\}.
\end{aligned}$$

APPENDIX B

Unification

We show that for any two constructor terms one can decide whether there exists a “unifier”, and if so, compute a “most general” one. A solution of this problem has been given by Robinson (1965). In the formulation of the algorithm below we follow Martelli and Montanari (1982).

By a constructor term t, s (term for short) we mean a term built from variables x, y, z and constructors C by application. A *substitution* is a finite set $\vartheta = \{t_1/x_1, \dots, t_n/x_n\}$ of pairs of variables and terms, such that $x_i \neq x_j$ for $i \neq j$, and $t_i \neq x_i$ for all i . An element t_i/x_i of ϑ is called a *binding* (of x_i to t_i). By $t\vartheta$ we denote the result of simultaneously replacing each variable x_i in t by t_i , and call $t\vartheta$ the *instance* of t induced by ϑ . We shall use ϑ, η, ζ for substitutions. Let ε be the empty substitution. For given substitutions

$$\begin{aligned}\vartheta &= \{t_1/x_1, \dots, t_n/x_n\} \\ \eta &= \{s_1/y_1, \dots, s_m/y_m\},\end{aligned}$$

the *composition* $\vartheta\eta$ of ϑ and η is the substitution obtained by deleting in the set

$$\{t_1\eta/x_1, \dots, t_n\eta/x_n, s_1/y_1, \dots, s_m/y_m\}$$

all bindings $t_i\eta/x_i$ such that $t_i\eta = x_i$, and also all bindings s_j/y_j such that $y_j \in \{x_1, \dots, x_n\}$. A substitution ϑ is *idempotent* if $\vartheta\vartheta = \vartheta$. A substitution ϑ is called *more general* than η (written $\eta \leq \vartheta$), if there is a substitution ζ such that $\eta = \vartheta\zeta$. ϑ and η are *equivalent*, if $\vartheta \leq \eta \leq \vartheta$.

It is easy to see that $(t\vartheta)\eta = t(\vartheta\eta)$, and that composition is associative.

We now come to the unification problem. By this we mean the question whether for two given terms t, s there is a substitution ϑ “unifying” the two terms, i.e., with the property $t\vartheta = s\vartheta$.

Let E denote finite equation systems, i.e., multisets

$$\{t_1 = s_1, \dots, t_n = s_n\}$$

of equations between terms (more precisely pairs of terms). Consider $\{\perp\}$ as a (contradictory) equation system. A substitution ϑ *unifies* E , if for every equation $t = s$ in E we have $t\vartheta = s\vartheta$; no ϑ unifies $\{\perp\}$. ϑ is a *most general unifier* (*mgu*) of E , if ϑ is a unifier of E and $\eta \leq \vartheta$ for every unifier η of E .

The following characterization of idempotent mgus will be useful in the proof of the Unification Theorem below.

LEMMA B.0.1 (Characterization of idempotent mgu's). *Let ϑ be a unifier of E . Then ϑ is an idempotent mgu of E iff $\eta = \vartheta\eta$ for all unifiers η of E .*

PROOF. Exercise. □

DEFINITION (Unification algorithm). $E \mapsto_{\vartheta} E'$ is defined by

- (a) $\{t = x\} \cup E \mapsto_{\varepsilon} \{x = t\} \cup E$, if t is not a variable.
- (b) $\{x = x\} \cup E \mapsto_{\varepsilon} E$.
- (c) $\{Ct_1 \dots t_n = Cs_1 \dots s_n\} \cup E \mapsto_{\varepsilon} \{t_1 = s_1, \dots, t_n = s_n\} \cup E$.
- (d) $\{Ct_1 \dots t_n = C's_1 \dots s_n\} \cup E \mapsto_{\varepsilon} \{\perp\}$ if $C \neq C'$.
- (e) $\{x = t, t_1(x) = s_1(x), \dots, t_n(x) = s_n(x)\} \mapsto_{\{t/x\}} \{t_1(t) = s_1(t), \dots, t_n(t) = s_n(t)\}$ if $x \notin \text{FV}(t)$.
- (f) $\{x = t\} \cup E \mapsto_{\varepsilon} \{\perp\}$, if $x \in \text{FV}(t)$ and $t \neq x$.

PROPOSITION. Assume $E \mapsto_{\vartheta} E'$.

- (a) If η' is a unifier of E' , then $\vartheta\eta'$ is a unifier of E .
- (b) If η is a unifier of E , then $\eta = \vartheta\eta$ and η is a unifier of E' .

PROOF. By cases according to the definition of $E \mapsto_{\vartheta} E'$. Clearly it suffices to treat case (e).

Let η' be a unifier of E' . Then $\{t/x\}\eta'$ is a unifier of E .

Let η be a unifier of E . Then $x\eta = t\eta$, hence $\eta = \{t/x\}\eta$ (since both substitutions coincide on all variables), and moreover

$$t_i\{t/x\}\eta = t_i\eta = s_i\eta = s_i\{t/x\}\eta.$$

Hence η is a unifier of E' . □

COROLLARY. Assume

$$E_1 \mapsto_{\vartheta_1} E_2 \mapsto_{\vartheta_2} \dots E_n \mapsto_{\vartheta_n} E_{n+1}.$$

- (a) If ϑ is a unifier of E_{n+1} , then $\vartheta_1 \dots \vartheta_n \vartheta$ is a unifier of E_1 .
- (b) If η is a unifier of E_1 , then $\eta = \vartheta_1 \dots \vartheta_n \eta$ and η is a unifier of E_{n+1} .

PROOF. The first part clearly follows from the first part of the Proposition. The second part is proved by induction on n . For $n = 0$ there is nothing to show. In the step we split the assumption into

$$E_1 \mapsto_{\vartheta_1} E_2 \quad \text{and} \quad E_2 \mapsto_{\vartheta_2} \dots E_n \mapsto_{\vartheta_n} E_{n+1}.$$

By the second part of the Proposition we have that $\eta = \vartheta_1\eta$ is a unifier of E_2 . Hence by IH $\eta = \vartheta_2 \dots \vartheta_n \eta$ is a unifier of E_{n+1} . Moreover we have $\eta = \vartheta_1\eta = \vartheta_1\vartheta_2 \dots \vartheta_n \eta$. □

UNIFICATION THEOREM. *Let E be a finite equation system. Then every sequence*

$$E = E_1 \mapsto_{\vartheta_1} E_2 \mapsto_{\vartheta_2} \dots$$

terminates with $E_{n+1} = \emptyset$ or $E_{n+1} = \{\perp\}$. In the first case E is unifiable, and $\vartheta_1 \dots \vartheta_n$ is an idempotent mgu of E . In the second case E is not unifiable.

PROOF. Exercise. Termination is proved using the lexicographic ordering of $\mathbb{N}^3$. To every $E = \{t_1 = s_1, \dots, t_n = s_n\}$ assign a triple $(n_1, n_2, n_3) \in \mathbb{N}^3$ by

$n_1 :=$ number of variables in E ,

$n_2 :=$ number of occurrences of variables and constructors in E ,

$n_3 :=$ number of equations $t = x$ in E such that t is not a variable. $\square$

APPENDIX C

Denotational semantics

We set up a connection between the model $(|\mathbf{C}_\rho|)_\rho$ of partial continuous functionals described in Section 2.2.2 and the term system T^+ from Section 2.3.1. The main point is to clarify how we can use computation rules to define an ideal z in a function space.

C.1. Ideals as values of terms

The general idea is to inductively define the set of tokens (U, a) that make up z . It is convenient to define the value $\llbracket \lambda_{\vec{x}} M \rrbracket$, where M is a term with free variables among $\vec{x}$. Since this value is a token set, we can define inductively the relation $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$.

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

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

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

DEFINITION C.1.1 (Inductive, of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$).

$$\frac{U_i \vdash a}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket}(V), \quad \frac{(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket(A)}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}$$

For every constructor C and defined constant D we have

$$\frac{\vec{V} \vdash a^*}{(\vec{U}, \vec{V}, Ca^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket}(C), \quad \frac{(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} M \rrbracket \quad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, a) \in \llbracket \lambda_{\vec{x}} D \rrbracket}(D)$$

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

The *height* of a derivation of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ is defined as usual, by adding 1 at each rule. We define its *D-height* similarly, where only rules (D) count.

