Logic II

Helmut Schwichtenberg

Mathematisches Institut der LMU, Sommersemester 2016

Contents

Preface v

Introduction vii

Chapter 1. Logic 1
1.1. Natural deduction 1
1.2. Embedding intuitionistic and classical logic 9

Chapter 2. Computability 17
2.1. Abstract computability via information systems 18
2.2. A term language for computable functionals 31
2.3. Denotational semantics 38

Chapter 3. A theory of computable functionals 39
3.1. Predicates and formulas 39
3.2. Axioms 42
3.3. Totality and induction 45
3.4. Coinductive definitions 47

Chapter 4. Computational content of proofs 49
4.1. Decoration 51
4.2. Realizers 57
4.3. Soundness 65

Chapter 5. Computational content of classical proofs 71
5.1. A-translation 72
5.2. Definite and goal formulas 73
5.3. Extraction from weak existence proofs 77
5.4. Applications 79

Chapter 6. Decorating proofs 83
6.1. Decoration algorithm 83
6.2. Applications 87

Appendix A. Denotational semantics: proofs 97
A.1. Unification 97

A.2. Ideals as denotations of terms 99
A.3. Preservation of values 105

Bibliography 109

Index 111

Preface

This is a script for the course "Logic II" at LMU, Sommersemester 2016. It is mainly based on Schwichtenberg and Wainer (2012). However, some concepts not in the textbook are included, most notably invariance under realizability.

I would like to thank the students of the course for their useful comments, Franziskus Wiesnet for carefully correcting the accompanying exercises and in particular Kenji Miyamoto for taking care of the exercises and for many helpful discussions and suggestions.

München, 13. Juli 2016
Helmut Schwichtenberg

Introduction

In this introduction we deal with the basics of formalizing proofs and, via the Curry-Howard correspondence, analysing their computational structure.

Minimal logic is a system of rules for deriving logical formulas based just on the two symbols $\to$ (implication) and $\forall$ (for all). Each symbol has two rules: an introduction rule ($\to^+$, $\forall^+$) and an elimination rule ($\to^-$, $\forall^-$). The rules for implication are

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

and the rules for universal quantification are

$$\frac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A}\,\forall^+ x \qquad \frac{\begin{array}{cc}\mid M & \\ \forall_x A(x) & r\end{array}}{A(r)}\,\forall^-$$

These are Gentzen's (1935) Natural Deduction rules (and there are others for $\exists$, $\vee$ and $\wedge$ which we shall come to later). Gentzen's idea was that natural deduction rules do indeed reflect the ways in which we construct logical arguments.

Notice that subderivations of the premises of rules are labelled M, N. In order to avoid obvious invalid derivations (for example $Px \to \forall_x Px$), the rule $\forall^+ x$ with conclusion $\forall_x A$ is subject to the following *(eigen-)variable condition*: the derivation M of the premise A should not contain any open (undischarged) assumptions having x as a free variable.

It is clear that derivations need to be started off somewhere, so in addition we need to introduce assumptions and – as in the implication introduction – allow some assumptions to be closed or discharged in the course of a derivation. The notation for a discharged assumption A is $[A]$.

Here is a simple example. Assume A, B are formulas and $x \notin \mathrm{FV}(A)$, the set of variables free in A.

$$\dfrac{\dfrac{\dfrac{[\forall_x(A \to B)] \qquad x}{A \to B}\,\forall^- \qquad A}{B}\,{\to}^-}{\dfrac{\forall_x B}{A \to \forall_x B}\,\forall^+x}\,{\to}^+}{\forall_x(A \to B) \to A \to \forall_x B}\,{\to}^+$$

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

It is possible that a derivation makes an unnecessary "detour" – an elimination immediately following an introduction – and we may want to remove it. This can be done for implication via

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \mid M\\ B\end{array}}{A \to B}\,{\to}^+ \qquad \begin{array}{c}\mid N\\ A\end{array}}{B}\,{\to}^- \qquad \text{reduces to} \qquad \begin{array}{c}\mid N\\ A\\ \mid M\\ B\end{array}$$

and for universal quantification via

$$\dfrac{\dfrac{\begin{array}{c}\mid M(x)\\ A(x)\end{array}}{\forall_x A(x)}\,\forall^+x \qquad r}{A(r)}\,\forall^- \qquad \text{reduces to} \qquad \begin{array}{c}\mid M(r)\\ A(r)\end{array}$$

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

$$(\lambda_u M(u))N \quad \text{reduces to} \quad M(N).$$

Similarly the unnecessary detour via $\forall$ will be represented as

$$(\lambda_x M(x))r \quad \text{reduces to} \quad M(r).$$

These reductions are instances of what, in lambda calculus terms, is known as *beta reduction* and we write

$$\begin{aligned}(\lambda_u M(u))N &\quad \mapsto_\beta \quad M(N),\\ (\lambda_x M(x))r &\quad \mapsto_\beta \quad M(r).\end{aligned}$$

The lambda calculus provides an abstract setting for representing and computing functions and beta reduction is the main computational mechanism. Now there comes one further detail: we want to be able to move back and forth from derivations to lambda representations of them and back again from lambda terms to derivations. For this reason it is necessary to assign to each lambda term a "type", which will be the formula whose proof it represents. The formula type will be written as a superscript. In detail then, the first beta reduction example above now becomes

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

In summary, the Curry-Howard correspondence is completely described by the Table 1 below.

Suppose that we have extended minimal logic with axioms introducing the existential quantifier:

$$\exists^+ \colon \forall_x(A \to \exists_x A), \qquad \exists^- \colon \exists_x A \to \forall_x(A \to B) \to B \quad (x \text{ not free in } B).$$

These now allow us to make computationally meaningful derivations. The underlying principle is this: suppose one has a derivation of a closed formula $\exists_x A(x)$ resulting from an existential introduction axiom $\exists^+$, i.e., the derivation (written as a Curry-Howard term) is of the form $\exists^+ r u^{A(r)}$. Then r (the computational content) is a witness for the existential quantifier, and it may be read off immediately. Of course the derivation may not end with an existential introduction. However, the process of normalization will beta-reduce the derivation term into one in which $\exists^+$ *is* the final operator to be applied. In general normalization is the process of computing out a lambda term, until no further beta reductions can be made. In other words, normalization reduces away all unnecessary detours.

Now suppose that the formula $\exists_x A(x)$ is not closed, say it has one free variable z. By instantiating z we obtain again a closed formula depending on the instantiated value. Extracting a witnessing term from a normalized derivation term, as above, then provides a witness depending on the instantiated value. However, to bring out the uniformity involved in this process requires a new method, *realizability*.

In the lecture course we will study such computational aspects of logic, both from theoretical and practical point of view. For the former, we will make heavy use of the textbook Schwichtenberg and Wainer (2012). For

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

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

the latter, we present many fundamental concepts and principles underlying our proof assistant Minlog[1]. For example, inductively defined data and predicates, recursion, induction and decoration. All will be developed theoretically as well as within the Minlog setting, and in each case a wide variety of practical case studies will illustrate program extraction.

[1] `www.minlog-system.de`

CHAPTER 1

Logic

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

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

We will also show that every derivation has a normal form, and analyze the shape of such normal derivations.

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

1.1. Natural deduction

Rules come in pairs: we have an introduction and an elimination rule for each of the logical connectives. The resulting system is called *minimal logic*;

it was introduced by Kolmogorov (1932), Gentzen (1935) and Johansson (1937). Notice that no negation is yet present. If we go on and require *ex-falso-quodlibet* for the nullary propositional symbol $\bot$ ("falsum") we can embed *intuitionistic logic* with negation as $A \to \bot$. To embed classical logic, we need to go further and add as an axiom schema the principle of *indirect proof*, also called *stability* ($\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x}$) for relation symbols R), but then it is appropriate to restrict to the language based on $\to$, $\forall$, $\bot$ and $\wedge$. The reason for this restriction is that we can neither prove $\neg\neg\exists_x A \to \exists_x A$ nor $\neg\neg(A \vee B) \to A \vee B$ (the former is Markov's scheme). However, we can prove them for the classical existential quantifier and disjunction defined by $\neg\forall_x\neg A$ and $\neg A \to \neg B \to \bot$. Thus we need to make a distinction between two kinds of "exists" and two kinds of "or": the classical ones are "weak" and the non-classical ones "strong" since they have constructive content. We mark the distinction by writing a tilde above the weak disjunction and existence symbols thus $\tilde{\vee}, \tilde{\exists}$.

We have alrady seen the rules of minimal logic for $\to$, $\forall$ and the corresponding Curry-Howard terms in Table 1.

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 $A_1, \dots, A_n \in \Gamma$.

1.1.1. Logic in Minlog: basic examples. As a first encounter with the Minlog[1] proof assistant we consider two simple logical facts. Let A, B, C be propositional variables.

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

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

For the second example involving quantifiers, let P be a unary predicate variable.

$$\forall_x(A \to Px) \to A \to \forall_x Px \quad (x \notin \mathrm{FV}(A)).$$

[1] `www.minlog-system.de`

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

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

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

For the second example we obtain

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

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

Let us now formalize these proofs in Minlog. We describe the interactions rather shortly; for a more thorough introduction the reader should consult the Minlog tutorial. After starting the system by typing

```
(load "~/git/minlog/init.scm")
```

we declare three propositional variables by executing

```
(add-pvar-name "A" "B" "C" (make-arity))
```

The proof then is generated by the following sequence of commands:

```
(set-goal "(A -> B -> C) -> (A -> B) -> A -> C")
(assume "u" "v" "w")
(use "u")
(use "w")
(use "v")
(use "w")
```

We save the proof and display it as lambda-expression, with formulas assigned to the assumption variables:

```
(define proof (current-proof))
(proof-to-expr-with-formulas proof)
```

The result is

```
u: A -> B -> C
v: A -> B
w: A

(lambda (u) (lambda (v) (lambda (w) ((u w) (v w)))))
```

For the second example involving quantifiers we proceed similarly. We declare x as a variable of type α (a type variable) and P as a unary predicate variable

```
(add-var-name "x" (py "alpha"))
(add-pvar-name "P" (make-arity (py "alpha")))
```

The proof is generated by the following sequence of commands:

```
(set-goal "all x(A -> P x) -> A -> all x P x")
(assume "u" "v" "x")
(use "u")
(use "v")
```

We again save the proof and display it as lambda-expression, with formulas assigned to the assumption variables:

```
(define proof (current-proof))
(proof-to-expr-with-formulas proof)
```

The result is

```
u: all x(A -> P x)
v: A

(lambda (u) (lambda (v) (lambda (x) ((u x) v))))
```

These lambda-expressions are exactly what the Curry-Howard correspondence gives us.

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

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

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

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

This proof can be generated by the following sequence of commands.

```
(set-goal "(((A -> bot) -> bot) -> bot) -> A -> bot")
(assume "u" "v")
(use "u")
(assume "w")
(use "w")
(use "v")
```

We can display it by executing

```
(define proof (current-proof))
(proof-to-expr-with-formulas proof)
```

The result then is

```
u: ((A -> bot) -> bot) -> bot
v: A
w: A -> bot

(lambda (u) (lambda (v) (u (lambda (w) (w v)))))
```

Derivations for the following formulas are left as exercises.

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

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

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

For disjunction the introduction and elimination axioms are

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

For conjunction we have

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

and for the existential quantifier

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

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

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

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

For conjunction we have

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

and for the existential quantifier

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

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

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

LEMMA. *The following are derivable.*

$$(A \wedge B \to C) \leftrightarrow (A \to B \to C),$$
$$(A \to B \wedge C) \leftrightarrow (A \to B) \wedge (A \to C),$$
$$(A \vee B \to C) \leftrightarrow (A \to C) \wedge (B \to C),$$
$$(A \to B \vee C) \leftarrow (A \to B) \vee (A \to C),$$

$$(\forall_x A \to B) \leftarrow \exists_x(A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),$$
$$(A \to \forall_x B) \leftrightarrow \forall_x(A \to B) \quad \textit{if } x \notin \mathrm{FV}(A),$$

$$(\exists_x A \to B) \leftrightarrow \forall_x(A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),$$
$$(A \to \exists_x B) \leftarrow \exists_x(A \to B) \quad \textit{if } x \notin \mathrm{FV}(A).$$

PROOF. A derivation of the final formula is

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

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

The Minlog proof for the final formula uses the axioms $\exists^+$ and $\exists^-$:

```
(set-goal "ex x(A -> P x) -> A -> ex x P x")
(assume "u" "v")
(by-assume "u" "x" "w")
(ex-intro "x")
(use "w")
(use "v")
```

Again we can display it by executing

```
(define proof (current-proof))
(proof-to-expr-with-formulas proof)
```

The result then is

```
Ex-Elim: ex x(A -> P x) -> all x((A -> P x) -> ex x0 P x0) ->
                           ex x P x
Ex-Intro: all x(P x -> ex x0 P x0)
u: ex x(A -> P x)
v: A
w: A -> P x

(lambda (u)
  (lambda (v)
    ((Ex-Elim u)
      (lambda (x) (lambda (w) ((Ex-Intro x) (w v)))))))
```

As an example of how to deal with the axioms for conjunction and disjunction we give Minlog proofs for the two directions of

$$(A \vee B \to C) \leftrightarrow (A \to C) \wedge (B \to C).$$

Here `andd` and `ord` indicate that we view $\wedge$, $\vee$ as in**d**uctively defined[2].

```
(set-goal "(A ord B -> C) -> (A -> C) andd (B -> C)")
(assume "u")
(split)
(assume "v")
(use "u")
(intro 0)
(use "v")
(assume "w")
(use "u")
(intro 1)
(use "w")
```

We display the proof as above and obtain

```
Intro: (A -> C) -> (B -> C) -> (A -> C) andd (B -> C)
Intro: A -> A ord B
Intro: B -> A ord B
u: A ord B -> C
v: A
w: B

(lambda (u)
  ((Intro (lambda (v) (u (Intro v))))
    (lambda (w) (u (Intro w)))))
```

For the other direction we have

```
(set-goal "(A -> C) andd (B -> C) -> A ord B -> C")
(assume "u" "v")
(elim "v")
(use "u")
(use "u")
```

and obtain

```
Elim: A ord B -> (A -> C) -> (B -> C) -> C
Elim: (A -> C) andd (B -> C) ->
      ((A -> C) -> (B -> C) -> A -> C) -> A -> C
```

[2]For the existential quantifier we could have proceeded similarly, using the inductively defined `exd`. However, there is also a "primitive" `ex` in Minlog, which was used above. There is also a primitive conjunction written &.

```
Elim: (A -> C) andd (B -> C) ->
      ((A -> C) -> (B -> C) -> B -> C) -> B -> C
u: (A -> C) andd (B -> C)
v: A ord B
u0: A -> C
u1: B -> C

(lambda (u)
  (lambda (v)
    (((Elim v) ((Elim u) (lambda (u0) (lambda (u1) u0))))
      ((Elim u) (lambda (u0) (lambda (u1) u1))))))
```

1.2. Embedding intuitionistic and classical logic

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

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

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

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

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

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

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

Moreover let

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

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

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

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

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

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

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

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

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

THEOREM (Stability, or principle of indirect proof).
(a) $\vdash (\neg\neg A \to A) \to (\neg\neg B \to B) \to \neg\neg(A \wedge B) \to A \wedge B$.
(b) $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$.
(c) $\vdash (\neg\neg A \to A) \to \neg\neg\forall_x A \to A$.
(d) $\vdash_c \neg\neg A \to A$ *for every formula* A *without* $\vee, \exists$.

PROOF. (a) is left as an exercise.

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

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

(c)

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

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

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

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

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

LEMMA. *The following are derivable.*

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

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

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

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

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

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

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

PROOF. (1)

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

(2)

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

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

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

(4)

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

(5)

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

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

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

Using this derivation M we obtain

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

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

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

Corollary.

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

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

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

There is a similar lemma on weak disjunction:

Lemma. *The following are derivable.*

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

Proof. The derivation of the final formula is

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

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

Corollary.