We begin with some simple consequences of this definition.

LEMMA C.1.2. *The following transformations preserve D -height:*

- (26) $\vec{V} \vdash \vec{U} \rightarrow (\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \rightarrow (\vec{V}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket,$
- (27) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket \leftrightarrow (\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad \text{if } y \notin \text{FV}(M),$
- (28) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y} (My) \rrbracket \leftrightarrow (\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad \text{if } y \notin \text{FV}(M),$
- (29) $(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} (M(\vec{P}(\vec{y}))) \rrbracket \leftrightarrow (\vec{U}, \vec{P}(\vec{V}), a) \in \llbracket \lambda_{\vec{x}, \vec{z}} (M(\vec{z})) \rrbracket.$

PROOF. (26) and (27) are both proved by easy inductions on the respective derivations.

(28). Assume $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y} (My) \rrbracket$. By (A) we then have W such that $(\vec{U}, V, W) \subseteq \llbracket \lambda_{\vec{x}, y} y \rrbracket$ (i.e., $V \vdash W$) and $(\vec{U}, V, W, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. By (26) from the latter we obtain $(\vec{U}, V, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. Now since $y \notin \text{FV}(M)$, (27) yields $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, as required. Conversely, assume $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$. Since $y \notin \text{FV}(M)$, (27) yields $(\vec{U}, V, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. Clearly we have $(\vec{U}, V, V) \subseteq \llbracket \lambda_{\vec{x}, y} y \rrbracket$. Hence by (A) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y} (My) \rrbracket$, as required. Notice that the D -height did not change in these transformations.

(29). By induction on $\vec{P}$, with a side induction on M . We distinguish cases on M . The cases x_i , C and D follow immediately from (27). In case MN the following are equivalent by the side induction hypothesis:

$$\begin{aligned}
& (\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} ((MN)(\vec{P}(\vec{y}))) \rrbracket \\
& \exists W ((\vec{U}, \vec{V}, W, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} (M(\vec{P}(\vec{y}))) \rrbracket \wedge (\vec{U}, \vec{V}, W) \subseteq \llbracket \lambda_{\vec{x}, \vec{y}} (N(\vec{P}(\vec{y}))) \rrbracket) \\
& \exists W ((\vec{U}, \vec{P}(\vec{V}), W, a) \in \llbracket \lambda_{\vec{x}, \vec{z}} (M(\vec{z})) \rrbracket \wedge (\vec{U}, \vec{P}(\vec{V}), W) \subseteq \llbracket \lambda_{\vec{x}, \vec{z}} (N(\vec{z})) \rrbracket) \\
& (\vec{U}, \vec{P}(\vec{V}), a) \in \llbracket \lambda_{\vec{x}, \vec{z}} ((MN)(\vec{z})) \rrbracket.
\end{aligned}$$

The final case is where M is z_i . Then we have to show

$$(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} (P(\vec{y})) \rrbracket \leftrightarrow P(\vec{V}) \vdash a.$$

We distinguish cases on $P(\vec{y})$. If $P(\vec{y})$ is y_j , then both sides are equivalent to $V_j \vdash a$. In case $P(\vec{y})$ is $(C\vec{Q})(\vec{y})$ the following are equivalent:

$$\begin{aligned}
& (\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} ((C\vec{Q})(\vec{y})) \rrbracket \\
& (\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} (C\vec{Q}(\vec{y})) \rrbracket \\
& (\vec{U}, \vec{Q}(\vec{V}), a) \in \llbracket \lambda_{\vec{x}, \vec{u}} (C\vec{u}) \rrbracket \quad \text{by IH for } \vec{Q}(\vec{y}) \\
& (\vec{U}, \vec{Q}(\vec{V}), a) \in \llbracket \lambda_{\vec{x}} C \rrbracket \quad \text{by (28)} \\
& \exists_{a^*} (a = Ca^* \wedge \vec{Q}(\vec{V}) \vdash a^*) \\
& C\vec{Q}(\vec{V}) \vdash a.
\end{aligned}$$

□

Let $\sim$ denote the equivalence relation on formal neighborhoods generated by entailment, i.e., $U \sim V$ means $(U \vdash V) \wedge (V \vdash U)$.

LEMMA C.1.3.

(30) *If $\vec{U} \vdash \vec{P}(\vec{V})$, then there are $\vec{W}$ such that $\vec{U} \sim \vec{P}(\vec{W})$ and $\vec{W} \vdash \vec{V}$.*

PROOF. By induction on $\vec{P}$. The cases x and $\langle \rangle$ are clear, and in case $\vec{P}, Q$ we can apply the induction hypothesis. It remains to treat the case $C\vec{P}(\vec{x})$. Since $U \vdash C\vec{P}(\vec{V})$ there is a $\vec{b}_0^*$ such that $C\vec{b}_0^* \in U$. Let

$$U_i := \{a \mid \exists_{a^*} (Ca^* \in U \wedge a = a_i^*)\}.$$

For the constructor pattern $C\vec{x}$ consider $C\vec{U}$. By definition

$$C\vec{U} = \{Ca^* \mid a_i^* \in U_i \text{ if } U_i \neq \emptyset, \text{ and } a_i^* = * \text{ otherwise}\}.$$

We first show $U \sim C\vec{U}$. Assume $Ca^* \in C\vec{U}$. For each i , if $U_i \neq \emptyset$, then there is an a_i^* such that $Ca_i^* \in U$ and $a_i^* = a_i^*$, and if $U_i = \emptyset$ then $a_i^* = *$. Hence

$$U \supseteq \{Ca_i^* \mid U_i \neq \emptyset\} \cup \{Cb_0^*\} \vdash Ca^*.$$

Conversely assume $Ca^* \in U$. We define $Cb^* \in C\vec{U}$ by $b_i^* = a_i^*$ if $a_i^* \neq *$, $b_i^* = *$ if $U_i = \emptyset$, and otherwise (i.e., if $a_i^* = *$ and $U_i \neq \emptyset$) take an arbitrary $b_i^* \in U_i$. Clearly $\{Cb^*\} \vdash Ca^*$.

By definition $\vec{U} \vdash \vec{P}(\vec{V})$. Hence by induction hypothesis there are $\vec{W}$ such that $\vec{U} \sim \vec{P}(\vec{W})$ and $\vec{W} \vdash \vec{V}$. Therefore $U \sim C\vec{U} \sim C\vec{P}(\vec{W})$. $\square$

LEMMA C.1.4 (Unification). *If $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$, then $\vec{P}_1, \dots, \vec{P}_n$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that*

$$(\vec{P}_1\vartheta)(\vec{W}) = \dots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n).$$

PROOF. Assume $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$. Then $\vec{P}_1(\vec{V}_1), \dots, \vec{P}_n(\vec{V}_n)$ are componentwise consistent and hence $\vec{P}_1, \dots, \vec{P}_n$ are unifiable with a most general unifier ϑ . We now proceed by induction on $\vec{P}_1, \dots, \vec{P}_n$. If they are either all empty or all variables the claim is trivial. In the case $(\vec{P}_1, P_1), \dots, (\vec{P}_n, P_n)$ it follows from the linearity condition on variables that a most general unifier of $(\vec{P}_1, P_1), \dots, (\vec{P}_n, P_n)$ is the union of most general unifiers of $\vec{P}_1, \dots, \vec{P}_n$ and of $P_1, \dots, P_n$. Hence the induction hypothesis applies. In the case $C\vec{P}_1, \dots, C\vec{P}_n$ the assumption $C\vec{P}_1(\vec{V}_1) \sim \dots \sim C\vec{P}_n(\vec{V}_n)$ implies $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$ and hence again the induction hypothesis applies. The remaining case is when some are variables and the other ones of the form $C\vec{P}_i$, say $x, C\vec{P}_2, \dots, C\vec{P}_n$. By assumption

$$V_1 \sim C\vec{P}_2(\vec{V}_2) \sim \dots \sim C\vec{P}_n(\vec{V}_n).$$

By induction hypothesis we obtain the required $\vec{W}$ such that

$$(\vec{P}_2\vartheta)(\vec{W}) = \dots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_2(\vec{V}_2) \sim \dots \sim \vec{P}_n(\vec{V}_n). \quad \square$$

LEMMA C.1.5 (Consistency). $\llbracket \lambda_{\vec{x}} M \rrbracket$ is consistent.

PROOF. Let $(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ for $i = 1, 2$. By coherence it suffices to prove that $(\vec{U}_1, a_1)$ and $(\vec{U}_2, a_2)$ are consistent. We shall prove this by induction on the maximum of the D -heights and a side induction on the maximum of the heights.

Case (V). Let $(\vec{U}_1, a_1), (\vec{U}_2, a_2) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket$, and assume that $\vec{U}_1$ and $\vec{U}_2$ are componentwise consistent. Then $U_{1i} \vdash a_1$ and $U_{2i} \vdash a_2$. Since $U_{1i} \cup U_{2i}$ is consistent, a_1 and a_2 must be consistent as well.

Case (C). For $i = 1, 2$ we have

$$\frac{\vec{V}_i \vdash a_i^*}{(\vec{U}_i, \vec{V}_i, Ca_i^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket}.$$

Assume $\vec{U}_1, \vec{V}_1$ and $\vec{U}_2, \vec{V}_2$ are componentwise consistent. The consistency of Ca_1^* and Ca_2^* follows from $\vec{V}_i \vdash a_i^*$ and the consistency of $\vec{V}_1$ and $\vec{V}_2$.

Case (A). For $i = 1, 2$ we have

$$\frac{(\vec{U}_i, V_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}_i, V_i) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}.$$

Assume $\vec{U}_1$ and $\vec{U}_2$ are componentwise consistent. By the side induction hypothesis for the right premises $V_1 \cup V_2$ is consistent. Hence by the side induction hypothesis for the left hand sides a_1 and a_2 are consistent.

Case (D). For $i = 1, 2$ we have

$$\frac{(\vec{U}_i, \vec{V}_i, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket \quad \vec{W}_i \vdash \vec{P}_i(\vec{V}_i)}{(\vec{U}_i, \vec{W}_i, a_i) \in \llbracket \lambda_{\vec{x}} D \rrbracket} (D)$$

for computation rules $D\vec{P}_i(\vec{y}_i) = M_i(\vec{y}_i)$. Assume $\vec{U}_1, \vec{W}_1$ and $\vec{U}_2, \vec{W}_2$ are componentwise consistent; we must show that a_1 and a_2 are consistent. Since $\vec{W}_1 \cup \vec{W}_2 \vdash \vec{P}_i(\vec{V}_i)$ for $i = 1, 2$, by (30) there are $\vec{V}_1', \vec{V}_2'$ such that $\vec{V}_i' \vdash \vec{V}_i$ and $\vec{W}_1 \cup \vec{W}_2 \sim \vec{P}_i(\vec{V}_i')$. Then by Lemma C.1.4 (on unification) there are $\vec{W}$ such that $(\vec{P}_1\vartheta)(\vec{W}) = (\vec{P}_2\vartheta)(\vec{W}) \sim \vec{P}_i(\vec{V}_i') \vdash \vec{P}_i(\vec{V}_i)$ for $i = 1, 2$, where ϑ is the most general unifier of $\vec{P}_1$ and $\vec{P}_2$. But then also

$$(\vec{y}_i\vartheta)(\vec{W}) \vdash \vec{V}_i,$$

and hence by (26) we have

$$(\vec{U}_i, (\vec{y}_i\vartheta)(\vec{W}), a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket$$

with smaller D -height. Now (29) gives

$$(\vec{U}_i, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{z}} M_i(\vec{y}_i) \vartheta \rrbracket$$

without increasing the D -height. Notice that $M_1(\vec{y}_i)\vartheta = M_2(\vec{y}_i)\vartheta$ by our condition on computation rules. Hence the induction hypothesis applied to $(\vec{U}_1, \vec{W}, a_1), (\vec{U}_2, \vec{W}, a_2) \in \llbracket \lambda_{\vec{x}, \vec{z}} M_1(\vec{y}_1)\vartheta \rrbracket$ implies the consistency of a_1 and a_2 , as required. $\square$

LEMMA C.1.6 (Deductive closure). *$\llbracket \lambda_{\vec{x}} M \rrbracket$ is deductively closed, i.e., if $W \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$ and $W \vdash (\vec{V}, b)$, then $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$.*

PROOF. By induction on the maximum of the D -heights and a side induction on the maximum of the heights of $W \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$. We distinguish cases on the last rule of these derivations (which is determined by M).

Case (V). For all $(\vec{U}, a) \in W$ we have

$$\frac{U_i \vdash a}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket}.$$

We must show $V_i \vdash b$. By assumption $W \vdash (\vec{V}, b)$, hence $W\vec{V} \vdash b$. It suffices to prove $V_i \vdash W\vec{V}$. Let $c \in W\vec{V}$; we show $V_i \vdash c$. There are $\vec{U}$ such that $\vec{V} \vdash \vec{U}$ and $(\vec{U}, c) \in W$. But then by the above $U_i \vdash c$, hence $V_i \vdash U_i \vdash c$.

Case (A). Let $W = \{(\vec{U}_1, a_1), \dots, (\vec{U}_n, a_n)\}$. For each $(\vec{U}_i, a_i) \in W$ there is U_i such that

$$\frac{(\vec{U}_i, U_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}_i, U_i) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}.$$

Define $U := \bigcup \{U_i \mid \vec{V} \vdash \vec{U}_i\}$. We first show that U is consistent. Let $a, b \in U$. There are i, j such that $a \in U_i$, $b \in U_j$ and $\vec{V} \vdash \vec{U}_i, \vec{U}_j$. Then $\vec{U}_i$ and $\vec{U}_j$ are consistent; hence since $\llbracket \lambda_{\vec{x}} N \rrbracket$ is consistent by Lemma C.1.5 a and b are consistent as well.

Next we show $(\vec{V}, U) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket$. Let $a \in U$; we show $(\vec{V}, a) \in \llbracket \lambda_{\vec{x}} N \rrbracket$. Fix i such that $a \in U_i$ and $\vec{V} \vdash \vec{U}_i$, and let $W_i := \{(\vec{U}_i, b) \mid b \in U_i\} \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket$. Since by the side induction hypothesis $\llbracket \lambda_{\vec{x}} N \rrbracket$ is deductively closed it suffices to prove $W_i \vdash (\vec{V}, a)$, i.e., $\{b \mid b \in U_i \wedge \vec{V} \vdash \vec{U}_i\} \vdash a$. But the latter set equals U_i , and $a \in U_i$.

Finally we show $(\vec{V}, U, b) \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$. Let

$$W' := \{(\vec{U}_1, U_1, a_1), \dots, (\vec{U}_n, U_n, a_n)\} \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket.$$

By side induction hypothesis it suffices to prove that $W' \vdash (\vec{V}, U, b)$, i.e., $\{a_i \mid \vec{V} \vdash \vec{U}_i \wedge U \vdash U_i\} \vdash b$. But by definition of U the latter set equals $\{a_i \mid \vec{V} \vdash \vec{U}_i\}$, which in turn entails b because by assumption $W \vdash (\vec{V}, b)$.

Now we can use (A) to infer $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, as required.

Case (C). Assume $W \subseteq \llbracket \lambda_{\vec{x}} C \rrbracket$. Then W consists of $(\vec{U}, \vec{U}', C\vec{a}^*)$ such that $\vec{U}' \vdash \vec{a}^*$. Assume further $W \vdash (\vec{V}, \vec{V}', b)$. Then

$$\{ C\vec{a}^* \mid \exists \vec{U}, \vec{U}', ((\vec{U}, \vec{U}', C\vec{a}^*) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}') \} \vdash b.$$

By definition of entailment b has the form Cb^* such that

$$W_i := \{ a \mid \exists \vec{U}, \vec{U}', \vec{a}^* (a = a_i^* \wedge (\vec{U}, \vec{U}', C\vec{a}^*) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}') \} \vdash b_i^*.$$

We must show $(\vec{V}, \vec{V}', Cb^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket$, i.e., $\vec{V}' \vdash b^*$. It suffices to show $V_i' \vdash W_i$, for every i . Let $a \in W_i$. Then there are $\vec{U}, \vec{U}', \vec{a}^*$ such that $a = a_i^*$, $(\vec{U}, \vec{U}', C\vec{a}^*) \in W$ and $\vec{V}' \vdash \vec{U}'$. Hence $V_i' \vdash U_i' \vdash a_i^* = a$.