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

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

$$\vdash A \to A \mathbin{\tilde{\vee}} B, \quad \vdash B \to A \mathbin{\tilde{\vee}} B,$$
$$\vdash_c A \mathbin{\tilde{\vee}} B \to (A \to C) \to (B \to C) \to C \quad (C \text{ without } \vee, \exists),$$
$$\vdash A \to \tilde{\exists}_x A,$$
$$\vdash_c \tilde{\exists}_x A \to \forall_x (A \to B) \to B \quad (x \notin \mathrm{FV}(B),\ B \text{ without } \vee, \exists).$$

The derivations of the second and the fourth formula are

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

and

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

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

is derivable in classical logic if and only if its translation A^g is derivable in minimal logic.

DEFINITION (Gödel-Gentzen translation A^g).

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

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

PROOF. Induction on A.

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

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

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

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

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

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

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

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

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

hypothesis and the remark at the end of 1.2.1. For example, in case $\exists^-$ the induction hypothesis gives

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

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

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

CHAPTER 2

Computability

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

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

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

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

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

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

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

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

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

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

The general theory of computability concerns partial functions and partial operations on them. However, we are primarily interested in total objects, so once the theory of partial objects is developed, we can look for ways to extract the total ones. We will define the total objects of base type inductively, and then by induction on types the notion of totality for arbitrary types.

2.1. Abstract computability via information systems

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

2.1.1. Information systems. The basic idea of information systems is to provide an axiomatic setting to describe approximations of abstract objects (like functions or functionals) by concrete, finite ones. We do not attempt to analyze the notion of "concreteness" or finiteness here, but rather take an arbitrary countable set A of "bits of data" or "tokens" as a basic notion to be explained axiomatically. In order to use such data to build approximations of abstract objects, we need a notion of "consistency", which determines when the elements of a finite set of tokens are consistent with each other. We also need an "entailment relation" between consistent sets U of data and single tokens a, which intuitively expresses the fact that the information contained in U is sufficient to compute the bit of information a.

The axioms below are a minor modification of Scott's (1982), due to Larsen and Winskel (1991).

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

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

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

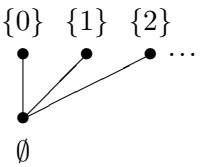
$$\begin{aligned}
&U \vdash V \quad \text{for} \quad U \in \mathrm{Con} \wedge \forall_{a \in V}(U \vdash a),\\
&a \uparrow b \quad \text{for} \quad \{a, b\} \in \mathrm{Con} \qquad (a, b \text{ are } \textit{consistent}),\\
&U \uparrow V \quad \text{for} \quad \forall_{a \in U, b \in V}(a \uparrow b).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

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

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

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

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



A rather important example is the following, which concerns approximations of functions from a countable set A into a countable set B. The tokens are the pairs (a, b) with $a \in A$ and $b \in B$, and

$$\mathrm{Con} := \{\, \{\, (a_i, b_i) \mid i < k \,\} \mid \forall_{i,j<k}(a_i = a_j \to b_i = b_j) \,\},$$

$$U \vdash (a,b) := (a,b) \in U.$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Since

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

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

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

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

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

For the final property, suppose

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

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

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

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

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

We write $r \colon \boldsymbol{A} \to \boldsymbol{B}$ to mean that r is an approximable map from $\boldsymbol{A}$ to $\boldsymbol{B}$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.3. Scott topology. We can also characterize approximable maps as continuous functions w.r.t. a certain topology on the space $|\boldsymbol{A}|$ of ideals of an information system $\boldsymbol{A}$.[1]

DEFINITION. Suppose $\boldsymbol{A} = (A, \mathrm{Con}, \vdash)$ is an information system and $U \in \mathrm{Con}$. Define $\mathcal{O}_U \subseteq |\boldsymbol{A}|$ by

$$\mathcal{O}_U := \{ x \in |\boldsymbol{A}| \mid U \subseteq x \}.$$

Note that, since the ideals $x \in |\boldsymbol{A}|$ are deductively closed, $x \in \mathcal{O}_U$ implies $\overline{U} \subseteq x$.

LEMMA. *The system of all $\mathcal{O}_U$ with $U \in \mathrm{Con}$ forms the basis of a topology on $|\boldsymbol{A}|$, called the Scott topology.*

[1]Here we refer to topology in the standard (non-constructive) sense, based on points. There are also recent attempts to build a constructive "point-free" topology; however, we will not enter this subject here.

PROOF. Suppose $U, V \in \mathrm{Con}$ and $x \in \mathcal{O}_U \cap \mathcal{O}_V$, i.e., $U \subseteq x$ and $V \subseteq x$. We need $W \in \mathrm{Con}$ such that $x \in \mathcal{O}_W \subseteq \mathcal{O}_U \cap \mathcal{O}_V$. Choose $W = U \cup V$. □

LEMMA. *Let $\boldsymbol{A}$ be an information system and $\mathcal{O} \subseteq |\boldsymbol{A}|$. Then the following are equivalent.*

(a) *$\mathcal{O}$ is open in the Scott topology.*
(b) *$\mathcal{O}$ satisfies*
  (i) *If $x \in \mathcal{O}$ and $x \subseteq y$, then $y \in \mathcal{O}$ (Alexandrov condition).*
  (ii) *If $x \in \mathcal{O}$, then $\overline{U} \in \mathcal{O}$ for some $U \subseteq x$ (Scott condition).*
(c) *$\mathcal{O} = \bigcup_{\overline{U} \in \mathcal{O}} \mathcal{O}_U$.*

Hence open sets $\mathcal{O}$ may be seen as those determined by a (possibly infinite) system of finitely observable properties, namely all U such that $\overline{U} \in \mathcal{O}$.

PROOF. (a) → (b). If $\mathcal{O}$ is open, then $\mathcal{O}$ is the union of some $\mathcal{O}_U$'s, $U \in \mathrm{Con}$. Since each $\mathcal{O}_U$ is upwards closed, also $\mathcal{O}$ is; this proves the Alexandrov condition. For the Scott condition assume $x \in \mathcal{O}$. Then $x \in \mathcal{O}_U \subseteq \mathcal{O}$ for some $U \in \mathrm{Con}$. Note that $\overline{U} \in \mathcal{O}_U$, hence $\overline{U} \in \mathcal{O}$, and $U \subseteq x$ since $x \in \mathcal{O}_U$.

(b) → (c). Assume that $\mathcal{O} \subseteq |\boldsymbol{A}|$ satisfies the Alexandrov and Scott conditions. Let $x \in \mathcal{O}$. By the Scott condition, $\overline{U} \in \mathcal{O}$ for some $U \subseteq x$, so $x \in \mathcal{O}_U$ for this U. Conversely, let $x \in \mathcal{O}_U$ for some $\overline{U} \in \mathcal{O}$. Then $\overline{U} \subseteq x$. Now $x \in \mathcal{O}$ follows from $\overline{U} \in \mathcal{O}$ by the Alexandrov condition.

(c) → (a). The $\mathcal{O}_U$'s are the basic open sets of the Scott topology. □

We now give some simple characterizations of the continuous functions $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$. Call f *monotone* if $x \subseteq y$ implies $f(x) \subseteq f(y)$.

LEMMA. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be information systems and $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$. Then the following are equivalent.*

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

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

Note that in (c) the set $\{\, f(x) \mid x \in D \,\}$ is directed by monotonicity of f; hence its union is indeed an ideal in $|\boldsymbol{A}|$. 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 $U \subseteq x$.

Hence continuous maps $f\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract

objects $x \in |\boldsymbol{A}|$ and $f(x) \in |\boldsymbol{B}|$: whenever we are given a finite approximation V to the value $f(x)$, then there is a finite approximation U to the argument x such that already $f(\overline{U})$ contains the information in V; note that by monotonicity $f(\overline{U}) \subseteq f(x)$.

PROOF. (a) $\to$ (b). Let f be continuous. Then for any basic open set $\mathcal{O}_V \subseteq |\boldsymbol{B}|$ (so $V \in \mathrm{Con}_B$) the set $f^{-1}[\mathcal{O}_V] = \{\, x \mid V \subseteq f(x) \,\}$ is open in $|\boldsymbol{A}|$. To prove monotonicity assume $x \subseteq y$; we must show $f(x) \subseteq f(y)$. So let $b \in f(x)$, i.e., $\{b\} \subseteq f(x)$. The open set $f^{-1}[\mathcal{O}_{\{b\}}] = \{\, z \mid \{b\} \subseteq f(z) \,\}$ satisfies the Alexandrov condition, so from $x \subseteq y$ we can infer $\{b\} \subseteq f(y)$, i.e., $b \in f(y)$. To prove PFS assume $b \in f(x)$. The open set $\{\, z \mid \{b\} \subseteq f(z) \,\}$ satisfies the Scott condition, so for some $U \subseteq x$ we have $\{b\} \subseteq f(\overline{U})$.

(b) $\to$ (a). Assume that f satisfies monotonicity and PFS. We must show that f is continuous, i.e., that for any fixed $V \in \mathrm{Con}_B$ the set $f^{-1}[\mathcal{O}_V] = \{\, x \mid V \subseteq f(x) \,\}$ is open. We prove

$$\{\, x \mid V \subseteq f(x) \,\} = \bigcup \{\, \mathcal{O}_U \mid U \in \mathrm{Con}_A \text{ and } V \subseteq f(\overline{U}) \,\}.$$

Let $V \subseteq f(x)$. Then by PFS $V \subseteq f(\overline{U})$ for some $U \in \mathrm{Con}_A$ such that $U \subseteq x$, and $U \subseteq x$ implies $x \in \mathcal{O}_U$. Conversely, let $x \in \mathcal{O}_U$ for some $U \in \mathrm{Con}_A$ such that $V \subseteq f(\overline{U})$. Then $\overline{U} \subseteq x$, hence $V \subseteq f(x)$ by monotonicity.

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

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

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

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

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

(a) *With any approximable map $r\colon \boldsymbol{A} \to \boldsymbol{B}$ we can associate a continuous function $|r|\colon |\boldsymbol{A}| \to |\boldsymbol{B}|$ by*

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

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

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

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

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

PROOF. Let r be an ideal of $\boldsymbol{A} \to \boldsymbol{B}$; then by the theorem just proved r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\boldsymbol{A}|$.

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

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

Continuity of $|r|$ follows immediately from part (b) of the lemma above, since by definition $|r|$ is monotone and satisfies PFS.

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

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

Finally, for any approximable map $r\colon \boldsymbol{A} \to \boldsymbol{B}$ we have

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

so $r = \widehat{|r|}$. □

Moreover, one can easily check that

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

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

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

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

2.1.4. Algebras and types. We now consider concrete information systems, our basis for continuous functionals.

Types will be built from base types by the formation of function types, $\rho \to \sigma$. As domains for the base types we choose non-flat and possibly infinitary free algebras, given by their constructors. The main reason for taking non-flat base domains is that we want the constructors to be injective and with disjoint ranges. This generally is not the case for flat domains.

Definition (Types). We inductively define *type forms*

$$\rho, \sigma ::= \alpha \mid \rho \to \sigma \mid \mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k}$$

with α, ξ type variables and $k \geq 1$. Note that $(\rho_\nu)_{\nu<n} \to \sigma$ means $\rho_0 \to \ldots \to \rho_{n-1} \to \sigma$, associated to the right. Let $\mathrm{TV}(\rho)$ denote the set of (free) type variables in ρ. We define $\mathrm{SP}(\alpha, \rho)$ "α occurs at most *strictly positive* in ρ" by induction on ρ.

$$\mathrm{SP}(\alpha, \beta) \qquad \frac{\alpha \notin \mathrm{TV}(\rho) \quad \mathrm{SP}(\alpha, \sigma)}{\mathrm{SP}(\alpha, \rho \to \sigma)} \qquad \frac{\mathrm{SP}(\alpha, \rho_{i\nu}) \text{ for all } i < k,\ \nu < n_i}{\mathrm{SP}(\alpha, \mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k})}$$

Now we can define $\mathrm{Ty}(\rho)$ "ρ is a *type*", again by induction on ρ.

$$\mathrm{Ty}(\alpha) \qquad \frac{\mathrm{Ty}(\rho) \quad \mathrm{Ty}(\sigma)}{\mathrm{Ty}(\rho \to \sigma)} \qquad \frac{\mathrm{Ty}(\rho_{i\nu}) \text{ and } \mathrm{SP}(\xi, \rho_{i\nu}) \text{ for all } i < k,\ \nu < n_i}{\mathrm{Ty}(\mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k})}$$

We call

$$\iota := \mu_\xi((\rho_{i\nu})_{\nu<n_i} \to \xi)_{i<k}$$

an *algebra*. Sometimes it is helpful to display the type parameters and write $\iota(\vec{\alpha}, \vec{\beta}\,)$, where $\vec{\alpha}, \vec{\beta}$ are all type variables except ξ free in some $\rho_{i\nu}$, and $\vec{\alpha}$ are the ones occuring only strictly positive. If we write the i-th component of ι in the form $(\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi$, then we call

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

the i-th *constructor type* of ι.

In $(\rho_{i\nu}(\xi))_{\nu<n_i} \to \xi$ we call $\rho_{i\nu}(\xi)$ a *parameter* argument type if ξ does not occur in it, and a *recursive* argument type otherwise. A recursive argument type $\rho_{i\nu}(\xi)$ is *nested* if it has an occurrence of ξ in a strictly positive parameter position of another (previously defined) algebra, and unnested otherwise. An algebra ι is called *nested* if it has a constructor with at least one nested recursive argument type, and *unnested* otherwise.

Examples.

$$\begin{aligned}
\mathbf{U} &:= \mu_\xi \xi \qquad \text{(unit)},\\
\mathbf{B} &:= \mu_\xi(\xi, \xi) \qquad \text{(booleans)},\\
\mathbf{N} &:= \mu_\xi(\xi, \xi \to \xi) \qquad \text{(natural numbers, unary)},
\end{aligned}$$

$$\begin{aligned}
\mathbf{P} &:= \mu_\xi(\xi, \xi \to \xi, \xi \to \xi) \qquad \text{(positive numbers, binary)},\\
\mathbf{D} &:= \mu_\xi(\xi, \xi \to \xi \to \xi) \qquad \text{(binary trees, or derivations)},\\
\mathbf{O} &:= \mu_\xi(\xi, \xi \to \xi, (\mathbf{N} \to \xi) \to \xi) \qquad \text{(ordinals)},\\
\mathbf{T}_0 &:= \mathbf{N}, \quad \mathbf{T}_{n+1} := \mu_\xi(\xi, (\mathbf{T}_n \to \xi) \to \xi) \qquad \text{(trees)}.
\end{aligned}$$

Examples of algebras strictly positive in their type parameters are

$$\begin{aligned}
\mathbf{I}(\alpha) &:= \mu_\xi(\alpha \to \xi) \qquad \text{(identity)},\\
\mathbf{L}(\alpha) &:= \mu_\xi(\xi, \alpha \to \xi \to \xi) \qquad \text{(lists)},\\
\alpha \times \beta &:= \mu_\xi(\alpha \to \beta \to \xi) \qquad \text{(product)},\\
\alpha + \beta &:= \mu_\xi(\alpha \to \xi, \beta \to \xi) \qquad \text{(sum)}.
\end{aligned}$$

An example of a nested algebra is

$$\mathbf{T} := \mu_\xi(\mathbf{L}(\xi) \to \xi) \quad \text{(finitely branching trees)}.$$

Note that $\mathbf{T}$ has a total inhabitant since $\mathbf{L}(\alpha)$ has one (given by the [] constructor).

Let ρ be a type; we write $\rho(\vec{\alpha}\,)$ for ρ to indicate its dependence on the type parameters $\vec{\alpha}$. We can substitute types $\vec{\sigma}$ for $\vec{\alpha}$, to obtain $\rho(\vec{\sigma}\,)$. Examples are $\mathbf{L}(\mathbf{B})$, the type of lists of booleans, and $\mathbf{N} \times \mathbf{N}$, the type of pairs of natural numbers.

Note that often there are many equivalent ways to define a particular type. For instance, we could take $\mathbf{U}+\mathbf{U}$ to be the type of booleans, $\mathbf{L}(\mathbf{U})$ to be the type of natural numbers, and $\mathbf{L}(\mathbf{B})$ to be the type of positive binary numbers.