Case (D). Let $W = \{(\vec{U}_1, \vec{U}_1'', a_1), \dots, (\vec{U}_n, \vec{U}_n'', a_n)\}$. For every i there is an $\vec{U}_i'$ such that

$$\frac{(\vec{U}_i, \vec{U}_i', a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket \quad \vec{U}_i'' \vdash \vec{P}_i(\vec{U}_i')}{(\vec{U}_i, \vec{U}_i'', a_i) \in \llbracket \lambda_{\vec{x}} D \rrbracket}$$

for $D\vec{P}_i(\vec{y}_i) = M_i(\vec{y}_i)$ a computation rule. Assume $W \vdash (\vec{V}, \vec{V}'', b)$. We must prove $(\vec{V}, \vec{V}'', b) \in \llbracket \lambda_{\vec{x}} D \rrbracket$. Let

$$I := \{ i \mid 1 \leq i \leq n \wedge \vec{V} \vdash \vec{U}_i \wedge \vec{V}'' \vdash \vec{U}_i'' \}.$$

Then $\{ a_i \mid i \in I \} \vdash b$, hence $I \neq \emptyset$. For $i \in I$ we have $\vec{V}'' \vdash \vec{U}_i'' \vdash \vec{P}_i(\vec{U}_i')$, hence by (30) there are $\vec{V}_i'$ such that $\vec{V}'' \sim \vec{P}_i(\vec{V}_i')$ and $\vec{V}_i' \vdash \vec{U}_i'$. In particular for $i, j \in I$

$$\vec{V}'' \sim \vec{P}_i(\vec{V}_i') \sim \vec{P}_j(\vec{V}_j').$$

To simplify notation assume $I = \{1, \dots, m\}$. Hence by the unification lemma $\vec{P}_1, \dots, \vec{P}_m$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that

$$(\vec{P}_1\vartheta)(\vec{W}) = \dots = (\vec{P}_m\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1') \sim \dots \sim \vec{P}_m(\vec{V}_m').$$

Let $i, j \in I$. Then by the conditions on computation rules $M_i\vartheta = M_j\vartheta$. Also $(\vec{y}_i\vartheta)(\vec{W}) \vdash \vec{V}_i' \vdash \vec{U}_i'$. Therefore by (26)

$$(\vec{V}, (\vec{y}_i\vartheta)(\vec{W}), a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket$$

and hence by (29)

$$(\vec{V}, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i\vartheta) \rrbracket.$$

But $M_i(\vec{y}_i\vartheta) = M_i\vartheta = M_1\vartheta = M_1(\vec{y}_1\vartheta)$ and hence for all $i \in I$

$$(\vec{V}, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1\vartheta) \rrbracket.$$

Therefore $X := \{(\vec{V}, \vec{W}, a_i) \mid i \in I\} \subseteq \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1 \vartheta) \rrbracket$. Since $\{a_i \mid i \in I\} \vdash b$, we have $X \vdash (\vec{V}, \vec{W}, b)$ and hence the induction hypothesis implies $(\vec{V}, \vec{W}, b) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1 \vartheta) \rrbracket$. Using (29) again we obtain $(\vec{V}, (\vec{y}_i \vartheta)(\vec{W}), b) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1) \rrbracket$. Since $\vec{V}'' \sim \vec{P}_1(\vec{V}_1') \sim \vec{P}_1((\vec{y}_1 \vartheta)(\vec{W}))$ we obtain $(\vec{V}, \vec{V}'', b) \in \llbracket \lambda_{\vec{x}} D \rrbracket$, by (D). $\square$

COROLLARY. $\llbracket \lambda_{\vec{x}} M \rrbracket$ is an ideal.

C.2. Preservation of values

We now prove that Definition C.1.1 of the denotation of a term is reasonable in the sense that this denotation is not changed by an application of the standard (β - and η -) conversions or a computation rule. For the β -conversion part of this proof it is helpful to first introduce a more standard notation, which involves variable environments.

DEFINITION. Assume that all free variables in M are among $\vec{x}$. Let $\llbracket M \rrbracket_{\vec{x}}^{\vec{U}} := \{b \mid (\vec{U}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket\}$ and $\llbracket M \rrbracket_{\vec{x}, \vec{y}}^{\vec{u}, \vec{V}} := \bigcup_{\vec{U} \subseteq \vec{u}} \llbracket M \rrbracket_{\vec{x}, \vec{y}}^{\vec{U}, \vec{V}}$.

From (27) we obtain $\llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V} = \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$ if $y \notin \text{FV}(M)$, and similarly for ideals $\vec{u}, v$ instead of $\vec{U}, V$. We have a useful monotonicity property, which follows from the deductive closure of $\llbracket \lambda_{\vec{x}} M \rrbracket$.

LEMMA C.2.1. (a) If $\vec{V} \vdash \vec{U}$, $a \vdash b$ and $a \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$, then $b \in \llbracket M \rrbracket_{\vec{x}}^{\vec{V}}$.
 (b) If $\vec{v} \supseteq \vec{u}$, $a \vdash b$ and $a \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}$, then $b \in \llbracket M \rrbracket_{\vec{x}}^{\vec{v}}$.

PROOF. (a) $\vec{V} \vdash \vec{U}$, $a \vdash b$ and $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ together imply $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, by the deductive closure of $\llbracket \lambda_{\vec{x}} M \rrbracket$. (b) follows from (a). $\square$

LEMMA C.2.2. (a) $\llbracket x_i \rrbracket_{\vec{x}}^{\vec{U}} = \overline{U}_i$ and $\llbracket x_i \rrbracket_{\vec{x}}^{\vec{u}} = u_i$.
 (b) $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{U}} = \{(V, b) \mid b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V}\}$ and $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}} = \{(V, b) \mid b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, V}\}$.
 (c) $\llbracket MN \rrbracket_{\vec{x}}^{\vec{U}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}$ and $\llbracket MN \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{u}} \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}$.

PROOF. (b) It suffices to prove the first part. But $(V, b) \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{U}}$ and $b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V}$ are both equivalent to $(\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$.

(c) For the first part we argue as follows.

$$\begin{aligned}
 c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}} &\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\
 &\leftrightarrow \exists_V ((\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket \wedge (\vec{U}, V, c) \in \llbracket \lambda_{\vec{x}} M \rrbracket) \\
 &\leftrightarrow (\vec{U}, c) \in \llbracket \lambda_{\vec{x}} (MN) \rrbracket \quad \text{by (A)} \\
 &\leftrightarrow c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{U}}.
 \end{aligned}$$

The second part is an easy consequence:

$$\begin{aligned}
c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}} \llbracket N \rrbracket_{\vec{x}}^{\vec{u}} &\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}) \\
&\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}} \exists_{\vec{U} \subseteq \vec{u}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\
&\leftrightarrow \exists_{\vec{U}_1 \subseteq \vec{u}} \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_1}} \exists_{\vec{U} \subseteq \vec{u}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\
&\leftrightarrow^{(*)} \exists_{\vec{U} \subseteq \vec{u}} \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\
&\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}) \\
&\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{U}}) \quad \text{by the first part} \\
&\leftrightarrow c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{u}}.
\end{aligned}$$

Here is the proof of the equivalence marked (*). The upward direction is obvious. For the downward direction we use monotonicity. Assume $\vec{U}_1 \subseteq \vec{u}$, $V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_1}$, $\vec{U} \subseteq \vec{u}$ and $(V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$. Let $\vec{U}_2 := \vec{U}_1 \cup \vec{U} \subseteq \vec{u}$. Then by monotonicity $V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_2}$ and $(V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}_2}$. $\square$

COROLLARY. $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}} v = \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, v}$.

PROOF.

$$\begin{aligned}
b \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}} v &\leftrightarrow \exists_{V \subseteq v} ((V, b) \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}}) \\
&\leftrightarrow \exists_{V \subseteq v} (b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, V}) \quad \text{by the lemma, part (b)} \\
&\leftrightarrow b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, v}. \quad \square
\end{aligned}$$

LEMMA C.2.3 (Substitution). $\llbracket M(z) \rrbracket_{\vec{x}, z}^{\vec{u}, \llbracket N \rrbracket_{\vec{z}}^{\vec{u}}}} = \llbracket M(N) \rrbracket_{\vec{x}}^{\vec{u}}$.

PROOF. By induction on M , and cases on the form of M .

Case $\lambda_y M$. For readability we leave out $\vec{x}$ and $\vec{u}$.

$$\begin{aligned}
\llbracket \lambda_y M(z) \rrbracket_z^{\llbracket N \rrbracket} &= \{ (V, b) \mid b \in \llbracket M(z) \rrbracket_{z, y}^{\llbracket N \rrbracket, V} \} \\
&= \{ (V, b) \mid b \in \llbracket M(N) \rrbracket_y^V \} \quad \text{by induction hypothesis} \\
&= \llbracket \lambda_y M(N) \rrbracket \quad \text{by the last lemma, part (b)} \\
&= \llbracket (\lambda_y M)(N) \rrbracket.
\end{aligned}$$

The other cases are easy. $\square$

LEMMA C.2.4 (Preservation of values, β). $\llbracket (\lambda_y M(y))N \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M(N) \rrbracket_{\vec{x}}^{\vec{u}}$.

PROOF. Again we leave out $\vec{x}$, $\vec{u}$. By the last two lemmata and the corollary, $\llbracket (\lambda_y M(y))N \rrbracket = \llbracket \lambda_y M(y) \rrbracket \llbracket N \rrbracket = \llbracket M(y) \rrbracket_y^{\llbracket N \rrbracket} = \llbracket M(N) \rrbracket$. $\square$

LEMMA C.2.5 (Preservation of values, η). $\llbracket \lambda_y(My) \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}$ if $y \notin \text{FV}(M)$.

PROOF.

$$\begin{aligned} (V, b) \in \llbracket \lambda_y(My) \rrbracket_{\vec{x}}^{\vec{u}} &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} ((\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}, y}(My) \rrbracket) \\ &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} ((\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket) \quad \text{by (28)} \\ &\leftrightarrow (V, b) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}. \quad \square \end{aligned}$$

LEMMA C.2.6 (Preservation of values, computation rules). *For every computation rule $D\vec{P}(\vec{y}) = M$ of a defined constant D*

$$\llbracket \lambda_{\vec{y}} D\vec{P}(\vec{y}) \rrbracket = \llbracket \lambda_{\vec{y}} M \rrbracket.$$

PROOF. The following are equivalent:

$$\begin{aligned} (\vec{V}, b) &\in \llbracket \lambda_{\vec{y}} D\vec{P}(\vec{y}) \rrbracket \\ (\vec{P}(\vec{V}), b) &\in \llbracket D \rrbracket = \llbracket \lambda_{\vec{z}} D\vec{z} \rrbracket \quad \text{by (29)} \\ (\vec{V}, b) &\in \llbracket \lambda_{\vec{y}} M \rrbracket, \end{aligned}$$

where the last step is by definition. $\square$

APPENDIX D

Realizers for axioms

We show that the extracted term of $I^\pm$, ${}^{\text{co}}I^\pm$ 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 D.0.1. *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 $(\sim_{\mathbb{L}})_1^+$:

$$\forall_{s_1, s_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \forall_{\ell_1, \ell_2} (\ell_1 \sim_{\mathbb{L}} \ell_2 \rightarrow s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2))$$

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

$$:: \mathbf{r} (s_1 \sim_{\text{sd}} s_2 \rightarrow \forall_{\ell_1, \ell_2} (\ell_1 \sim_{\mathbb{L}} \ell_2 \rightarrow s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2)).$$

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

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

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

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

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

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

LEMMA D.0.2. *The recursion operator $\mathcal{R}_{\mathbb{L}}^\alpha$ realizes the least-fixed-point axiom $\sim_{\mathbb{L}}^-$.*

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

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

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

$$\mathbb{L} \rightarrow \alpha \rightarrow (\text{sd} \rightarrow \mathbb{L} \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha.$$

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

$$\begin{aligned} & \mathcal{R}_{\perp}^{\alpha} \mathbf{r} (\ell_1 \sim_{\perp} \ell_2 \rightarrow \\ & \quad X[] \rightarrow \\ & \quad \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\perp} \ell_2 \rightarrow X\ell_1\ell_2 \rightarrow X(s_1 :: \ell_1, s_2 :: \ell_2)) \rightarrow \\ & \quad X\ell_1\ell_2). \end{aligned}$$

Pick ℓ with $\sim_{\perp}^{\mathbf{r}}(\ell_1, \ell_2, \ell)$. Then the goal is

$$\begin{aligned} & \mathcal{R}_{\perp}^{\alpha} \ell \mathbf{r} (X[] \rightarrow \\ & \quad \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\perp} \ell_2 \rightarrow X\ell_1\ell_2 \rightarrow X(s_1 :: \ell_1, s_2 :: \ell_2)) \rightarrow \\ & \quad X\ell_1\ell_2). \end{aligned}$$

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

$$\mathcal{R}_{\perp}^{\alpha} \ell x \mathbf{r} (\forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\perp} \ell_2 \rightarrow X\ell_1\ell_2 \rightarrow X(s_1 :: \ell_1, s_2 :: \ell_2)) \rightarrow X\ell_1\ell_2).$$

Pick f with

$$f \mathbf{r} \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\perp} \ell_2 \rightarrow X\ell_1\ell_2 \rightarrow X(s_1 :: \ell_1, s_2 :: \ell_2)),$$

which implies

$$\begin{aligned} & \forall_{s_1, s_2, s, \ell_1, \ell_2, \ell, y} (\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s) \rightarrow \sim_{\perp}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \rightarrow X^{\mathbf{r}}\ell_1\ell_2y \rightarrow \\ & \quad X^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, fsl y)). \end{aligned}$$

Our goal is

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

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

$$\begin{aligned} & \forall_{\ell_1, \ell_2, \ell} (\sim_{\perp}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \rightarrow \\ & \quad Q[] \rightarrow \\ & \quad \forall_{s_1, s_2, s, \ell_1, \ell_2, \ell} (\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s) \rightarrow \sim_{\perp}^{\mathbf{r}}(\ell_1, \ell_2, \ell) \rightarrow Q\ell_1\ell_2\ell \rightarrow \\ & \quad \quad Q(s_1 :: \ell_1, s_2 :: \ell_2, s :: \ell)) \rightarrow \\ & \quad Q\ell_1\ell_2\ell). \end{aligned}$$

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

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

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

$$X^{\mathbf{r}}(s_1 :: \ell_1, s_2 :: \ell_2, fsl(\mathcal{R}_{\perp}^{\alpha} \ell x f)).$$

But with $y := \mathcal{R}_{\perp}^{\alpha} \ell x f$ this follows from what we have. $\square$

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

LEMMA D.0.3. *The destructor $\mathcal{D}_{\mathbb{S}}$ realizes the closure axiom $\approx_{\mathbb{S}}^{-}$.*

PROOF. Recall $\approx_{\mathbb{S}}^{-}$:

$$\forall_{u_1, u_2} (u_1 \approx_{\mathbb{S}} u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2 \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2))$$

with cotype ${}^{\text{co}}\mathbb{S} \rightarrow \text{sd} \times {}^{\text{co}}\mathbb{S}$. The goal is $\mathcal{D}_{\mathbb{S}} \mathbf{r} \approx_{\mathbb{S}}^{-}$, which unfolds into

$$\forall_{u_1, u_2, u} (\approx_{\mathbb{S}}^{\mathbf{r}}(u_1, u_2, u) \rightarrow \mathcal{D}_{\mathbb{S}} u \mathbf{r} \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2 \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)).$$

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

$$\exists_{s_1, s_2, u'_1, u'_2} (\mathcal{D}_{\mathbb{S}} u \mathbf{r} (s_1 \sim_{\text{sd}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2) \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2).$$