For every constructor type of an algebra we provide a (typed) *constructor symbol* C_i. In some cases they have standard names, for instance

$\mathtt{tt}^{\mathbf{B}}, \mathtt{ff}^{\mathbf{B}}$ for the two constructors of the type $\mathbf{B}$ of booleans,

$0^{\mathbf{N}}, S^{\mathbf{N}\to\mathbf{N}}$ for the type $\mathbf{N}$ of (unary) natural numbers,

$1^{\mathbf{P}}, S_0^{\mathbf{P}\to\mathbf{P}}, S_1^{\mathbf{P}\to\mathbf{P}}$ for the type $\mathbf{P}$ of (binary) positive numbers,

$0^{\mathbf{O}}, S^{\mathbf{O}\to\mathbf{O}}, \mathrm{Sup}^{(\mathbf{N}\to\mathbf{O})\to\mathbf{O}}$ for the type $\mathbf{O}$ of ordinals,

$[]^{\mathbf{L}(\rho)}, \mathrm{cons}^{\rho\to\mathbf{L}(\rho)\to\mathbf{L}(\rho)}$ for the type $\mathbf{L}(\rho)$ of lists,

$(\mathrm{Inl}_{\rho\sigma})^{\rho\to\rho+\sigma}, (\mathrm{Inr}_{\rho\sigma})^{\sigma\to\rho+\sigma}$ for the sum type $\rho+\sigma$,

$\mathrm{Branch}\colon \mathbf{L}(\mathbf{T}) \to \mathbf{T}$ for the type $\mathbf{T}$ of finitely branching trees.

An algebra form ι is *structure-finitary* if all its argument types $\rho_{i\nu}$ are not of arrow form. It is *finitary* if in addition it has no type variables. In the examples above $\mathbf{U}$, $\mathbf{B}$, $\mathbf{N}$, $\mathbf{P}$ and $\mathbf{D}$ are all finitary, but $\mathbf{O}$ and $\mathbf{T}_{n+1}$

are not. $\mathbf{L}(\rho)$, $\rho \times \sigma$ and $\rho + \sigma$ are structure-finitary, and finitary if their parameter types are. The nested algebra $\mathbf{T}$ above is finitary.

An algebra is *explicit* if all its constructor types have parameter argument types only (i.e., no recursive argument types). In the examples above $\mathbf{U}$, $\mathbf{B}$, $\rho \times \sigma$ and $\rho + \sigma$ are explicit, but $\mathbf{N}$, $\mathbf{P}$, $\mathbf{L}(\rho)$, $\mathbf{D}$, $\mathbf{O}$, $\mathbf{T}_{n+1}$ and $\mathbf{T}$ are not. We will also need the notion of the *level* of a type, which is defined by

$$\mathrm{lev}(\iota) := 0, \qquad \mathrm{lev}(\rho \to \sigma) := \max\{\mathrm{lev}(\sigma), 1 + \mathrm{lev}(\rho)\}.$$

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

2.1.5. Partial continuous functionals. For every type ρ we define the information system $\boldsymbol{C}_\rho = (C_\rho, \mathrm{Con}_\rho, \vdash_\rho)$. The ideals $x \in |\boldsymbol{C}_\rho|$ are the *partial continuous functionals* of type ρ. Since we will have $\boldsymbol{C}_{\rho\to\sigma} = \boldsymbol{C}_\rho \to \boldsymbol{C}_\sigma$, the partial continuous functionals of type $\rho \to \sigma$ will correspond to the continuous functions from $|\boldsymbol{C}_\rho|$ to $|\boldsymbol{C}_\sigma|$ w.r.t. the Scott topology. It will not be possible to define $\boldsymbol{C}_\rho$ by recursion on the type ρ, since we allow algebras with constructors having function arguments (like $\mathbf{O}$ and Sup). Instead, we shall use recursion on the "height" of the notions involved, defined below.

Definition (Information system of type ρ). We simultaneously define C_ι, $C_{\rho\to\sigma}$, Con_ι and $\mathrm{Con}_{\rho\to\sigma}$.

(a) The *tokens* $a \in C_\iota$ are the type correct constructor expressions $\mathrm{C}a_1^* \dots a_n^*$ where a_i^* is an *extended token*, i.e., a token or the special symbol $*$ which carries no information.
(b) The tokens in $C_{\rho\to\sigma}$ are the pairs (U, b) with $U \in \mathrm{Con}_\rho$ and $b \in C_\sigma$.
(c) A finite set U of tokens in C_ι is *consistent* (i.e., $\in \mathrm{Con}_\iota$) if all its elements start with the same constructor C, say of arity $\tau_1 \to \dots \to \tau_n \to \iota$, and all $U_i \in \mathrm{Con}_{\tau_i}$ for $i = 1, \dots, n$, where U_i consists of all (proper) tokens at the i-th argument position of some token in $U = \{\mathrm{C}\vec{a_1^*}, \dots, \mathrm{C}\vec{a_m^*}\}$.
(d) $\{\, (U_i, b_i) \mid i \in I \,\} \in \mathrm{Con}_{\rho\to\sigma}$ is defined to mean $\forall_{J\subseteq I}(\bigcup_{j\in J} U_j \in \mathrm{Con}_\rho \to \{\, b_j \mid j \in J \,\} \in \mathrm{Con}_\sigma)$.

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

(e) $\{\mathrm{C}\vec{a_1^*}, \dots, \mathrm{C}\vec{a_m^*}\} \vdash_\iota \mathrm{C}'\vec{a^*}$ is defined to mean $\mathrm{C} = \mathrm{C}'$, $m \geq 1$ and $U_i \vdash a_i^*$, with U_i as in (c) above (and $U \vdash *$ taken to be true).
(f) $W \vdash_{\rho\to\sigma} (U, b)$ is defined to mean $WU \vdash_\sigma b$, where application WU of $W = \{\, (U_i, b_i) \mid i \in I \,\} \in \mathrm{Con}_{\rho\to\sigma}$ to $U \in \mathrm{Con}_\rho$ is defined to be $\{\, b_i \mid U \vdash_\rho U_i \,\}$; recall that $U \vdash V$ abbreviates $\forall_{a\in V}(U \vdash a)$.

If we define the *height* of the syntactic expressions involved by

$$|\mathrm{C}a_1^* \dots a_n^*| := 1 + \max\{\, |a_i^*| \mid i = 1, \dots, n \,\}, \qquad |*| := 0,$$
$$|(U, b)| := \max\{1 + |U|, 1 + |b|\},$$

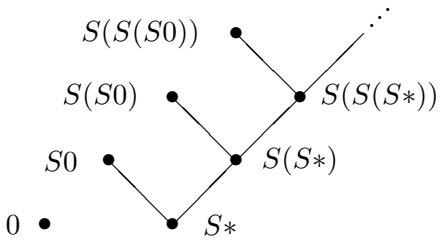


FIGURE 1. Tokens and entailment for **N**

$$|\{\, a_i \mid i \in I \,\}| := \max\{\, 1 + |a_i| \mid i \in I \,\},$$
$$|U \vdash a| := \max\{1 + |U|, 1 + |a|\},$$

these are definitions by recursion on the height.

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

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

Notice that we have $\boldsymbol{C}_{\rho\to\sigma} = \boldsymbol{C}_\rho \to \boldsymbol{C}_\sigma$, as defined generally for information systems.

For example, the tokens for the algebra **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 algebra **D** of derivations with a nullary constructor 0 and a binary C. Then $\{\mathrm{C}0*, \mathrm{C}*0\}$ is consistent, and $\{\mathrm{C}0*, \mathrm{C}*0\} \vdash \mathrm{C}00$.

2.1.6. Constructors as continuous functions. Let ι be an algebra. Every constructor C generates the following ideal in the function space:

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

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

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

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

An immediate consequence is that the (continuous maps corresponding to) constructors are injective and their ranges are disjoint, which is what we wanted to achieve by associating non-flat rather than flat information systems with algebras.

LEMMA (Constructors are injective and have disjoint ranges). *Let ι be an algebra and* C *be a constructor of ι. Then*

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

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

PROOF. Immediate from the definitions. □

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

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

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

2.1.7. Total and cototal ideals in a finitary algebra. In the information system $\boldsymbol{C}_\iota$ associated with an algebra ι, the "total" and "cototal" ideals are of special interest. Here we give an explicit definition for finitary algebras. For general algebras totality can be defined inductively and cototality coinductively.

Recall that a token in ι is a constructor tree P possibly containing the special symbol $*$. Because of the possibility of parameter arguments we need to distinguish between "structure-" and "fully" total and cototal ideals. For the definition it is easiest to refer to a constructor tree $P(*)$ with a distinguished occurrence of $*$. This occurrence is called *non-parametric* if the path from it to the root does not pass through a parameter argument of a constructor. For a constructor tree $P(*)$, an arbitrary $P(\mathrm{C}\vec{a^*})$ is called *one-step extension* of $P(*)$, written $P(\mathrm{C}\vec{a^*}) \succ_1 P(*)$.

DEFINITION. Let ι be an algebra, and $\boldsymbol{C}_\iota$ its associated information system. An ideal $x \in |\boldsymbol{C}_\iota|$ is *cototal* if every constructor tree $P(*) \in x$ has a $\succ_1$-predecessor $P(\mathrm{C}\vec{*}) \in x$; it is called *total* if it is cototal and the relation $\succ_1$ on x is well-founded. It is called *structure-cototal* (*structure-total*) if the same holds with $\succ_1$ defined w.r.t. $P(*)$ with a non-parametric distinguished occurrence of $*$.

If there are no parameter arguments, we shall simply speak of total and cototal ideals. For example, for the algebra $\mathbf{N}$ every total ideal is the deductive closure of a token $S(S\ldots(S0)\ldots)$, and the set of all tokens $S(S\ldots(S*)\ldots)$ is a cototal ideal. For the algebra $\mathbf{L}(\mathbf{N})$ of lists of natural

numbers the total ideals are the finite lists and the cototal ones the finite or infinite lists. For the algebra $\mathbf{D}$ of derivations the total ideals can be viewed as the finite derivations, and the cototal ones as the finite or infinite "locally correct" derivations of Mints (1978); arbitrary ideals can be viewed as "partial" or "incomplete" derivations, with "holes".

Call a partial continuous functional *total* if it maps total arguments into total values, and *cototal* if it maps cototal arguments into cototal values.

2.2. A term language for computable functionals

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

Recall that a partial continuous functional is defined to be computable if it is the limit of a recursively enumerable set of finite approximations. We introduce a convenient way to define computable functionals, by means of defining equations or more precisely, computation rules. Therefore we extend the term language by constants D defined by certain "computation rules", as in (Berger et al., 2003; Berger, 2005). The resulting term system can be seen as a common extension of Gödel's T (1958) and Plotkin's PCF; we call it T^+.

2.2.1. Structural recursion operators and Gödel's T. We begin with a discussion of particularly important examples of such constants D, the (structural) higher type *recursion operators* $\mathcal{R}^\tau_\iota$ introduced by Hilbert (1925) and Gödel (1958). They are used to construct maps from the algebra ι to τ, by recursion on the structure of ι. For instance, $\mathcal{R}^\tau_{\mathbf{N}}$ has type $\mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$. 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, $\mathcal{R}^{\mathbf{N}}_{\mathbf{N}} nm\lambda_{n,p}(Sp)$ defines addition $m+n$ by recursion on n. For $\lambda_{n,p}(Sp)$ we often write $\lambda_{_,p}(Sp)$ since the bound variable n is not used.

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

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

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

For some algebras we spell out the type of their recursion operators:

$\mathcal{R}^\tau_{\mathbf{B}}\colon \mathbf{B} \to \tau \to \tau \to \tau$,
$\mathcal{R}^\tau_{\mathbf{N}}\colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\mathbf{P}}\colon \mathbf{P} \to \tau \to (\mathbf{P} \to \tau \to \tau) \to (\mathbf{P} \to \tau \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\mathbf{D}}\colon \mathbf{D} \to \tau \to (\mathbf{D} \to \tau \to \mathbf{D} \to \tau \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\mathbf{O}}\colon \mathbf{O} \to \tau \to (\mathbf{O} \to \tau \to \tau) \to ((\mathbf{N} \to \mathbf{O}) \to (\mathbf{N} \to \tau) \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\mathbf{L}(\rho)}\colon \mathbf{L}(\rho) \to \tau \to (\rho \to \mathbf{L}(\rho) \to \tau \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\rho+\sigma}\colon \rho + \sigma \to (\rho \to \tau) \to (\sigma \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\rho\times\sigma}\colon \rho \times \sigma \to (\rho \to \sigma \to \tau) \to \tau$,
$\mathcal{R}^\tau_{\mathbf{T}}\colon \mathbf{T} \to (\mathbf{L}(\mathbf{T} \times \tau) \to \tau) \to \tau$.

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

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

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

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

It can be used to define the *predecessor* function on $\mathbf{N}$, i.e., $\mathrm{P}0 := 0$ and $\mathrm{P}(Sn) := n$, by the term

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

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 written in the form

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

We shall also need *map operators*. Let $\rho(\vec\alpha\,)$ be a type and $\vec\alpha$ strictly positive type parameters. We define

$$\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\rho(\vec\alpha\,)}\colon \rho(\vec\sigma\,) \to (\vec\sigma \to \vec\tau\,) \to \rho(\vec\tau\,)$$

(where $(\vec\sigma \to \vec\tau) \to \rho(\vec\tau\,)$ means $(\sigma_1 \to \tau_1) \to \ldots \to (\sigma_n \to \tau_n) \to \rho(\vec\tau\,)$). If none of $\vec\alpha$ appears free in $\rho(\vec\alpha\,)$ let

$$\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\rho(\vec\alpha\,)} x\vec f := x.$$

Otherwise we use an outer recursion on $\rho(\vec\alpha\,)$ and if $\rho(\vec\alpha\,)$ is $\iota(\vec\alpha\,)$ an inner one on x. In case $\rho(\vec\alpha\,)$ is $\iota(\vec\alpha\,)$ we abbreviate $\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\iota(\vec\alpha\,)}$ by $\mathcal{M}^{\vec\sigma\to\vec\tau}_{\iota}$ or $\mathcal{M}^{\vec\tau}_{\iota(\vec\sigma\,)}$.

The immediate cases for the outer recursion are

$$\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\alpha_i} x\vec f := f_i x, \qquad \mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}(\sigma\to\rho)} h\vec f x := \mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\rho}(hx)\vec f.$$

It remains to consider $\iota(\vec\pi(\vec\alpha\,))$. In case $\vec\pi(\vec\alpha\,)$ is not $\vec\alpha$ let

$$\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\iota(\vec\pi(\vec\alpha\,))} x\vec f := \mathcal{M}^{\vec\pi(\vec\sigma\,)\to\vec\pi(\vec\tau\,)}_{\iota} x(\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\pi_i(\vec\alpha\,)} \cdot \vec f\,)_{i<|\vec\pi\,|}$$

with $\mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\pi_i(\vec\alpha\,)} \cdot \vec f := \lambda_x \mathcal{M}^{\vec\sigma\to\vec\tau}_{\lambda_{\vec\alpha}\pi_i(\vec\alpha\,)} x\vec f$. In case $\vec\pi(\vec\alpha\,)$ is $\vec\alpha$ we use recursion on x and define for a constructor $\mathrm{C}_i\colon (\rho_{i\nu}(\vec\sigma, \iota(\vec\sigma\,)))_{\nu<n_i} \to \iota(\vec\sigma\,)$

$$\mathcal{M}^{\vec\sigma\to\vec\tau}_{\iota}(\mathrm{C}_i\vec x\,)\vec f$$

to be the result of applying C'_i of type $(\rho_{i\nu}(\vec\tau, \iota(\vec\tau\,)))_{\nu<n_i} \to \iota(\vec\tau\,)$ (the same constructor as C_i with only the type changed) to, for each $\nu < n_i$,

$$\mathcal{M}^{\vec\sigma,\iota(\vec\sigma\,)\to\vec\tau,\iota(\vec\tau\,)}_{\lambda_{\vec\alpha,\beta}\rho_{i\nu}(\vec\alpha,\beta)} x_{i\nu}\vec f(\mathcal{M}^{\vec\sigma\to\vec\tau}_{\iota} \cdot \vec f\,).$$

Note that the final function argument provides the recursive call w.r.t. the recursion on x.