By $(\approx_{\mathbb{S}}^{\mathbf{r}})^{-}$ from $\approx_{\mathbb{S}}^{\mathbf{r}}(u_1, u_2, u)$ we obtain $s_1, s_2, s, u'_1, u'_2, u'$ such that

$$\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s) \wedge \approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u') \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2 \wedge u \equiv s :: u'.$$

Take s_1, s_2, u'_1, u'_2 . It remains to show $\mathcal{D}_{\mathbb{S}} u \mathbf{r} (s_1 \sim_{\text{sd}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2)$. By the computation rule of $\mathcal{D}_{\mathbb{S}}$ we know $\mathcal{D}_{\mathbb{S}} u \equiv \mathcal{D}_{\mathbb{S}}(s :: u') \equiv \langle s, u' \rangle$. Hence we must prove $\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s)$ and $\approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u')$, which we both have. $\square$

LEMMA D.0.4. *The corecursion operator ${}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}$ realizes the greatest-fixed-point axiom $\approx_{\mathbb{S}}^{+}$.*

PROOF. We equivalently rewrite $\approx_{\mathbb{S}}^{+}$ as $C :=$

$$\begin{aligned} \forall_{u_1, u_2} (X u_1 u_2 \rightarrow & \forall_{u_1, u_2} (X u_1 u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee X u'_1 u'_2) \wedge \\ & u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \rightarrow \\ & u_1 \approx_{\mathbb{S}} u_2) \end{aligned}$$

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

$$\alpha \rightarrow (\alpha \rightarrow \text{sd} \times ({}^{\text{co}}\mathbb{S} + \alpha)) \rightarrow {}^{\text{co}}\mathbb{S}.$$

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

$$\begin{aligned} {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha} \mathbf{r} (X u_1 u_2 \rightarrow & \forall_{u_1, u_2} (X u_1 u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee X u'_1 u'_2) \wedge \\ & u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \rightarrow \\ & u_1 \approx_{\mathbb{S}} u_2). \end{aligned}$$

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

$$\begin{aligned} & {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u \mathbf{r} \forall_{u_1, u_2} (Xu_1u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge \\ & \quad u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \rightarrow \\ & \quad u_1 \approx_{\mathbb{S}} u_2). \end{aligned}$$

Pick f such that

$$\begin{aligned} & f \mathbf{r} \forall_{u_1, u_2} (Xu_1u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge \\ & \quad u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \end{aligned}$$

i.e.,

$$\begin{aligned} & \forall_{u_1, u_2, u} (X^{\mathbf{r}}u_1u_2u \rightarrow fu \mathbf{r} \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge \\ & \quad u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)). \end{aligned}$$

Our goal is

$$\approx_{\mathbb{S}}^{\mathbf{r}}(u_1, u_2, {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}uf).$$

To this end we use the greatest-fixed-point axiom for $\approx_{\mathbb{S}}^{\mathbf{r}}$ in the form

$$\begin{aligned} & \forall_{u_1, u_2, u} (Qu_1u_2u \rightarrow \\ & \quad \forall_{u_1, u_2, u} (Qu_1u_2u \rightarrow \\ & \quad \quad \exists_{s_1, s_2, s, u'_1, u'_2, u'} (\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s) \wedge (\approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u') \vee Qu_1u_2u') \wedge \\ & \quad \quad \quad u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2 \wedge u \equiv s :: u')) \rightarrow \\ & \quad \approx_{\mathbb{S}}^{\mathbf{r}}(u_1, u_2, u)) \end{aligned}$$

with

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

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

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

We need to show

$$\begin{aligned} & \exists_{s_1, s_2, s, u'_1, u'_2, u'} (\\ (31) \quad & \sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s) \wedge (\approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u') \vee \exists_{u''} (X^{\mathbf{r}}u_1u_2u'' \wedge u \equiv {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u''f)) \wedge \\ & \quad u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2 \wedge u \equiv s :: u'). \end{aligned}$$

First note that $\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s)$ is equivalent to $s_1 \equiv s_2 \equiv s$. Since $X^{\mathbf{r}}u_1u_2u'$ we know

$$fu' \mathbf{r} \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2).$$

Then $fu' \equiv \langle s, w \rangle$ with $\sim_{\text{sd}}^{\mathbf{r}}(s_1, s_2, s)$ and $w \mathbf{r} (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2)$, 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

$$\exists_{u'} (\approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u') \wedge w \equiv \text{InL}(u')) \vee \exists_{u''} (X^{\mathbf{r}}u'_1u'_2u'' \wedge w \equiv \text{InR}(u'')).$$

We distinguish cases on this disjunction. Recall

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

Case L. $\approx_{\mathbb{S}}^{\mathbf{r}}(u'_1, u'_2, u') \wedge w \equiv \text{InL}(u')$ for some u' . Then (31) holds, since $u \equiv {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u'f \equiv s :: u'$.

Case R. $X^{\mathbf{r}}u'_1u'_2u'' \wedge w \equiv \text{InR}(u'')$ for some u'' . Then again (31) holds with $u' := {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u''f$, since $u \equiv {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u'f \equiv s :: {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}u''f \equiv s :: u'$. $\square$

LEMMA D.0.5 (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

$$\begin{aligned} & (\lambda_z z) \mathbf{r} \forall_{\vec{x}}(I\vec{x} \rightarrow \forall_{\vec{y}}(\vec{A} \rightarrow X\vec{t}) \rightarrow X\vec{x}) \\ & \forall_{\vec{x}}(I\vec{x} \rightarrow (\lambda_z z) \mathbf{r} (\forall_{\vec{y}}(\vec{A} \rightarrow X\vec{t}) \rightarrow X\vec{x})) \\ & \forall_{\vec{x}}(I\vec{x} \rightarrow \forall_z(z \mathbf{r} \forall_{\vec{y}}(\vec{A} \rightarrow X\vec{t}) \rightarrow z \mathbf{r} X\vec{x})) \\ & \forall_{\vec{x}}(I\vec{x} \rightarrow \forall_z(\forall_{\vec{y}}(\vec{A} \rightarrow z \mathbf{r} X\vec{t}) \rightarrow z \mathbf{r} X\vec{x})) \end{aligned}$$

which is an instance of the same elimination axiom. $\square$

APPENDIX E

Extensionality of extracted terms

We want to show that

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

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

$$\begin{aligned} (\sim_{\mathbb{L}})_0^+ &: [] \sim_{\mathbb{L}} [], \\ (\sim_{\mathbb{L}})_1^+ &: \forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\mathbb{L}} \ell_2 \rightarrow s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2) \end{aligned}$$

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

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

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

LEMMA E.0.1. *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 $\text{sd} \rightarrow \mathbb{L} \rightarrow \mathbb{L}$ of $\sim_{\mathbb{L}}$ ’s second clause, which by definition of $\doteq$ means

$$\forall_{s_1, s_2, \ell_1, \ell_2} (s_1 \sim_{\text{sd}} s_2 \rightarrow \ell_1 \sim_{\mathbb{L}} \ell_2 \rightarrow s_1 :: \ell_1 \sim_{\mathbb{L}} s_2 :: \ell_2).$$

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

LEMMA E.0.2. *$\mathcal{R}_{\mathbb{L}}^\alpha$ is extensional w.r.t. the least-fixed-point axiom $\sim_{\mathbb{L}}^-$.*

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

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

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

$$\mathbb{L} \rightarrow \alpha \rightarrow (\text{sd} \rightarrow \mathbb{L} \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha.$$

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

$$\begin{aligned} & \forall_{x_1, x_2, f_1, f_2, \ell_1, \ell_2} (x_1 \doteq_\alpha x_2 \rightarrow f_1 \doteq_{\text{sd} \rightarrow \mathbb{L} \rightarrow \alpha \rightarrow \alpha} f_2 \rightarrow \ell_1 \sim_{\mathbb{L}} \ell_2 \rightarrow \\ & \quad \mathcal{R}_{\mathbb{L}}^\alpha \ell_1 x_1 f_1 \doteq_\alpha \mathcal{R}_{\mathbb{L}}^\alpha \ell_2 x_2 f_2). \end{aligned}$$

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

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

Case []. By the computation rules for $\mathcal{R}_{\mathbb{L}}^\alpha$ the claim $X[]$ follows from $x_1 \doteq_\alpha x_2$.

Case ::. Assume $s_1 \sim_{\text{sd}} s_2$ and $f_1 \doteq_{\text{sd} \rightarrow \mathbb{L} \rightarrow \alpha \rightarrow \alpha} f_2$. Let $y_1 := \mathcal{R}_{\mathbb{L}}^\alpha \ell_1 x_1 f_1$ and $y_2 := \mathcal{R}_{\mathbb{L}}^\alpha \ell_2 x_2 f_2$. Then $y_1 \doteq_\alpha y_2$ by assumption. The goal $f_1 s_1 \ell_1 y_1 \doteq_\alpha f_2 s_2 \ell_2 y_2$ follows by definition of $\doteq$ from $f_1 \doteq f_2$, $s_1 \sim s_2$, $\ell_1 \sim \ell_2$ and $y_1 \doteq_\alpha y_2$. $\square$

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

$$\begin{aligned} \approx_{\mathbb{S}}^- : \forall_{u_1, u_2} (u_1 \approx_{\mathbb{S}} u_2 \rightarrow \\ \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge u'_1 \approx_{\mathbb{S}} u'_2 \wedge u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \end{aligned}$$

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

$$\begin{aligned} \forall_{u_1, u_2} (X u_1 u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee X u'_1 u'_2) \wedge \\ u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \rightarrow \end{aligned}$$

$$X \subseteq \approx_{\mathbb{S}}.$$

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

LEMMA E.0.3. *The destructor $\mathcal{D}_{\mathbb{S}}$ is extensional w.r.t. the closure axiom $\approx_{\mathbb{S}}^-$.*

PROOF. The closure axiom $\approx_{\mathbb{S}}^-$ has cotype ${}^{\text{co}}\mathbb{S} \rightarrow \text{sd} \times {}^{\text{co}}\mathbb{S}$. The goal is $\mathcal{D}_{\mathbb{S}} \doteq_{({}^{\text{co}}\mathbb{S} \rightarrow \text{sd} \times {}^{\text{co}}\mathbb{S})} \mathcal{D}_{\mathbb{S}}$, which unfolds into

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

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

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

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

LEMMA E.0.4. ${}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}$ is extensional w.r.t. the greatest-fixed-point axiom $\approx_{\mathbb{S}}^+$.

PROOF. We equivalently rewrite $\approx_{\mathbb{S}}^+$ as

$$\begin{aligned} \forall_{u_1, u_2} (Xu_1u_2 \rightarrow \\ \forall_{u_1, u_2} (Xu_1u_2 \rightarrow \exists_{s_1, s_2, u'_1, u'_2} (s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee Xu'_1u'_2) \wedge \\ u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2)) \rightarrow \\ u_1 \approx_{\mathbb{S}} u_2) \end{aligned}$$

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

$$\alpha \rightarrow (\alpha \rightarrow \text{sd} \times ({}^{\text{co}}\mathbb{S} + \alpha)) \rightarrow {}^{\text{co}}\mathbb{S}.$$

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

$$\forall_{x_1, x_2} (x_1 \dot{=}_{\alpha} x_2 \rightarrow \forall_{f_1, f_2} (f_1 \dot{=}_{\alpha \rightarrow \text{sd} \times ({}^{\text{co}}\mathbb{S} + \alpha)} f_2 \rightarrow {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}x_1f_1 \approx_{\mathbb{S}} {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha}x_2f_2)).$$

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

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

This means that we have to show

$$\begin{aligned} \exists_{s_1, s_2, u'_1, u'_2} (\\ s_1 \sim_{\text{sd}} s_2 \wedge (u'_1 \approx_{\mathbb{S}} u'_2 \vee \exists_{y_1, y_2} (u'_1 \equiv {}^{\text{co}}\mathcal{R}y_1f_1 \wedge u'_2 \equiv {}^{\text{co}}\mathcal{R}y_2f_2 \wedge y_1 \dot{=}_{\alpha} y_2)) \wedge \\ u_1 \equiv s_1 :: u'_1 \wedge u_2 \equiv s_2 :: u'_2). \end{aligned}$$

From $x_1 \dot{=}_{\alpha} x_2$ and $f_1 \dot{=}_{\alpha \rightarrow \text{sd} \times ({}^{\text{co}}\mathbb{S} + \alpha)} f_2$ we obtain $f_1x_1 \sim_{\text{sd} \times ({}^{\text{co}}\mathbb{S} + \alpha)} f_2x_2$. By definition of $\sim_{\times}$ this implies the existence of s_1, s_2, a_1, a_2 with

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

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

$$\begin{aligned} (a_1 \equiv \text{InL}(u'_1) \wedge a_2 \equiv \text{InL}(u'_2) \wedge u'_1 \approx_{\mathbb{S}} u'_2) \vee \\ (a_1 \equiv \text{InR}(x'_1) \wedge a_2 \equiv \text{InR}(x'_2) \wedge x'_1 \dot{=}_{\alpha} x'_2). \end{aligned}$$

We argue by cases on this disjunction. Recall

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

Case L. Then we have s_1, s_2, u'_1, u'_2 with $s_1 \sim_{\text{sd}} s_2$ and $u'_1 \approx_{\mathbb{S}} u'_2$ such that $f_i x_i \equiv \langle s_i, \text{InL}(u'_i) \rangle$. Hence $u_i := {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha} x_i f_i \equiv s_i :: u'_i$, and the claim follows.

Case R. Then we have s_1, s_2, x'_1, x'_2 with $s_1 \sim_{\text{sd}} s_2$ and $x'_1 \doteq_{\alpha} x'_2$ such that $f_i x_i \equiv \langle s_i, \text{InR}(x'_i) \rangle$. Hence $u_i := {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha} x_i f_i \equiv s_i :: u'_i$ with $u'_i := {}^{\text{co}}\mathcal{R}_{\mathbb{S}}^{\alpha} x'_i f_i$, and again the claim follows. $\square$

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 E.0.5 (Compatibility of extracted terms). *Let $M: A$ be a proof of a c.r. formula A and $u_i: C_i$ ($i = 1, \dots, n$) all free c.r. assumptions whose associated object variable z_{u_i} is free in $\text{et}(M)$. Then we can find a proof of*

$$\text{et}(M)(z'_{u_1}, \dots, z'_{u_n}) \doteq_A \text{et}(M)(z''_{u_1}, \dots, z''_{u_n})$$

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

PROOF. By induction on M . *Case $u: C$.* Immediate. *Case $c: A$ an axiom.* This is clear in case the extracted term is the identity. For the axioms $I^{\pm}$ and ${}^{\text{co}}I^{\pm}$ it was proved in Lemmas E.0.1, E.0.2, E.0.3 and E.0.4.

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

$$\forall_{z'_u, z''_u} (z'_u \doteq_A z''_u \rightarrow \text{et}(M)(z'_u) \doteq_B \text{et}(M)(z''_u)).$$

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

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

$$\forall_{z'_u, z''_u} (z'_u \doteq_A z''_u \rightarrow \text{et}(M)z'_u \doteq_B \text{et}(M)z''_u).$$

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

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

COROLLARY E.0.6 (Extensionality of extracted terms). *Let $M: A$ be a proof of a c.r. formula A and $u_i: C_i$ ($i = 1, \dots, n$) all free c.r. assumptions*

whose associated object variable z_{u_i} is free in $\text{et}(M)$. Then we can find a proof of $\text{et}(M) \doteq_A \text{et}(M)$ from assumptions $z_{u_i} \doteq_{C_i} z_{u_i}$ for $i = 1, \dots, n$.