Example. We write $x :: l$ as shorthand for $\mathrm{cons}(x, l)$.

$$\begin{aligned}
&\mathcal{M}^{\tau}_{\mathbf{L}(\sigma)}[]f^{\sigma\to\tau} := [],\\
&\mathcal{M}^{\tau}_{\mathbf{L}(\sigma)}(x^\sigma :: l^{\mathbf{L}(\sigma)})f^{\sigma\to\tau} := (fx) :: (\mathcal{M}\, l\, f).
\end{aligned}$$

Definition. *Terms of Gödel's* T for nested algebras are inductively defined from typed variables x^ρ and constants for constructors C^ι_i, recursion operators $\mathcal{R}^\tau_\iota$, cases operators $\mathcal{C}^\tau_\iota$ and map operators $\mathcal{M}^{\vec\rho\to\vec\tau}_{\lambda_{\vec\alpha}\pi}$ by abstraction $\lambda_{x^\rho} M^\sigma$ and application $M^{\rho\to\sigma} N^\rho$.

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

$$\begin{aligned}
\mathrm{FV}(x) &:= \{x\},\\
\mathrm{FV}(C) &:= \emptyset \quad \text{for } C \text{ constructor or a recursion, cases or map operator,}\\
\mathrm{FV}(\lambda_x M) &:= \mathrm{FV}(M) \setminus \{x\},\\
\mathrm{FV}(MN) &:= \mathrm{FV}(M) \cup \mathrm{FV}(N).
\end{aligned}$$

2.2.2. Conversion. We define a *conversion relation* $\mapsto_\rho$ between terms of type ρ by

$$(\lambda_x M(x))N \mapsto M(N), \tag{7}$$

$$\lambda_x(Mx) \mapsto M \quad \text{if } x \notin \mathrm{FV}(M) \ (M \text{ not an abstraction}), \tag{8}$$

$$\mathcal{R}^\tau_\iota(\mathrm{C}^\iota_i \vec{N})\vec{M} \mapsto M_i(\mathcal{M}^{\iota\to\iota\times\tau}_{\lambda_\alpha \rho_{i\nu}(\alpha)} N_{i\nu}\lambda_x\langle x^\iota, \mathcal{R}^\tau_\iota x\vec{M}\rangle)_{\nu<n_i} \tag{9}$$

where $(\rho_{i\nu}(\iota))_{\nu<n_i} \to \iota$ is the type of the i-th constructor C_i.

In the special case $\rho_{i\nu}(\alpha) = \alpha$ we can avoid the product type and instead of the pair

$$\mathcal{M}^{\iota\to\iota\times\tau}_{\lambda_\alpha\alpha} N_{i\nu}\lambda_x\langle x^\iota, \mathcal{R}^\tau_\iota x\vec{M}\rangle \quad \text{i.e.,} \quad \langle N^\iota_{i\nu}, \mathcal{R}^\tau_\iota N_{i\nu}\vec{M}\rangle$$

take its two components $N^\iota_{i\nu}$ and $\mathcal{R}^\tau_\iota N_{i\nu}\vec{M}$ as separate arguments of M_i.

The rule (7) is called β-conversion, and (8) η-conversion; their left hand sides are called *β-redexes* or *η-redexes*, respectively. The left hand side of (9) is called *$\mathcal{R}$-redex*; it is a special case of a redex associated with a constant D defined by "computation rules" (cf. 2.2.4), and hence also called a *D-redex*.

We give some examples of what can be defined in Gödel's T. The *projections* of a pair to its components can be defined by

$$M0 := \mathcal{R}^\rho_{\rho\times\sigma} M^{\rho\times\sigma}(\lambda_{x^\rho,y^\sigma} x^\rho), \quad M1 := \mathcal{R}^\sigma_{\rho\times\sigma} M^{\rho\times\sigma}(\lambda_{x^\rho,y^\sigma} y^\sigma).$$

The *append* function $*$ for lists satisfies the equations

$$[] * l_2 := l_2, \qquad (x :: l_1) * l_2 := x :: (l_1 * l_2).$$

It can be defined as the term

$$l_1 * l_2 := \mathcal{R}^{\mathbf{L}(\alpha)}_{\mathbf{L}(\alpha)} l_1 l_2 \lambda_{x,_,p}(x :: p).$$

Here "_" is taken for a bound variable which is not used. Using the append function $*$ we can define *list reversal* Rev by

$$\mathrm{Rev}([]) := [], \qquad \mathrm{Rev}(x :: l) := \mathrm{Rev}(l) * (x :: []).$$

The corresponding term is

$$\mathrm{Rev}(l) := \mathcal{R}^{\mathbf{L}(\alpha)}_{\mathbf{L}(\alpha)} l\ []\lambda_{x,_,p}(p * (x :: [])).$$

Assume we want to define by simultaneous recursion two functions on $\mathbf{N}$, say $\mathrm{even}, \mathrm{odd}\colon \mathbf{N} \to \mathbf{B}$ satisfying

$$\mathrm{even}(0) := \mathtt{tt}, \qquad \mathrm{odd}(0) := \mathtt{ff},$$
$$\mathrm{even}(Sn) := \mathrm{odd}(n), \qquad \mathrm{odd}(Sn) := \mathrm{even}(n).$$

This can be achieved using pair types: we recursively define the single function $\mathrm{evenodd}\colon \mathbf{N} \to \mathbf{B} \times \mathbf{B}$ by $\mathrm{evenodd}\, m := \mathcal{R}^{\mathbf{B}\times\mathbf{B}}_{\mathbf{N}} m\langle \mathtt{tt}, \mathtt{ff}\rangle \lambda_{n,p}\langle p1, p0\rangle$.

General recursion with respect to a measure. In practice it often happens that one needs to recur to an argument which is not an immediate component of the present constructor object; this is not allowed in structural recursion. Of course, in order to ensure that the recursion terminates we have to assume that the recurrence is w.r.t. a given well-founded set; for simplicity we restrict ourselves to the algebra **N**. However, we do allow that the recurrence is with respect to a measure function μ, with values in **N**. The operator $\mathcal{F}$ of *general recursion* then is defined by

$$\mathcal{F}\mu x G = Gx(\lambda_y[\textbf{if } \mu y < \mu x \textbf{ then } \mathcal{F}\mu y G \textbf{ else } \varepsilon]), \tag{10}$$

where ε denotes a canonical inhabitant of the range. We leave it as an exercise to prove that $\mathcal{F}$ is definable from an appropriate structural recursion operator.

2.2.3. Corecursion. One can show that an arbitrary "reduction sequence" beginning with a term in Gödel's T terminates. For this to hold it is essential that the constants allowed in T are restricted to constructors C and recursion, cases and map operators $\mathcal{R}$, $\mathcal{C}$, $\mathcal{M}$. A consequence is that every closed term of a base type denotes a total ideal. The conversion rules for $\mathcal{R}$ (cf. 2.2.2) work from the leaves towards the root, and terminate because total ideals are well-founded. If, however, we deal with cototal ideals (infinitary derivations, for example), 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 $\iota = \mu_\xi(\kappa_0, \ldots, \kappa_{k-1})$:

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

The product of these k constructor types is isomorphic to

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

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

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

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

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

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

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

We list the types of the corecursion operators for some algebras:

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

The conversion relation for each of these is defined below. For $f \colon \rho \to \tau$ and $g \colon \sigma \to \tau$ we denote $\lambda_x(\mathcal{R}^\tau_{\rho+\sigma} x f g)$ of type $\rho + \sigma \to \tau$ by $[f, g]$, and similary for ternary sumtypes etcetera. x_0, x_1 are shorthand for the two projections of x of type $\rho \times \sigma$. The identity functions id below are of type $\iota \to \iota$ with ι the respective algebra.

$$\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{B}} NM &\mapsto [\lambda_\mathtt{tt}, \lambda_\mathtt{ff}](MN),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{N}} NM &\mapsto [\lambda_0, \lambda_x(S([\mathrm{id}^{\mathbf{N}\to\mathbf{N}}, \lambda_y({}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{N}} y M)]x))](MN),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{P}} NM &\mapsto [\lambda_1, \lambda_x(S_0([\mathrm{id}, P_{\mathbf{P}}]x)), \lambda_x(S_1([\mathrm{id}, P_{\mathbf{P}}]x))](MN),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{D}} NM &\mapsto [\lambda_0, \lambda_x(\mathrm{C}([\mathrm{id}, P_{\mathbf{D}}]x_0)([\mathrm{id}, P_{\mathbf{D}}]x_1))](MN),\\
{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{L}(\rho)} NM &\mapsto [\lambda_[], \lambda_x(x_0 :: [\mathrm{id}, \lambda_y({}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbf{L}(\rho)} y M)]x_1)](MN),
\end{aligned}$$

with $P_\alpha := \lambda_y({}^{\mathrm{co}}\mathcal{R}^\tau_\alpha y M)$ for $\alpha \in \{\mathbf{P}, \mathbf{D}\}$.

2.2.4. A common extension T^+ of Gödel's T and Plotkin's PCF. *Terms* of T^+ are built from (typed) variables and (typed) constants (constructors C or defined constants D, see below) by (type-correct) application and abstraction:

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

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

$$D\vec{P}_i(\vec{y}_i) = M_i \qquad (i = 1, \dots, n) \tag{11}$$

with free variables of $\vec{P}_i(\vec{y}_i)$ and M_i among $\vec{y}_i$, where the arguments on the left hand side must be "constructor patterns", i.e., lists of applicative terms built from constructors and distinct variables. To ensure consistency of the defining equations, we require that for $i \neq j$ $\vec{P}_i$ and $\vec{P}_j$ have disjoint free variables, and either $\vec{P}_i$ and $\vec{P}_j$ are non-unifiable (i.e., there is no substitution

which identifies them), or else for the most general unifier ϑ of $\vec{P}_i$ and $\vec{P}_j$ we have $M_i\vartheta = M_j\vartheta$. Notice that the substitution ϑ assigns to the variables $\vec{y}_i$ in M_i constructor patterns $\vec{R}_k(\vec{z})$ $(k = i, j)$. A further requirement on a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ is that the lengths of all $\vec{P}_i(\vec{y}_i)$ are the same; this number is called the *arity* of D, denoted by $\mathrm{ar}(D)$. A substitution instance of a left hand side of (11) is called a *D-redex*.

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

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

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 map and (structural) recursion operators can be viewed as defined by computation rules, which in this case are called *conversion* rules; cf. 2.2.2.

The boolean connectives andb, impb and orb are defined by

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

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

Decidable *equality* $=_\iota : \iota \to \iota \to \mathbf{B}$ for a finitary algebra ι can be defined easily by computation rules. For example,

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

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

$$(0 =_{\mathbf{D}} 0) = \mathtt{tt}, \qquad (\mathrm{C}ab =_{\mathbf{D}} 0) = \mathtt{ff},$$
$$(0 =_{\mathbf{D}} \mathrm{C}ab) = \mathtt{ff}, \qquad (\mathrm{C}ab =_{\mathbf{D}} \mathrm{C}a'b') = (a =_{\mathbf{D}} a' \text{ andb } b =_{\mathbf{D}} b').$$

The *predecessor* functions introduced in 2.2.1 by means of the cases-operator $\mathcal{C}$ can also be viewed as defined constants, for instance

$$\mathrm{P}0 = 0, \qquad \mathrm{P}(Sn) = n.$$

2.3. Denotational semantics

How can we use computation rules to define an ideal z in a function space? The general idea is to inductively define the set of tokens (U, a) that make up z. It is convenient to define the value $[\![\lambda_{\vec{x}} M]\!]$, where M is a term with free variables among $\vec{x}$. Since this value is a token set, we can define inductively the relation $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$.

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

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

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

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

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

For every constructor C and defined constant D we have

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

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

This "denotational semantics" has good properties; however, we do not carry out the proofs here (cf. Appendix A or Schwichtenberg and Wainer (2012)). First of all, one can prove that $[\![\lambda_{\vec{x}} M]\!]$ is an ideal . Moreover, our definition above of the denotation of a term is reasonable in the sense that it is not changed by an application of the standard (β- and η-) conversions or a computation rule.

CHAPTER 3

A theory 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 concept is that of an inductively defined predicate.

3.1. Predicates and formulas

To properly introduce inductively defined predicates we first have to define what predicates and formulas are, and also the concept of strictly positive occurrences of predicate variables.

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

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

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

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

The reason for writing it in this way is that it fits more conveniently with the logical elimination rules, which will be useful in the proof of the soundness theorem. It expresses that every "competitor" $\{\, n \mid A(n) \,\}$ satisfying the same clauses contains $T_{\mathbf{N}}$. This is the usual induction schema for natural numbers, which clearly only holds for "total" numbers (i.e., total ideals in the information system for $\mathbf{N}$). Notice that we have used a "strengthened" form of the "step formula", namely $\forall_n(T_{\mathbf{N}}n \to A(n) \to A(Sn))$ rather than $\forall_n(A(n) \to A(Sn))$. In applications of the least-fixed-point axiom this simplifies the proof of the "induction step", since we have the additional hypothesis $T(n)$ available. Totality for an arbitrary algebra can be defined similarly.

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

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

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

Definition (Predicates and formulas). By simultaneous induction we define *predicate forms*

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

and *formula forms*

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

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

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

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

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

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

$$\frac{\mathrm{F}(A_{i\nu}) \text{ and } \mathrm{SP}(X, A_{i\nu}) \text{ for all } i{<}k,\ \nu{<}n_i \quad \mathrm{P}(\vec{Q}, \vec{R}\,)}{\mathrm{P}(I(\vec{\rho}, \vec{Q}, \vec{R}\,))}$$

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

with

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

where $\vec{Y}, \vec{Z}$ are all predicate variables free in some $A_{i\nu}$ except X, and $\vec{Y}$ are the ones occuring only strictly positive. We call I an *inductively defined predicate* or shortly *inductive predicate*.

Here $\vec{A} \to B$ means $A_0 \to \cdots \to A_{n-1} \to B$, associated to the right. The terms $\vec{r}$ are those introduced in 2.2.4, i.e., typed terms built from variables and constants by abstraction and application, and (importantly) those with a common reduct are identified. In $\forall_{\vec{x}}((A_\nu(X))_{\nu<n} \to X\vec{r})$ we call $A_\nu(X)$ a *parameter* premise if X does not occur in it, and a *recursive* premise otherwise. A recursive premise $A_\nu(X)$ is *nested* if it has an occurrence of X in a strictly positive parameter position of another (previously defined) inductive predicate, and unnested otherwise. An inductive predicate I is called *nested* if it has a clause with at least one nested recursive premise, and *unnested* otherwise.

A predicate of the form $\{\,\vec{x} \mid C\,\}$ is called a *comprehension term*. We identify $\{\,\vec{x} \mid C(\vec{x}\,)\,\}\vec{r}$ with $C(\vec{r}\,)$. For a predicate C of arity $(\rho, \vec{\sigma}\,)$ we write Cr for $\{\,\vec{y} \mid Cr\vec{y}\,\}$. An inductive predicate is *finitary* if its clauses have recursive premises of the form $X\vec{s}$ only.

Remark (Substitution for predicate parameters). Let $C(\vec{X})$ be a predicate or formula and $\vec{P}$ be predicates of the same arities as $\vec{X}$. By induction on C one can see easily that $C(\vec{P})$ is a predicate or formula again.

Examples. The *even numbers* are inductively defined by

$$\mathrm{Even} := \mu_X(X0, \forall_n(Xn \to X(S(Sn)))).$$

Let $\prec$ be a binary relation. Its *transitive closure* is inductively defined by

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

An important example of an inductive predicate is (Leibniz) equality. But a word of warning is in order here: we need to distinguish four separate but closely related equalities.

(i) Firstly, defined function constants D are introduced by computation rules, written $l = r$, but intended as left-to-right rewrites.
(ii) Secondly, we have Leibniz equality $=^{\mathrm{d}}$ inductively defined below.
(iii) Thirdly, pointwise equality between partial continuous functionals will be defined inductively as well.
(iv) Fourthly, if l and r have a finitary algebra as their type, $l = r$ can be read as a boolean term, where $=$ is the decidable equality defined in 2.2.4 as a boolean-valued binary function.

We define *Leibniz equality* by

$$\mathrm{EqD} := \mu_X(\forall_x Xxx).$$

Existence, intersection and union can be defined inductively by

$$\begin{aligned} \mathrm{Ex}_Y &:= \mu_X(\forall_x(Yx \to X)), \\ \mathrm{Cap}_{Y,Z} &:= \mu_X(\forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X\vec{x}\,)), \end{aligned}$$

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

We will use the abbreviations

$$(x =^{\mathrm{d}} y) := \mathrm{EqD}(x,y), \qquad P \cap Q := \mathrm{Cap}_{P,Q},$$
$$\exists_x A := \mathrm{Ex}_{\{x|A\}}, \qquad P \cup Q := \mathrm{Cup}_{P,Q}.$$

3.2. Axioms

We define a theory of computable functionals, called TCF. Formulas are those in F defined above, involving typed variables. Derivations use the rules of minimal logic for $\to$ and $\forall$, and the following axioms. For each inductive predicate, there are "closure" or introduction axioms, together with a "least-fixed-point" or elimination axiom. In more detail, consider an inductive predicate

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

For every $i < k$ we have a *clause* (or *introduction axiom*)

$$I_i^+ : \forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to I\vec{r}_i). \tag{12}$$

Moreover, we have an *elimination axiom*

$$I^- : \forall_{\vec{x}}(I\vec{x} \to (\forall_{\vec{x}_i}((A_{i\nu}(I \cap X))_{\nu<n_i} \to X\vec{r}_i))_{i<k} \to X\vec{x}) \tag{13}$$

($I \cap X$ was inductively defined above). Here X can be thought of as a "competitor" predicate. We take all substitution instances of I_i^+, I^- (w.r.t. substitutions for type and predicate variables) as axioms.

Examples. (i) For the even numbers defined by

$$\mathrm{Even} := \mu_X(X0, \forall_n(Xn \to X(S(Sn))))$$

the introduction axioms are

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

and the elimination axiom is

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

(ii) The transitive closure $\mathrm{TC}_\prec$ of a binary relation $\prec$ was defined inductively by

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

The introduction axioms are

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

and the elimination axiom is

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

(iii) Leibniz equality was defined above by

$$\mathrm{EqD} := \mu_X(\forall_x Xxx).$$

The introduction axiom is

$$\forall_x(x^\rho =^{\mathrm{d}} x^\rho)$$

and the elimination axiom

$$\forall_{x,y}(x =^{\mathrm{d}} y \to \forall_x Xxx \to Xxy),$$

where $x =^{\mathrm{d}} y$ abbreviates $\mathrm{EqD}(\rho)(x^\rho, y^\rho)$.

Lemma (Compatibility of EqD). $\forall_{x,y}(x =^{\mathrm{d}} y \to A(x) \to A(y))$.

Proof. Exercise. □

Using compatibility of EqD one easily proves symmetry and transitivity. Define *falsity* by $\mathbf{F} := (\mathtt{ff} =^{\mathrm{d}} \mathtt{tt})$.

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

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

The claim can now be proved by induction on $A \in \mathrm{F}$. *Case $I\vec{s}$.* If I has no nullary clause take Efq_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 $s_i =^{\mathrm{d}} t_i$, by the remark above. Hence $I\vec{s}$ by compatibility. The cases $Y\vec{s}$, $A \to B$ and $\forall_x A$ are obvious. □

A crucial use of the equality predicate EqD is that it allows us to lift a boolean term $r^{\mathbf{B}}$ to a formula, using $\mathrm{atom}(r^{\mathbf{B}}) := (r^{\mathbf{B}} =^{\mathrm{d}} \mathtt{tt})$. This opens up a convenient way to deal with equality on finitary algebras. The computation rules ensure that, for instance, the boolean term $Sr =_{\mathbf{N}} Ss$, or more precisely $=_{\mathbf{N}}(Sr, Ss)$, is identified with $r =_{\mathbf{N}} s$. We can now turn this boolean term into the formula $(Sr =_{\mathbf{N}} Ss) =^{\mathrm{d}} \mathtt{tt}$, which again is abbreviated

by $Sr =_{\mathbf{N}} Ss$, but this time with the understanding that it is a formula. Then (importantly) the two formulas $Sr =_{\mathbf{N}} Ss$ and $r =_{\mathbf{N}} s$ are identified because the latter is a reduct of the first. Consequently there is no need to prove the implication $Sr =_{\mathbf{N}} Ss \to r =_{\mathbf{N}} s$ explicitly.

Examples (continued). (iv) Let $\prec$ be a binary relation. Its *accessible part* is inductively defined by

$$\mathrm{Acc}_{\prec} := \mu_X(\forall_x(\forall_{y\prec x} Xy \to Xx)).$$

The introduction axiom is

$$\forall_x(\forall_{y\prec x}\mathrm{Acc}_{\prec}(y) \to \mathrm{Acc}_{\prec}(x)),$$

where $\forall_{y\prec x}A$ stands for $\forall_y(y \prec x \to A)$. The elimination axiom is

$$\forall_x(\mathrm{Acc}_{\prec}(x) \to \forall_x(\forall_{y\prec x}\mathrm{Acc}_{\prec}(y) \to \forall_{y\prec x} Py \to Xx) \to Xx).$$

(v) Existence, intersection and union were defined inductively by

$$\begin{aligned}
\mathrm{Ex}_Y &:= \mu_X(\forall_x(Yx \to X)),\\
\mathrm{Cap}_{Y,Z} &:= \mu_X(\forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X\vec{x})),\\
\mathrm{Cup}_{Y,Z} &:= \mu_X(\forall_{\vec{x}} Y\vec{x} \to X\vec{x},\ Z \to X\vec{x})
\end{aligned}$$

together with the abbreviations

$$\begin{aligned}
\exists_x A &:= \mathrm{Ex}_{\{x|A\}},\\
P \cap Q &:= \mathrm{Cap}_{P,Q},\\
P \cup Q &:= \mathrm{Cup}_{P,Q}.
\end{aligned}$$

For nullary predicates $P = \{\,|\,A\,\}$ and $Q = \{\,|\,B\,\}$ we write $A \wedge B$ for $P \cap Q$ and $A \vee B$ for $P \cup Q$. Then – as in Chapter 1 – the introduction axioms are

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

and the elimination axioms

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

3.3. Totality and induction

We now inductively define general totality predicates. Let us first look at some examples. The clauses[1] defining totality for the algebra $\mathbf{N}$ are

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

The least-fixed-point axiom is

$$\forall_n(T_{\mathbf{N}}n \to X0 \to \forall_n(T_{\mathbf{N}}n \to Xn \to X(Sn)) \to Xn).$$

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

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

$$T_{\mathbf{D}}0^{\mathbf{D}}, \qquad \forall_x(T_{\mathbf{D}}x \to \forall_y(T_{\mathbf{D}}y \to T_{\mathbf{D}}(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}xy))),$$

with least-fixed-point axiom

$$\begin{aligned}\forall_x(T_{\mathbf{D}}x \to X0^{\mathbf{D}} \to \\ \forall_x(T_{\mathbf{D}}x \to Xx \to \forall_y(T_{\mathbf{D}}y \to Xy \to X(\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}xy))) \to \\ Xx).\end{aligned}$$

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

Generally we define by induction on the type ρ

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

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

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

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

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

and

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

[1] They will be refined in 4.1.4, when decorations are available.

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

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

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

and

$$T_{\alpha_j}(X) := T_{\alpha_j}, \quad T_\xi(X) := X, \quad T_{\sigma\to\rho}(X) := \{\, f \mid \forall_x(T_\sigma x \to T_\rho(X)(fx))\,\}.$$

Another important notion is structural totality, where in the clauses all premises ending with Y are omitted.

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

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

and

$$\begin{aligned}
\mathrm{ST}_{\alpha_j}(X) &:= \{\, x \mid \top\,\} \quad \text{(will be omitted)},\\
\mathrm{ST}_\xi(X) &:= X,\\
\mathrm{ST}_{\sigma\to\rho}(X) &:= \{\, f \mid \forall_x(\mathrm{ST}_\sigma x \to \mathrm{ST}_\rho(X)(fx))\,\}.
\end{aligned}$$

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

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

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

$$\forall_{x,l}(\mathrm{ST}_{\mathbf{L}(\alpha)}l \to \mathrm{ST}_{\mathbf{L}(\alpha)}(x :: l))$$

with no assumptions on x.

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

$$\forall_l(\mathrm{ST}(l) \to X([]) \to \forall_{x,l}((\mathrm{ST} \cap X)l \to X(x :: l)) \to Xl^{\mathbf{L}(\rho)}).$$

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

$$\forall_l(\mathrm{ST}(l) \to X([]) \to \forall_{x,l}(\mathrm{ST}(l) \to Xl \to X(x :: l)) \to Xl^{\mathbf{L}(\rho)})$$

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

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

$$\forall_{as}(\mathrm{RT}_{\mathbf{L(T)}}(T_{\mathbf{T}})(as) \to T_{\mathbf{T}}(\mathrm{Branch}(as))).$$

Clearly all three notions of totality coincide for algebras without type parameters. Abbreviating $\forall_x(Tx \to A)$ by $\forall_{x\in T} A$ we obtain from the elimination axioms the usual *induction axioms*, for example

$$\mathrm{Ind}_{p,A(p)}\colon \forall_{p\in T}(A(\mathtt{tt}) \to A(\mathtt{ff}) \to A(p^{\mathbf{B}})),$$
$$\mathrm{Ind}_{n,A(n)}\colon \forall_{n\in T}(A(0) \to \forall_{n\in T}(A(n) \to A(Sn)) \to A(n^{\mathbf{N}})).$$

Parallel to general recursion, one can also consider *general induction*, which allows recurrence to *all* points "strictly below" the present one. For applications it is best to make the necessary comparisons w.r.t. a "measure function" μ. Then it suffices to use an initial segment of the ordinals instead of a well-founded set. For simplicity we here restrict ourselves to the segment given by ω, so the order we refer to is just the standard $<$-relation on the natural numbers. The principle of general induction then is

$$\forall_{\mu,x\in T}(\mathrm{Prog}^{\mu}_{x}A(x) \to A(x)) \tag{14}$$

where $\mathrm{Prog}^{\mu}_{x}A(x)$ expresses "progressiveness" w.r.t. the measure function μ and the order $<$:

$$\mathrm{Prog}^{\mu}_{x}A(x) := \forall_{x\in T}(\forall_{y\in T;\mu y<\mu x}A(y) \to A(x)).$$

It is easy to see that in our special case of the $<$-relation we can prove (14) from structural induction. However, it will be convenient to use general induction as a primitive axiom.

3.4. Coinductive definitions

We now extend TCF by allowing coinductive definitions as well as inductive ones. For instance, in the algebra $\mathbf{N}$ we can coinductively define *cototality* by the clause

$$^{\mathrm{co}}T_{\mathbf{N}}n \to n =^{\mathrm{d}} 0 \vee \exists_m(n =^{\mathrm{d}} Sm \wedge {}^{\mathrm{co}}T_{\mathbf{N}}m).$$

Its greatest-fixed-point axiom is

$$Xn \to \forall_n(Xn \to n =^{\mathrm{d}} 0 \vee \exists_m(n =^{\mathrm{d}} Sm \wedge ({}^{\mathrm{co}}T_{\mathbf{N}}m \vee Xm)) \to {}^{\mathrm{co}}T_{\mathbf{N}}n.$$

It expresses that every "competitor" X satisfying the same clause is a subset of $^{\mathrm{co}}T_{\mathbf{N}}$. The partial continuous functionals with $^{\mathrm{co}}T_{\mathbf{N}}$ interpreted as the cototal ideals for $\mathbf{N}$ provide a model of TCF extended by these axioms.

The greatest-fixed-point axiom is called the *coinduction* axiom for natural numbers.

Similarly, for the algebra $\mathbf{D}$ of derivations with constructors $0^{\mathbf{D}}$ and $\mathrm{C}^{\mathbf{D}\to\mathbf{D}\to\mathbf{D}}$ cototality is coinductively defined by the clause

$$^{\mathrm{co}}T_{\mathbf{D}}x \to x =^{\mathrm{d}} 0 \vee \exists_{y,z}(x =^{\mathrm{d}} \mathrm{C}yz \wedge {}^{\mathrm{co}}T_{\mathbf{D}}y \wedge {}^{\mathrm{co}}T_{\mathbf{D}}z).$$

Its greatest-fixed-point axiom is

$$Xx \to \forall_x(Xx \to x =^{\mathrm{d}} 0 \vee \exists_{y,z}(x =^{\mathrm{d}} \mathrm{C}yz \wedge ({}^{\mathrm{co}}T_{\mathbf{D}}x \vee Xy) \wedge ({}^{\mathrm{co}}T_{\mathbf{D}}x \vee Xz))) \to {}^{\mathrm{co}}T_{\mathbf{D}}x.$$

The partial continuous functionals with ${}^{\mathrm{co}}T_{\mathbf{D}}$ interpreted as the cototal ideals for $\mathbf{D}$ (i.e., the finite or infinite locally correct derivations) provide a model.

Generally, a coinductive predicate J is given by exactly one clause, which is of the form

$$\forall_{\vec{x}}(J\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i} \bigwedge_{\nu<n_i} A_{i\nu}(J)).$$

However, here we do not need this generality, and restrict ourselves to a special situation: every inductive predicate I gives rise to an important example of a coinductive predicate, its *dual* or *companion* ${}^{\mathrm{co}}I$. Let I be inductively defined by the clauses

$$\forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to I\vec{t}_i) \quad (i<k).$$

The conjunction of these k clauses is equivalent to

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

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

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

Its *greatest-fixed-point axiom* ${}^{\mathrm{co}}I^{+}$ is

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

More precisely, we extend the definition of formulas and predicates in Section 3.1 to also include the dual of an inductive predicate I, defined by

$${}^{\mathrm{co}}I(\vec{\alpha}, \vec{Y}, \vec{Z}\,) := \nu_X(\forall_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to X\vec{r}_i))_{i<k}.$$

The proof of the ex-falso-quodlibet theorem in Section 3.2 can be extended to also cover ${}^{\mathrm{co}}I$, even in cases where no nullary clause is present. To see this, use the greatest-fixed-point axiom for ${}^{\mathrm{co}}I$ with $X\vec{x} := \mathbf{F}$. Then any $\exists_{\vec{x}}(\vec{x} =^{\mathrm{d}} \vec{t} \wedge \bigwedge_{\nu<n} A_\nu({}^{\mathrm{co}}I \vee X))$ is provable, since $\vec{t} =^{\mathrm{d}} \vec{t}$ is, and also all $A_\nu({}^{\mathrm{co}}I \vee \mathbf{F})$ can be proved from $\mathbf{F}$ by induction hypothesis.

CHAPTER 4

Computational content of proofs

Proofs have two aspects: they provide insight into why an argument is correct, and they can also have computational content. The Brouwer-Heyting-Kolmogorov interpretation (BHK-interpretation for short) gives a good analysis of the latter.

A formula can be seen as a problem, and its proof as providing a solution to this problem. The clauses of the BHK-interpretation are:

(i) p proves $A \to B$ if and only if p is a construction transforming any proof q of A into a proof $p(q)$ of B;
(ii) $\bot$ is a proposition without proof;
(iii) p proves $\forall_{x \in D} A(x)$ if and only if p is a construction such that for all $d \in D$, $p(d)$ proves $A(d)$;

The problem with the BHK-interpretation clearly is its reliance on some unexplained notions, in particular

what is a "construction"?

what is a proof of a prime formula?

Here we propose to take

construction := computable functional,

proof of a prime formula $I\vec{r}$:= a "generation tree" for $I\vec{r}$.

For example, let Even be defined by the clauses Even(0) and $\forall_n(\mathrm{Even}(n) \to \mathrm{Even}(S(Sn)))$. A generation tree for Even(6) should consist of a single branch with nodes Even(0), Even(2), Even(4) and Even(6). More formally, such a generation tree can seen as an ideal in a certain algebra ι_I associated naturally with I.

Consider the more general situation when parameters are involved, i.e., when we have a proof (in TCF) of a closed formula $\forall_{\vec{x}}(\vec{A} \to I\vec{r})$. It is of obvious interest which of the variables $\vec{x}$ and assumptions $\vec{A}$ are actually used in the "solution" provided by the proof (in the sense of Kolmogorov (1932)). To be able to express dependence on and independence of such parameters we split each of our (only) logical connectives $\to, \forall$ into two variants, a "computational" one $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and a "non-computational" one $\to^{\mathrm{nc}}$

$, \forall^{\mathrm{nc}}$. This distinction (for the universal quantifier) is due to Berger (1993, 2005). One can view this "decoration" of $\to, \forall$ as turning our (minimal) logic into a "computational logic", which is able to express dependence on and independence of parameters. The rules for $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ are similar to the ones for $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$; they will be defined in 4.1.2.