PROOF. In the proof constructed in Theorem E.0.5 we only need to substitute z'_{u_i}, z''_{u_i} by z_{u_i} . $\square$

Bibliography

- H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Massachusetts, second edition, 1996.
- S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.
- U. Berger and M. 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.
- H. B. Curry. Grundlagen der kombinatorischen Logik. *Amer. J. Math.*, 52: 509–536, 789–834, 1930.
- Y. L. Ershov. Model *C* of partial continuous functionals. In R. Gandy and M. Hyland, editors, *Logic Colloquium 1976*, pages 455–467. North-Holland, Amsterdam, 1977.
- R. Gandy. *On axiomatic systems in mathematics and theories in physics*. PhD thesis, University of Cambridge, 1953.
- R. Gandy. On the axiom of extensionality – part I. *The Journal of Symbolic Logic*, 21(1):36–48, 1956.
- G. Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
- K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.
- A. Heyting, editor. *Constructivity in Mathematics*, 1959. North-Holland, Amsterdam.
- D. Hilbert. Über das Unendliche. *Mathematische Annalen*, 95:161–190, 1925.
- W. A. Howard. The formulae-as-types notion of construction. In J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.
- I. Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 4:119–136, 1937.
- A. N. Kolmogorov. Zur Deutung der intuitionistischen Logik. *Math. Zeitschr.*, 35:58–65, 1932.

- K. G. Larsen and G. Winskel. Using information systems to solve recursive domain equations. *Information and Computation*, 91:232–258, 1991.
- A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.
- P. Martin-Löf. Infinite terms and a system of natural deduction. *Compositio Mathematica*, 24(1):93–103, 1972.
- Minlog. The Minlog Proof Assistant. <http://minlog-system.de/>.
- G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.
- J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.
- H. Schwichtenberg. Recursion on the partial continuous functionals. In C. Dimitracopoulos, L. Newelski, D. Normann, and J. Steel, editors, *Logic Colloquium '05*, volume 28 of *Lecture Notes in Logic*, pages 173–201. Association for Symbolic Logic, 2006.
- H. Schwichtenberg and S. S. Wainer. *Proofs and Computations*. Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.
- D. Scott. Domains for denotational semantics. In E. Nielsen and E. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *LNCS*, pages 577–613. Springer Verlag, Berlin, Heidelberg, New York, 1982.
- G. Takeuti. On a generalized logic calculus. *Japanese Journal of Mathematics*, 23:39–96, 1953.
- A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, second edition, 2000.
- A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction*, volume 121, 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1988.

Index

- $\perp$, 2
- $\tilde{\wedge}$, 4
- acis, 26, 77
 - flat, 78
 - of a closed type τ , 84
- algebra, 21, 83
 - associated with $(I/^{\text{co}}I)(\vec{\rho}, \vec{P})$, 38
 - non-recursive, 50
- algebra form, 20, 82
 - explicit, 20, 82
 - non-recursive, 20, 82
- application, 15, 18, 80
- approximable map, 16, 79
- ApproxSplit, 69
- arity, 36
- binary tree, 20, 83
- bisimilarity, 36, 47, 50
 - axiom, 48
 - relative, 49
- boolean, 20, 83
- case-construct, 31
- $\mathcal{C}$ -operator, 30
- Cauchy modulus, 63
- Cauchy sequence, 63
- clause, 36
 - nullary, 36
- closure
 - transitive, 38
- closure axiom, 44
- compatibility
 - of extracted terms, 108
 - of terms, 52
- composition, 85
- comprehension term, 37
- computation rule, 27
- conjunction
 - decoration of, 41
 - inductive definition, 40
- consistent, 14
- constructor, 82
 - as continuous function, 25
- constructor pattern, 27
- constructor type, 20, 82
- conversion, 10, 34, 95
 - D -, 27
- corecursion, 31
 - operator, 32
- cototal, 23
- cototality
 - relative, 37
- cotype, 48
 - of a formula, 50
- Curry, 9
- Curry-Howard correspondence, 9
- decoration
 - of conjunction, 41
 - of disjunction, 40
 - of existence, 41
- derivable, 3
 - classically, 5
 - intuitionistically, 4
- derivation, 20, 83
 - $\mathbf{r}$ -free, 55
 - n.c. part, 41
 - term, 9
- destructor, 31, 82
- disjunction
 - decoration of, 40
 - inductive definition, 39

- drinker formula, 7
- duplication, 30
- E-rule, 2
- EfEqD, 46
- elimination axiom, 41
- entailment, 13
- equality, 38
 - decidable, 28
 - Leibniz, 39, 46
 - of formulas, 35
 - of terms, 35
 - pointwise, 51
 - w.r.t. a cotype, 51
- ex-falso-quodlibet, 2, 4, 46
- existence
 - decoration of, 41
 - inductive definition, 41
- extensionality, 19, 81
 - of extracted terms, 108
 - of terms, 53
 - w.r.t. a cotype, 51
- falsity $\mathbf{F}$, 46
- falsum $\perp$, 2
- finite support principle, 17, 79
- formal neighborhood, 14
- formula, 37
 - $\mathbf{r}$ -free, 55
 - computationally relevant (c.r.), 37
 - non-computational (n.c.), 37
- function
 - continuous, 18, 80
 - monotone, 17, 79
 - strict, 25
- functional
 - computable, 22
 - partial continuous, 22
- Gandy, 48
- Gentzen, 1, 2
- greatest-fixed-point axiom, 44
- Harrop
 - formula, 37
 - predicate, 37
- height
 - of a cotype, 51
 - of syntactic expressions, 22, 84
- Howard, 9
- I-rule, 2
- ideal, 14, 77
 - cototal, 22, 23
 - total, 22, 23
- idempotent, 85
- identity, 20, 83
- information system, 13
 - atomic, 26
 - coherent, 26
 - flat, 14
 - of a closed type τ , 21
- intersection
 - inductive definition, 40
- intuitionistic logic, 2
- invariance axiom, 56
- Johansson, 2
- Kolmogorov, 2
- Larsen, 13, 77
- least-fixed-point axiom, 41
- Leibniz equality, 39, 45, 46
- LeIsNotGt, 69
- level
 - of a type, 21, 83
- list, 20, 83
- Martin-Löf, 9
- mgu, 85
- minimal logic, 3
- modus ponens, 2
- normal form, 11
- normalization, 9
- normalizing
 - strongly, 12
- number, 20, 83
 - binary, 20, 83
 - even, 38
 - positive, 20, 83
 - unary, 20, 83
- one-clause-nc, 45
- parameter
 - argument type, 20, 82
- predecessor, 30
- predicate, 37
 - $\mathbf{r}$ -free, 55
 - coinductive, 37

- computationally relevant (c.r.), 37
 - form, 36
 - inductive, 37
 - non-computational (n.c.), 37
 - one-clause-nc, 45, 103
- premise
 - major, 2
 - minor, 2
 - parameter, 36
 - recursive, 36
- product, 20, 83
- real
 - nonnegative, 65
 - positive, 65
- RealApprox, 68
- RealBound, 64
- RealEqChar, 64
- RealNNegChar, 65
- RealNNegCompat, 65
- RealPosChar, 66
- RealPosCompat, 66
- RealPosPlus, 69
- reals
 - equal, 64
 - equivalent, 64
- recursion
 - operator, 29
- recursive
 - argument type, 20, 82
- redex
 - β , 9
- reduction
 - one-step, 11
- reduction sequence, 11
- reduction tree, 11
- relation
 - logical, 51
- Scott, 13, 77, 79
- sequence
 - reduction, 11
- set of tokens
 - deductively closed, 14, 77
- similarity, 36, 47, 50
 - relative, 49
- soundness theorem
 - for realizability, 61
- stability, 4
- stream, 20, 83
- strongly normalizing, 12
- substitution, 85
 - more general, 85
 - sharp, 37
- sum, 20, 83
- Takeuti, 48
- TCF, 41
- term
 - extracted, 60
 - of T^+ , 27
- token, 13
 - extended, 21, 84
- total, 23
- totality
 - relative, 37
 - w.r.t. a cotype, 51
- transitive closure, 42
- tree
 - binary, 20, 83
 - reduction, 11
- type, 21, 83
 - base, 21, 83
 - closed, 21, 83
 - higher, 21, 83
 - level of, 21, 83
 - of a formula, 38
 - of a predicate, 38
- unifier
 - most general, 85
- union
 - inductive definition, 39
- unit, 20, 83
- uysum, 20, 83
- variable condition, 2
- Winskel, 13, 77
- ysumu, 20, 83