Now the clauses of inductive predicates can and should be decorated as well, for instance in the form

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

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

A special case occurs when there is exactly one clause, and this clause only uses the non-computational $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$. Then we have $\iota_I = \mathbf{U}$, and hence prime formulas $I\vec{r}$ only have a trivial generation tree; in this sense they are without computational content. We call such inductive predicates *one-clause n.c.* defined, or *unitary*. Examples will be Leibniz equality $=^{\mathrm{d}}$, and the non-computational variants $\exists^{\mathrm{nc}}$ and $\wedge^{\mathrm{nc}}$ of the existential quantifier and of conjunction (see 4.1.3).

Formulas with such a one-clause n.c. inductive predicate as conlusion clearly are without computational content as well. These formulas are called *non-computational* (n.c.) or *Harrop formulas*. Moreover, a Harrop formula in a premise can be ignored when we are interested in the computational content of a proof of this formula: its only contribution would be of unit type. We will define the "type of a formula" (i.e., the type of its solution) accordingly; it will not involve the unit type.

The next thing to do is to properly accomodate the BHK-interpretation and define what it means that a term t "realizes" the formula A, written $t\ \mathbf{r}\ A$. In the prime formula case $I\vec{r}$ this will involve a predicate "t realizes $I\vec{r}$", which will be defined inductively as well, following the clauses of I. But since this is a "meta" statement already containing the term t representing a generation tree, we are not interested in the generation tree for such realizing formulas and consider them as non-computational.

Now we can formulate a consequence of Kolmogorov's view of a formula as a problem asking for a solution: we view a formula A and the existence of a realizer x of A as the same thing, and state it as an *invariance axiom*

$$\mathrm{Inv}_A \colon A \leftrightarrow \exists_x (x\ \mathbf{r}\ A).$$

A realizer of such an axiom will be the identity.

Finally we will define in 4.2.3 the "extracted term" $\mathrm{et}(M)$ of a proof M of a formula A. This is a term in T^+ incorporating the computational content of the proof M. In Section 4.3 we will then prove the important soundness theorem $\mathrm{et}(M)\ \mathbf{r}\ A$.

4.1. Decoration

4.1.1. Decorated predicates and formulas. We introduce decorated connectives $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$, and also decorated least-fixed-point operators $\mu^{\mathrm{c}}, \mu^{\mathrm{nc}}$. Moreover we distinguish two sorts of predicate variables, computationally relevant ones written $X, Y, Z \dots$ and non-computational ones written $X^{\mathrm{nc}}, Y^{\mathrm{nc}}, Z^{\mathrm{nc}} \dots$. Then we can define *decorated predicates and formulas* by essentially the same definition as in Section 3.1, provided we take both X and X^{nc} as initial decorated predicate forms. For readability we usually write $\to$, $\forall$, μ for $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$, μ^{c}, and also apply the following notational conventions.

(i) In the special case on a one-clause n.c. inductive predicate we use μ rather than μ^{nc}, since the fact that we have exactly one clause, which uses the non-computational $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ only makes it clear that no computational content is present.

(ii) In the general case of an n.c. inductive predicate $I^{\mathrm{nc}} := \mu^{\mathrm{nc}}_X \vec{K}$ we use the non-decorated $\to, \forall$ in the clauses $\vec{K}$, since elimination axiom $(I^{\mathrm{nc}})^-$ is restricted to n.c. competitor predicates. Hence the clauses will appear in n.c. parts of proofs only, where decorations are ignored.

Example. For the even numbers we have two variants:

$$\begin{aligned} \mathrm{Even} &:= \mu_X(X0, \forall^{\mathrm{nc}}_n(Xn \to X(S(Sn)))), \\ \mathrm{Even}^{\mathrm{nc}} &:= \mu^{\mathrm{nc}}_X(X0, \forall_n(Xn \to X(S(Sn)))). \end{aligned}$$

Generally for every c.r. inductive predicate I defined as $\mu_X \vec{K}$ we have a non-computational variant I^{nc} defined as $\mu^{\mathrm{nc}}_X \vec{K}$.

To every predicate or formula C we assign its *final predicate* $\mathrm{fp}(C)$ by

$$\begin{aligned} \mathrm{fp}(X) := X, \quad \mathrm{fp}(X^{\mathrm{nc}}) := X^{\mathrm{nc}} \qquad & \mathrm{fp}(P\vec{r}) := \mathrm{fp}(P) \\ \mathrm{fp}(\{\, \vec{x} \mid A \,\}) := \mathrm{fp}(A) \qquad & \mathrm{fp}(A \to^{\mathrm{c/nc}} B) := \mathrm{fp}(B) \\ \mathrm{fp}(I) := I, \quad \mathrm{fp}(I^{\mathrm{nc}}) := I^{\mathrm{nc}} \qquad & \mathrm{fp}(\forall^{\mathrm{c/nc}}_x A) := \mathrm{fp}(A) \end{aligned}$$

We call a predicate or formula C *non-computational* (n.c., or *Harrop*) if its final predicate $\mathrm{fp}(C)$ is of the form X^{nc} or I^{nc}, or it is a one-clause n.c. inductive predicate. The other predicates and formulas are called *computationally relevant* (c.r.).

Similarly we assign to every predicate or formula C its *non-computational variant* C^{nc}: we already have X^{nc} and I^{nc}, and in the other cases let

$$\begin{aligned} \{\, \vec{x} \mid A \,\}^{\mathrm{nc}} &:= \{\, \vec{x} \mid A^{\mathrm{nc}} \,\} \\ (P\vec{r})^{\mathrm{nc}} &:= P^{\mathrm{nc}}\vec{r} \\ (A \to^{\mathrm{c/nc}} B)^{\mathrm{nc}} &:= A \to B^{\mathrm{nc}} \end{aligned}$$

$$(\forall_x^{\mathrm{c/nc}} A)^{\mathrm{nc}} := \forall_x A^{\mathrm{nc}}$$

Clearly each C^{nc} is non-computational in the sense above.

Since decorations can be inserted arbitrarily and parameter predicate variables can be chosen as either n.c. or c.r. we obtain many useful variants of inductive predicates. For the existential quantifier we have

$$\begin{aligned}
\mathrm{ExD}_Y &:= \mu_X(\forall_x(Yx \to X)),\\
\mathrm{ExL}_Y &:= \mu_X(\forall_x(Yx \to^{\mathrm{nc}} X)),\\
\mathrm{ExR}_Y &:= \mu_X(\forall_x^{\mathrm{nc}}(Yx \to X)),\\
\mathrm{ExNc}_Y &:= \mu_X(\forall_x^{\mathrm{nc}}(Yx \to^{\mathrm{nc}} X)).
\end{aligned}$$

Here D is for "double", L for "left", R for "right". We will use the abbreviations

$$\begin{aligned}
\exists_x^{\mathrm{d}} A &:= \mathrm{ExD}_{\{x|A\}},\\
\exists_x^{\mathrm{l}} A &:= \mathrm{ExL}_{\{x|A\}},\\
\exists_x^{\mathrm{r}} A &:= \mathrm{ExR}_{\{x|A\}},\\
\exists_x^{\mathrm{nc}} A &:= \mathrm{ExNc}_{\{x|A\}}.
\end{aligned}$$

For intersection we only consider the nullary case (i.e., conjunction). Then

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

We use the abbreviations

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

For union again we only consider the nullary case (i.e., disjunction). Then

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

Here U stands for "uniform". The final nc-variant is used to suppress even the information which clause has been used. We use the abbreviations

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

For Leibniz equality we from now on take the definition

$$\mathrm{EqD} := \mu_X(\forall_x^{\mathrm{nc}} Xxx).$$

4.1.2. Logic with decorations. 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., define a modified notion of equality of formulas there).

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

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

and similarly

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

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

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

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

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

An alternative way to formulate these rules is simultaneously with the notion of the "extracted term" $\mathrm{et}(M)$ of a derivation M. This will be done in 4.2.3.

4.1.3. Decorated axioms. Consider a c.r. inductive predicate

$$I := \mu_X(\forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k}.$$

As in Section 3.2, for every $i < k$ we have a *clause* (or *introduction axiom*)

$$I^+_i\colon \forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to^{\mathrm{c/nc}} I\vec{r}_i). \tag{15}$$

Moreover, we have an *elimination axiom*

$$I^-\colon \forall^{\mathrm{nc}}_{\vec{x}}(I\vec{x} \to (\forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu}(I \cap^{\mathrm{d}} X))_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k} \to X\vec{x}). \tag{16}$$

For an n.c. inductive predicate I^{nc} the introduction axioms $(I^{\mathrm{nc}})^+_i$ are formed similarly, but with an important restriction: the elimination axiom $(I^{\mathrm{nc}})^-$ can only be used with X substituted by a *non-computational* competitor predicate. This is needed in the proof of the soundness theorem.

However, there is an important exception: in the special case of a *one-clause-nc* definition I (i.e., with only one clause involving $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ only) there are *no* restrictions on the elimination axiom. This is the case for Leibniz equality $=^{\mathrm{d}}$, and the non-computational variants $\exists^{\mathrm{nc}}$ and $\wedge^{\mathrm{nc}}$ of the existential quantifier and of conjunction.

For the decorated variants $\exists^{\mathrm{d}}, \exists^{\mathrm{l}}, \exists^{\mathrm{r}}, \exists^{\mathrm{nc}}, \wedge^{\mathrm{d}}, \wedge^{\mathrm{l}}, \wedge^{\mathrm{r}}, \wedge^{\mathrm{nc}}, \vee^{\mathrm{d}}, \vee^{\mathrm{l}}, \vee^{\mathrm{r}}, \vee^{\mathrm{u}}, \vee^{\mathrm{nc}}$ of the existential quantifier, conjunction and disjunction we obtain the following introduction and elimination axioms. For the existential quantifier we have (assuming $x \notin \mathrm{FV}(B)$)

$$(\exists^{\mathrm{d}})^+\colon \forall_x(A \to \exists^{\mathrm{d}}_x A), \qquad (\exists^{\mathrm{d}})^-\colon \exists^{\mathrm{d}}_x A \to \forall_x(A \to B) \to B,$$
$$(\exists^{\mathrm{l}})^+\colon \forall_x(A \to^{\mathrm{nc}} \exists^{\mathrm{l}}_x A), \qquad (\exists^{\mathrm{l}})^-\colon \exists^{\mathrm{l}}_x A \to \forall_x(A \to^{\mathrm{nc}} B) \to B,$$
$$(\exists^{\mathrm{r}})^+\colon \forall^{\mathrm{nc}}_x(A \to \exists^{\mathrm{r}}_x A), \qquad (\exists^{\mathrm{r}})^-\colon \exists^{\mathrm{r}}_x A \to \forall^{\mathrm{nc}}_x(A \to B) \to B,$$

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

Here for instance $(\exists^{\mathrm{d}})^+$ abbreviates $(\mathrm{ExD}_{\{x|A\}})_0^+$. Similar for $\wedge$ we have

$$(\wedge^{\mathrm{d}})^+\colon A \to B \to A \wedge^{\mathrm{d}} B, \qquad (\wedge^{\mathrm{d}})^-\colon A \wedge^{\mathrm{d}} B \to (A \to B \to C) \to C,$$
$$(\wedge^{\mathrm{l}})^+\colon A \to B \to^{\mathrm{nc}} A \wedge^{\mathrm{l}} B, \qquad (\wedge^{\mathrm{l}})^-\colon A \wedge^{\mathrm{l}} B \to (A \to B \to^{\mathrm{nc}} C) \to C,$$
$$(\wedge^{\mathrm{r}})^+\colon A \to^{\mathrm{nc}} B \to A \wedge^{\mathrm{r}} B, \qquad (\wedge^{\mathrm{r}})^-\colon A \wedge^{\mathrm{r}} B \to (A \to^{\mathrm{nc}} B \to C) \to C,$$
$$(\wedge^{\mathrm{nc}})^+\colon A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} A \wedge^{\mathrm{nc}} B$$

and $(\wedge^{\mathrm{nc}})^-\colon A \wedge^{\mathrm{nc}} B \to (A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} C) \to C$. For $\vee$ we have

$$(\vee^{\mathrm{d}})_0^+\colon A \to A \vee^{\mathrm{d}} B, \qquad (\vee^{\mathrm{d}})_1^+\colon B \to A \vee^{\mathrm{d}} B,$$
$$(\vee^{\mathrm{l}})_0^+\colon A \to A \vee^{\mathrm{l}} B, \qquad (\vee^{\mathrm{l}})_1^+\colon B \to^{\mathrm{nc}} A \vee^{\mathrm{l}} B,$$
$$(\vee^{\mathrm{r}})_0^+\colon A \to^{\mathrm{nc}} A \vee^{\mathrm{r}} B, \qquad (\vee^{\mathrm{r}})_1^+\colon B \to A \vee^{\mathrm{r}} B,$$
$$(\vee^{\mathrm{u}})_0^+\colon A \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B, \qquad (\vee^{\mathrm{u}})_1^+\colon B \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B,$$
$$(\vee^{\mathrm{nc}})_0^+\colon A \to^{\mathrm{nc}} A \vee^{\mathrm{nc}} B, \quad (\vee^{\mathrm{nc}})_1^+\colon B \to^{\mathrm{nc}} A \vee^{\mathrm{nc}} B$$

with elimination axioms

$$(\vee^{\mathrm{d}})^-\colon A \vee^{\mathrm{d}} B \to (A \to C) \to (B \to C) \to C,$$
$$(\vee^{\mathrm{l}})^-\colon A \vee^{\mathrm{l}} B \to (A \to C) \to (B \to^{\mathrm{nc}} C) \to C,$$
$$(\vee^{\mathrm{r}})^-\colon A \vee^{\mathrm{r}} B \to (A \to^{\mathrm{nc}} C) \to (B \to C) \to C,$$
$$(\vee^{\mathrm{u}})^-\colon A \vee^{\mathrm{u}} B \to (A \to^{\mathrm{nc}} C) \to (B \to^{\mathrm{nc}} C) \to C,$$
$$(\vee^{\mathrm{nc}})^-\colon A \vee^{\mathrm{nc}} B \to (A \to^{\mathrm{nc}} C) \to (B \to^{\mathrm{nc}} C) \to C \quad \text{for } C \text{ n.c.}$$

Lemma (Converse of $(\exists^{\mathrm{d}})^-, (\exists^{\mathrm{l}})^-, (\exists^{\mathrm{r}})^-, (\exists^{\mathrm{nc}})^-$). *Assume $x \notin \mathrm{FV}(B)$. The following are derivable.*

$$(\exists_x^{\mathrm{d}} A \to B) \to \forall_x(A \to B),$$
$$(\exists_x^{\mathrm{l}} A \to B) \to \forall_x(A \to^{\mathrm{nc}} B),$$
$$(\exists_x^{\mathrm{r}} A \to B) \to \forall_x^{\mathrm{nc}}(A \to B),$$
$$(\exists_x^{\mathrm{nc}} A \to B) \to \forall_x^{\mathrm{nc}}(A \to^{\mathrm{nc}} B).$$

Proof. Exercise. □

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

Note first that in our comparison we should use the arithmetical form $\forall_x(A \to \mathbf{F}) \to \mathbf{F}$ rather than the logical form $\forall_x(A \to \bot) \to \bot$ of the weak

existential quantifier. Otherwise there can be no relation, because $\bot$ is a predicate variable with computational content.

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

$$((\exists^{\mathrm{nc}}_x A \to \mathbf{F}) \to \mathbf{F}) \to \exists^{\mathrm{nc}}_x A. \tag{17}$$

Conversely, from $\tilde{\exists}_x A \to \exists^{\mathrm{nc}}_x A$ we can easily derive (17). Although usage of such an n.c. axiom does not have any effect on computational content, we prefer to avoid it.

4.1.4. Decorating totality and induction axioms. The inductive definitions in Section 3.3 are decorated as well, and then read as follows.

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

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

and

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

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

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

and

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

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

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

and

$$\begin{aligned}
\mathrm{ST}_{\alpha_j}(X) &:= \{\, x \mid \top\,\} \quad \text{(will be omitted)},\\
\mathrm{ST}_{\xi}(X) &:= X,
\end{aligned}$$

$$\mathrm{ST}_{\sigma\to\rho}(X) := \{\, f \mid \forall^{\mathrm{nc}}_x(\mathrm{ST}_\sigma x \to \mathrm{ST}_\rho(X)(fx))\,\}.$$

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

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

with no assumptions on x.

The least-fixed-point axiom for $\mathrm{ST}_{\mathbf{L}(\alpha)}$ is according to (16)

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

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

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

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

Generally, abbreviating $\forall^{\mathrm{nc}}_x(Tx \to A)$ by $\forall_{x\in T} A$ we obtain from the elimination axioms *computational induction axioms*, for example

$$\mathrm{Ind}_{p,A(p)} \colon \forall_{p\in T}(A(\mathtt{tt}) \to A(\mathtt{ff}) \to A(p^{\mathbf{B}})),$$
$$\mathrm{Ind}_{n,A(n)} \colon \forall_{n\in T}(A(0) \to \forall_{n\in T}(A(n) \to A(Sn)) \to A(n^{\mathbf{N}})).$$

The types of these formulas (as defined below in Section 4.2) will be the types of the recursion operators of the respective algebras.

The decorated form of the general induction schema is

$$\forall_{\mu\in T}\forall_{x\in T}(\mathrm{Prog}^\mu_x A(x) \to A(x)) \tag{18}$$

with

$$\mathrm{Prog}^\mu_x A(x) := \forall_{x\in T}(\forall_{y\in T}(\mu y < \mu x \to A(y)) \to A(x)).$$

The type of general induction (18) will be

$$(\alpha \to \mathbf{N}) \to \alpha \to (\alpha \to (\alpha \to \tau) \to \tau) \to \tau,$$

which is the type of the general recursion operator $\mathcal{F}$ defined in (10).

4.2. Realizers

The next thing to do is to view a formula A as a "computational problem", as done by Kolmogorov (1932). Then what should be the solution to the problem posed by the formula $I\vec{r}$, where I is inductively defined? The obvious idea here is to take a "generation tree", witnessing how the arguments $\vec{r}$ were put into I. For example, consider the clauses Even(0) and $\forall^{\mathrm{nc}}_n(\mathrm{Even}(n) \to \mathrm{Even}(S(Sn)))$. A generation tree for Even(6) should consist of a single branch with nodes Even(0), Even(2), Even(4) and Even(6).

When we want to generally define this concept of a generation tree, it seems natural to let the clauses of I determine the algebra to which such trees belong. Hence we will define ι_I to be the type $\mu_\xi(\kappa_0, \dots, \kappa_{k-1})$

generated from constructor types $\kappa_i := \tau(K_i)$, where K_i is the i-th clause of the inductive definition of I as $\mu_X(K_0, \dots K_{k-1})$, and $\tau(K_i)$ is the type of the clause K_i, relative to $\tau(X\vec{r}) := \xi$.

We begin with the definition of the type $\tau(A)$ of a formula A, the type of the solution to the problem posed by this formula. Then we define what it means for an x of type $\tau(A)$ to be a "realizer" (i.e., a solution) of the formula A. Next we assign to any derivation M of a formula A its "extracted term" $\mathrm{et}(M)$, which should be the realizer of A provided by the proof M. Finally we prove the soundness theorem, saying that this indeed is the case.

4.2.1. Types of predicates and formulas. We refine the distinction between computationally relevant (c.r.) and non-computational (n.c.) predicates and formulas given in 4.1.1 by providing a type in the former case. To indicate that there is no computational content we introduce a "nulltype" symbol $\circ$ and extend the use of $\rho \to \sigma$ by

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

Definition (Type $\tau(()C)$ of a predicate or formula C). Assume a global injective assignment of type variables ξ to c.r. predicate variables X. Let $\tau(C) := \circ$ if C is non-computational. In case C is c.r. let

$$\begin{aligned}
&\tau(X) := \xi,\\
&\tau(\{\, \vec{x} \mid A \,\}) := \tau(A),\\
&\tau(\underbrace{\mu_X(\forall^{\mathrm{nc}}_{\vec{x}_i}\forall_{\vec{y}_i}(\vec{A}_i \to^{\mathrm{nc}} \vec{B}_i \to X\vec{r}_i))_{i<k}}_{I}) := \underbrace{\mu_\xi(\tau(\vec{y}_i) \to \tau(\vec{B}_i) \to \xi)_{i<k}}_{\iota_I}.\\
&\tau(P\vec{r}) := \tau(P),\\
&\tau(A \to B) := (\tau(A) \to \tau(B)), \quad \tau(A \to^{\mathrm{nc}} B) := \tau(B),\\
&\tau(\forall_{x^\rho} A) := (\rho \to \tau(A)), \quad \tau(\forall^{\mathrm{nc}}_{x^\rho} A) := \tau(A),
\end{aligned}$$

We call ι_I the *algebra associated with* I.

4.2.2. Realizability. Assume that we have a global assignment giving for every (c.r.) predicate variable X of arity $\vec{\rho}$ an n.c. predicate variable $X^{\mathbf{r}}$ of arity $(\tau(X), \vec{\rho})$ together with an invariance axiom

$$\forall^{\mathrm{nc}}_{\vec{x}}(X\vec{x} \leftrightarrow \exists^{\mathrm{l}}_z X^{\mathbf{r}} z\vec{x}\,).$$

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

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

For a c.r. inductive predicate

$$I := \mu_X(\forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k}.$$

we define the witnessing predicate $I^{\mathbf{r}}$ by

$$I^{\mathbf{r}} := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(\forall_{\vec{x}_i,\vec{z}_i}((z_{i\nu}\ \mathbf{r}\ A_{i\nu})_{\nu<n_i} \to \mathrm{C}_i\vec{x}_i\vec{z}_i\ \mathbf{r}\ X\vec{r}_i))_{i<k}$$

with the understanding that instead of $(z_{i\nu}\ \mathbf{r}\ A_{i\nu})_{\nu<n_i} \to$

(i) if $A_{i\nu}$ is n.c. or followed by $\to^{\mathrm{nc}}$ we have $A_{i\nu} \to$ and there is no $z_{i\nu}$,
(ii) if $A_{i\nu}$ is c.r. and followed by $\to$ we have $z_{i\nu}\ \mathbf{r}\ A_{i\nu} \to$.

Only those x_{ij} with a computational $\forall_{x_{ij}}$ occur as arguments in $\mathrm{C}_i\vec{x}_i\vec{z}_i$. Here C_i is the i-th constructor of the algebra ι_I generated from the constructor types $\tau(K_i)$ with K_i the i-th clause of I. If I has a c.r. parameter predicate Y, then both Y and $Y^{\mathbf{r}}$ may appear in $I^{\mathbf{r}}$. For c.r. *formulas* let

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

Example. For the even numbers

$$\mathrm{Even} := \mu_X(X0, \forall^{\mathrm{nc}}_n(Xn \to X(S(Sn))))$$

we obtain the associated algebra $\iota_{\mathrm{Even}} = \mathbf{N}$ and

$$\mathrm{Even}^{\mathbf{r}} := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(0\ \mathbf{r}\ X0, \forall_{n,m}(m\ \mathbf{r}\ Xn \to Sm\ \mathbf{r}\ X(S(Sn)))).$$

The introduction axioms (15) are

$$\begin{aligned}
&(\mathrm{Even}^{\mathbf{r}})^+_0 \colon 0\ \mathbf{r}\ \mathrm{Even}(0),\\
&(\mathrm{Even}^{\mathbf{r}})^+_1 \colon \forall_{n,m}(m\ \mathbf{r}\ \mathrm{Even}(n) \to Sm\ \mathbf{r}\ \mathrm{Even}(S(Sn)))
\end{aligned}$$

Recall that the elimination axiom for an n.c. inductive predicate I^{nc} can only be used with non-computational competitor predicates (except one-clause-nc inductive predicates like $=^{\mathrm{d}}$, $\exists^{\mathrm{nc}}$ and $\wedge^{\mathrm{nc}}$). Hence the elimination axiom (16) is

$$\begin{aligned}
(\mathrm{Even}^{\mathbf{r}})^- \colon \forall_{n,m}(m\ \mathbf{r}\ \mathrm{Even}(n) \to Q^{\mathrm{nc}}00 \to\\
\forall_{n,m}(m\ \mathbf{r}\ \mathrm{Even}(n) \to Q^{\mathrm{nc}}mn \to Q^{\mathrm{nc}}(Sm, S(Sn))) \to\\
Q^{\mathrm{nc}}mn).
\end{aligned}$$

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

LEMMA (Realizers for $\exists^{\mathrm{d}}$, $\exists^{\mathrm{l}}$ and $\exists^{\mathrm{r}}$).

$$\langle x, z\rangle \mathbf{r} \exists^{\mathrm{d}}_x A \leftrightarrow z \mathbf{r} A \quad \textit{for } A \textit{ c.r.}$$
$$x \mathbf{r} \exists^{\mathrm{l}}_x A \leftrightarrow A \quad \textit{for } A \textit{ n.c.}$$
$$z \mathbf{r} \exists^{\mathrm{r}}_x A \leftrightarrow \exists^{\mathrm{nc}}_x (z \mathbf{r} A) \quad \textit{for } A \textit{ c.r.}$$

PROOF. *Case* $\exists^{\mathrm{d}}$. Recall

$$\mathrm{ExD}_Y := \mu_X(\forall_x(Yx^\rho \to X))$$

with the associated algebra $\iota_{\mathrm{ExD}_Y} := \rho \times \tau(Y)$. Then

$$\mathrm{ExD}^{\mathbf{r}}_{Y^{\mathbf{r}}} := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(\forall_{x,z}(z \mathbf{r} Yx \to \langle x, z\rangle \mathbf{r} X)).$$

Substituting $Y^{\mathbf{r}}$ by $\{\, z, x \mid z \mathbf{r} A \,\}$ in the introduction axiom (15) gives

$$(\mathrm{ExD}^{\mathbf{r}}_{\{z,x|z\mathbf{r}A\}})^+_0 \colon \forall_{x,z}(z \mathbf{r} A \to \langle x, z\rangle \mathbf{r} \exists^{\mathrm{d}}_x A)$$

where $\langle x, z\rangle \mathbf{r} \exists^{\mathrm{d}}_x A$ abbreviates $\mathrm{ExD}^{\mathbf{r}}_{\{z,x|z\mathbf{r}A\}}\langle x, z\rangle$. Conversely, the elimination axiom (16) is

$$(\mathrm{ExD}^{\mathbf{r}}_{Y^{\mathbf{r}}})^- \colon \forall_p(\mathrm{ExD}^{\mathbf{r}}_{Y^{\mathbf{r}}} p \to \forall_{x,z}(z \mathbf{r} Yx \to X\langle x, z\rangle) \to Xp).$$

Substituting X by $\{\, p \mid \mathrm{rht}(p) \mathbf{r} Y\mathrm{lft}(p) \,\}$ makes the middle part provable. Thus with $\{\, z, x \mid z \mathbf{r} A \,\}$ for $Y^{\mathbf{r}}$ we obtain $\forall_{x,z}(\langle x, z\rangle \mathbf{r} \exists^{\mathrm{d}}_x A \to z \mathbf{r} A)$ as a consequence of $(\mathrm{ExD}^{\mathbf{r}}_{\{z,x|z\mathbf{r}A\}})^-$.

Case $\exists^{\mathrm{l}}$. Recall

$$\mathrm{ExL}_Y := \mu_X(\forall_x(Yx^\rho \to^{\mathrm{nc}} X)).$$

Then

$$\mathrm{ExL}^{\mathbf{r}}_Y := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(\forall_x(Yx \to x \mathbf{r} X)).$$

Substituting Y by $\{\, x \mid A \,\}$ in the introduction axiom (15) gives

$$(\mathrm{ExL}^{\mathbf{r}}_{\{x|A\}})^+_0 \colon \forall_x(A \to x \mathbf{r} \exists^{\mathrm{l}}_x A)$$

where $x \mathbf{r} \exists^{\mathrm{l}}_x A$ abbreviates $\mathrm{ExL}^{\mathbf{r}}_{\{x|A\}} x$. Conversely, substituting Y by $\{\, x \mid A \,\}$ in the elimination axiom (16) gives

$$(\mathrm{ExL}^{\mathbf{r}}_{\{x|A\}})^- \colon \forall_x(x \mathbf{r} \exists^{\mathrm{l}}_x A \to \forall_x(A \to^{\mathrm{nc}} X^{\mathrm{nc}}) \to X^{\mathrm{nc}}).$$

With A for X^{nc} the middle part is provable (recall that A is assumed to be n.c.). Therefore we have $x \mathbf{r} \exists^{\mathrm{l}}_x A \to A$.

Case $\exists^{\mathrm{r}}$. Recall

$$\mathrm{ExR}_Y := \mu_X(\forall^{\mathrm{nc}}_x(Yx^\rho \to X)).$$

Then

$$\mathrm{ExR}^{\mathbf{r}}_{Y^{\mathbf{r}}} := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(\forall_{x,z}(z \mathbf{r} Yx \to z \mathbf{r} X)).$$

Substituting $Y^{\mathbf{r}}$ by $\{\, z, x \mid z\ \mathbf{r}\ A \,\}$ in the introduction axiom (15) gives

$$(\mathrm{ExR}^{\mathbf{r}}_{\{z,x|z\mathbf{r}A\}})^{+}_{0} : \forall_{x,z}(z\ \mathbf{r}\ A \to z\ \mathbf{r}\ \exists^{\mathrm{r}}_{x} A)$$

where $z\ \mathbf{r}\ \exists^{\mathrm{r}}_{x} A$ abbreviates $\mathrm{ExR}^{\mathbf{r}}_{\{z,x|z\mathbf{r}A\}} z$, whence $\forall_z(\exists^{\mathrm{nc}}_{x}(z\ \mathbf{r}\ A) \to z\ \mathbf{r}\ \exists^{\mathrm{r}}_{x} A)$. Conversely, the elimination axiom (16) is

$$(\mathrm{ExR}^{\mathbf{r}}_{Y^{\mathbf{r}}})^{-} : \forall_z(\mathrm{ExR}^{\mathbf{r}}_{Y^{\mathbf{r}}} z \to \forall_{x,z}(z\ \mathbf{r}\ Yx \to X^{\mathrm{nc}} z) \to X^{\mathrm{nc}} z).$$

Substituting X^{nc} by $\{\, z \mid \exists^{\mathrm{nc}}_{x}(z\ \mathbf{r}\ Yx) \,\}$ makes the middle part provable. Thus with $\{\, z, x \mid z\ \mathbf{r}\ A \,\}$ for $Y^{\mathbf{r}}$ we obtain $\forall_z(z\ \mathbf{r}\ \exists^{\mathrm{r}}_{x} A \to \exists^{\mathrm{nc}}_{x}(z\ \mathbf{r}\ A))$. □

Similarly we have

LEMMA (Realizers for $\wedge^{\mathrm{d}}$, $\wedge^{\mathrm{l}}$ and $\wedge^{\mathrm{r}}$).

$$z\ \mathbf{r}\ A \to w\ \mathbf{r}\ B \to \langle z, w\rangle\ \mathbf{r}\ (A \wedge^{\mathrm{d}} B) \quad \textit{for } A, B \textit{ c.r.}$$
$$z\ \mathbf{r}\ A \to B \to z\ \mathbf{r}\ (A \wedge^{\mathrm{l}} B) \quad \textit{for } A \textit{ c.r. and } B \textit{ n.c.}$$
$$A \to w\ \mathbf{r}\ B \to w\ \mathbf{r}\ (A \wedge^{\mathrm{r}} B) \quad \textit{for } A \textit{ n.c. and } B \textit{ c.r.}$$

$$\langle z, w\rangle\ \mathbf{r}\ (A \wedge^{\mathrm{d}} B) \to (z\ \mathbf{r}\ A) \wedge^{\mathrm{nc}} (w\ \mathbf{r}\ B) \quad \textit{for } A, B \textit{ c.r.}$$
$$z\ \mathbf{r}\ (A \wedge^{\mathrm{l}} B) \to (z\ \mathbf{r}\ A) \wedge^{\mathrm{nc}} B \quad \textit{for } A \textit{ c.r. and } B \textit{ n.c.}$$
$$w\ \mathbf{r}\ (A \wedge^{\mathrm{r}} B) \to A \wedge^{\mathrm{nc}} (w\ \mathbf{r}\ B) \quad \textit{for } A \textit{ n.c. and } B \textit{ c.r.}$$

PROOF. Exercise. □

LEMMA (Realizers for $\vee^{\mathrm{d}}$, $\vee^{\mathrm{l}}$, $\vee^{\mathrm{r}}$ and $\vee^{\mathrm{u}}$).

$$z\ \mathbf{r}\ A \to \mathrm{Inl}(z)\ \mathbf{r}\ (A \vee^{\mathrm{d}} B) \quad \textit{for } A, B \textit{ c.r.}$$
$$w\ \mathbf{r}\ B \to \mathrm{Inr}(w)\ \mathbf{r}\ (A \vee^{\mathrm{d}} B) \quad \textit{for } A, B \textit{ c.r.}$$
$$z\ \mathbf{r}\ A \to \mathrm{InlYsumu}(z)\ \mathbf{r}\ (A \vee^{\mathrm{l}} B) \quad \textit{for } A \textit{ c.r. and } B \textit{ n.c.}$$
$$B \to \mathrm{DummyR}\ \mathbf{r}\ (A \vee^{\mathrm{l}} B) \quad \textit{for } A \textit{ c.r. and } B \textit{ n.c.}$$
$$A \to \mathrm{DummyL}\ \mathbf{r}\ (A \vee^{\mathrm{r}} B) \quad \textit{for } A \textit{ n.c. and } B \textit{ c.r.}$$
$$w\ \mathbf{r}\ B \to \mathrm{InrUysum}(w)\ \mathbf{r}\ (A \vee^{\mathrm{r}} B) \quad \textit{for } A \textit{ n.c. and } B \textit{ c.r.}$$
$$A \to \mathtt{tt}\ \mathbf{r}\ (A \vee^{\mathrm{u}} B) \quad \textit{for } A, B \textit{ n.c.}$$
$$B \to \mathtt{ff}\ \mathbf{r}\ (A \vee^{\mathrm{u}} B) \quad \textit{for } A, B \textit{ n.c.}$$

and for C *n.c.*

$$u\ \mathbf{r}\ (A \vee^{\mathrm{d}} B) \to \forall_z(z\ \mathbf{r}\ A \to C) \to \forall_w(w\ \mathbf{r}\ B \to C) \to C) \quad \textit{for } A, B \textit{ c.r.}$$
$$u\ \mathbf{r}\ (A \vee^{\mathrm{l}} B) \to \forall_z(z\ \mathbf{r}\ A \to C) \to (B \to C) \to C) \quad \textit{for } A \textit{ c.r. and } B \textit{ n.c.}$$
$$u\ \mathbf{r}\ (A \vee^{\mathrm{r}} B) \to (A \to C) \to \forall_w(w\ \mathbf{r}\ B \to C) \to C) \quad \textit{for } A \textit{ n.c. and } B \textit{ c.r.}$$
$$u\ \mathbf{r}\ (A \vee^{\mathrm{u}} B) \to (A \to C) \to (B \to C) \to C) \quad \textit{for } A, B \textit{ n.c.}$$

PROOF. The introduction group follows easily from clauses of the decorated disjunctions. For the elimination group one needs to use the elimination axioms; this is left as an exercise. □

We can now state the *invariance axioms* Inv_A and prove that identities realize them.

AXIOM (Invariance under realizability).

$$\mathrm{Inv}_A\colon A \leftrightarrow \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A) \qquad \textit{for c.r. formulas } A. \tag{19}$$

LEMMA. *For c.r. formulas A we have*

$$(\lambda_z z)\ \mathbf{r}\ (A \to \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A)),$$
$$(\lambda_z z)\ \mathbf{r}\ (\exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A) \to A).$$

PROOF. Unfolding the definitions we obtain

$$(\lambda_z z)\ \mathbf{r}\ (A \to \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A))$$
$$\forall_z(z\ \mathbf{r}\ A \to z\ \mathbf{r}\ \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A))$$

and similarly

$$(\lambda_z z)\ \mathbf{r}\ (\exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A) \to A)$$
$$\forall_z(z\ \mathbf{r}\ \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A) \to z\ \mathbf{r}\ A).$$

But $z\ \mathbf{r}\ \exists^{\mathrm{l}}_z(z\ \mathbf{r}\ A)$ is equivalent to $z\ \mathbf{r}\ A$ by the lemma above. □

Important consequences of the invariance axioms are the theorems of choice and of independence of premise.

THEOREM (Choice). *From the invariance axioms we can derive*

$$\forall_x\exists^{\mathrm{l}}_y A(y) \to \exists^{\mathrm{l}}_f\forall_x A(fx) \quad \textit{for } A \textit{ n.c.} \tag{20}$$

$$\forall_x\exists^{\mathrm{d}}_y A(y) \to \exists^{\mathrm{d}}_f\forall_x A(fx) \quad \textit{for } A \textit{ c.r.} \tag{21}$$

PROOF. By the invariance axioms it suffices to find a realizer. (20).

$$(\lambda_f f)\ \mathbf{r}\ (\forall_x\exists^{\mathrm{l}}_y A(y) \to \exists^{\mathrm{l}}_f\forall_x A(fx))$$
$$\forall_f(f\ \mathbf{r}\ \forall_x\exists^{\mathrm{l}}_y A(y) \to f\ \mathbf{r}\ \exists^{\mathrm{l}}_f\forall_x A(fx))$$
$$\forall_f(\forall_x(fx\ \mathbf{r}\ \exists^{\mathrm{l}}_y A(y)) \to \forall_x A(fx))$$
$$\forall_f(\forall_x A(fx) \to \forall_x A(fx)).$$

(21).

$$(\lambda_g\langle\lambda_x\mathrm{lft}(gx), \lambda_x\mathrm{rht}(gx)\rangle)\ \mathbf{r}\ (\forall_x\exists^{\mathrm{d}}_y A(y) \to \exists^{\mathrm{d}}_f\forall_x A(fx))$$
$$\forall_g(g\ \mathbf{r}\ \forall_x\exists^{\mathrm{d}}_y A(y) \to \langle\lambda_x\mathrm{lft}(gx), \lambda_x\mathrm{rht}(gx)\rangle\ \mathbf{r}\ \exists^{\mathrm{d}}_f\forall_x A(fx))$$

$$\forall_g(\forall_x(gx\ \mathbf{r}\ \exists^{\mathrm{d}}_y A(y)) \to (\lambda_x \mathrm{rht}(gx))\ \mathbf{r}\ \forall_x A(\mathrm{lft}(gx)))$$
$$\forall_g(\forall_x(\mathrm{rht}(gx)\ \mathbf{r}\ A(\mathrm{lft}(gx))) \to \forall_x(\mathrm{rht}(gx)\ \mathbf{r}\ A(\mathrm{lft}(gx)))). \qquad \square$$

THEOREM (Independence of premise). *Assume $x \notin \mathrm{FV}(A)$. From the invariance axioms we can derive*

(22) $(A \to \exists^{\mathrm{l}}_x B) \to \exists^{\mathrm{l}}_x(A \to B)$ *for A, B n.c.*

(23) $(A \to^{\mathrm{nc}} \exists^{\mathrm{l}}_x B) \to \exists^{\mathrm{l}}_x(A \to B)$ *for B n.c.*

(24) $(A \to \exists^{\mathrm{d}}_x B) \to \exists^{\mathrm{d}}_x(A \to B)$ *for A n.c., B c.r.*

(25) $(A \to^{\mathrm{nc}} \exists^{\mathrm{d}}_x B) \to \exists^{\mathrm{d}}_x(A \to^{\mathrm{nc}} B)$ *for B c.r.*

PROOF. By the invariance axioms it suffices to find a realizer. (22).

$$(\lambda_x x)\ \mathbf{r}\ ((A \to \exists^{\mathrm{l}}_x B) \to \exists^{\mathrm{l}}_x(A \to B))$$
$$\forall_x(x\ \mathbf{r}\ (A \to \exists^{\mathrm{l}}_x B) \to x\ \mathbf{r}\ \exists^{\mathrm{l}}_x(A \to B))$$
$$\forall_x((A \to x\ \mathbf{r}\ \exists^{\mathrm{l}}_x B) \to x\ \mathbf{r}\ \exists^{\mathrm{l}}_x(A \to B))$$
$$\forall_x((A \to B) \to A \to B).$$

(23) is proved similarly. (24).

$$(\lambda_p p)\ \mathbf{r}\ ((A \to \exists^{\mathrm{d}}_x B(x)) \to \exists^{\mathrm{d}}_x(A \to B(x)))$$
$$\forall_p(p\ \mathbf{r}\ (A \to \exists^{\mathrm{d}}_x B(x)) \to p\ \mathbf{r}\ \exists^{\mathrm{d}}_x(A \to B(x)))$$
$$\forall_p((A \to p\ \mathbf{r}\ \exists^{\mathrm{d}}_x B(x)) \to \mathrm{rht}(p)\ \mathbf{r}\ (A \to B(\mathrm{lft}(p))))$$
$$\forall_p((A \to \mathrm{rht}(p)\ \mathbf{r}\ B(\mathrm{lft}(p))) \to A \to \mathrm{rht}(p)\ \mathbf{r}\ B(\mathrm{lft}(p))).$$

(25) is proved similarly. $\square$

LEMMA (Monotonicity). *Let C be a predicate or formula with all its strictly positive predicate variables among $\vec{X}$. Assume $\tau(C) = \rho(\vec{\alpha}\,)$. Let X_i be of arity $\vec{\rho}_i$ and $Y_i^{\boldsymbol{r}}$, $Z_i^{\boldsymbol{r}}$ of arity $(\beta_i, \vec{\rho}_i)$, $(\gamma_i, \vec{\rho}_i)$, respectively. Then we can derive*

$$\vec{w}\ \mathbf{r}\ (\vec{Y} \subseteq \vec{Z}\,) \to u\ \mathbf{r}^{\vec{Y^{\boldsymbol{r}}}}_{\vec{X}}\ C\vec{x} \to (\mathcal{M}^{\vec{\beta}\to\vec{\gamma}}_{\lambda_{\vec{\alpha}}\rho(\vec{\alpha}\,)} u\vec{w}\,)\ \mathbf{r}^{\vec{Z^{\boldsymbol{r}}}}_{\vec{X}}\ C\vec{x}$$

where

$$\vec{w}\ \mathbf{r}\ (\vec{Y} \subseteq \vec{Z}\,) := (\forall_{v_i}(Y_i^{\boldsymbol{r}} v_i \subseteq Z_i^{\boldsymbol{r}}(w_i v_i))_{i<n},$$
$$u\ \mathbf{r}^P_X\ A := A^{\boldsymbol{r}}[X^{\boldsymbol{r}} := P]u.$$

PROOF. By induction on C. *Case $I(\vec{P}\,)$.* We first deal with the case where I has its original predicate parameters $\vec{X}$; the general case will then follow by substitution. Let $\tau(I) = \iota(\vec{\alpha}\,)$. We must show

$$\vec{w}\ \mathbf{r}\ (\vec{Y} \subseteq \vec{Z}\,) \to u\ \mathbf{r}^{\vec{Y^{\mathbf{r}}}}_{\vec{X}}\ I\vec{x} \to (\mathcal{M}^{\vec{\beta}\to\vec{\gamma}}_{\iota} u\vec{w})\ \mathbf{r}^{\vec{Z^{\mathbf{r}}}}_{\vec{X}}\ I\vec{x}.$$

Fix $\vec{w}$ and abbreviate

$$Qu\vec{x} := (\mathcal{M}_\iota^{\vec{\beta}\to\vec{\gamma}} u\vec{w})\ \mathbf{r}_{\vec{X}}^{\vec{Z^{\mathbf{r}}}}\ I\vec{x}.$$

Let $u\ \mathbf{r}_{\vec{X}}^{\vec{Y^{\mathbf{r}}}}\ I\vec{x}$. Use $(I^{\mathbf{r}})^-$ with the n.c. competitor predicate Q:

$$(\forall_{\vec{x}_i,\vec{u}_i}((u_{i\nu}\ \mathbf{r}_{X,\vec{X}}^{I^{\mathbf{r}}(\vec{Y^{\mathbf{r}}})\cap Q,\vec{Y^{\mathbf{r}}}}\ A_{i\nu})_{\nu<n_i}\to Q(\mathrm{C}_i\vec{x}_i\vec{u}_i,\vec{r}_i)))_{i<k}\to I^{\mathbf{r}}(\vec{Y^{\mathbf{r}}})\subseteq Q.$$

It suffices to prove the premises. Fix $i<k$. By the conversion rule for $\mathcal{M}_\iota^{\vec{\beta}\to\vec{\gamma}}$ its conclusion $Q(\mathrm{C}_i\vec{x}_i\vec{u}_i,\vec{r}_i)$ is

$$\mathrm{C}_i\vec{x}_i(\underbrace{\mathcal{M}^{\vec{\beta},\iota(\vec{\beta})\to\vec{\gamma},\iota(\vec{\gamma})}_{\lambda_{\vec{\alpha},\alpha}\rho_{i\nu}(\vec{\alpha},\alpha)} u_{i\nu}\vec{w}(\mathcal{M}_\iota^{\vec{\beta}\to\vec{\gamma}}\cdot\vec{w})}_{v_{i\nu}})_{\nu<n_i}\ \mathbf{r}_{\vec{X}}^{\vec{Z^{\mathbf{r}}}}\ I\vec{r}_i.$$

Use the i-th introduction axiom for $I^{\mathbf{r}}$ with $\vec{Z^{\mathbf{r}}}$, i.e.,

$$(v_{i\nu}\ \mathbf{r}_{X,\vec{X}}^{I^{\mathbf{r}}(\vec{Z^{\mathbf{r}}}),\vec{Z^{\mathbf{r}}}}\ A_{i\nu})_{\nu<n_i}\to \mathrm{C}_i\vec{x}_i\vec{v_i}\ \mathbf{r}_{\vec{X}}^{\vec{Z^{\mathbf{r}}}}\ I\vec{r}_i.$$

It suffices to prove its premises. We use the induction hypothesis for $A_{i\nu}$:

$$\begin{aligned}&\vec{w}\ \mathbf{r}\ (\vec{Y}\subseteq\vec{Z}\,)\to\forall_u((I^{\mathbf{r}}(\vec{Y^{\mathbf{r}}})\cap Q)u\subseteq I^{\mathbf{r}}(\vec{Z^{\mathbf{r}}})(gu))\to\\ &u_{i\nu}\ \mathbf{r}_{X,\vec{X}}^{I^{\mathbf{r}}(\vec{Y^{\mathbf{r}}})\cap Q,\vec{Y^{\mathbf{r}}}}\ A_{i\nu}\to(\mathcal{M}^{\vec{\beta},\iota(\vec{\beta})\to\vec{\gamma},\iota(\vec{\gamma})}_{\lambda_{\vec{\alpha},\alpha}\rho_{i\nu}(\vec{\alpha},\alpha)}u_{i\nu}\vec{w}g)\ \mathbf{r}_{X,\vec{X}}^{I^{\mathbf{r}}(\vec{Z^{\mathbf{r}}}),\vec{Z^{\mathbf{r}}}}\ A_{i\nu}.\end{aligned}$$

Define $gu := \mathcal{M}_\iota^{\vec{\beta}\to\vec{\gamma}}u\vec{w}$. It remains to prove $(I^{\mathbf{r}}(\vec{Y^{\mathbf{r}}})\cap Q)u\vec{x}\to I^{\mathbf{r}}(\vec{Z^{\mathbf{r}}})(gu)\vec{x}$. The conclusion is $(gu)\ \mathbf{r}_{\vec{X}}^{\vec{Z^{\mathbf{r}}}}\ I\vec{x}$, i.e., $Qu\vec{x}$ which we have.

We now deal with the general case $I(\vec{P}\,)$. Recall that we just proved

$$\vec{f}\ \mathbf{r}\ (\vec{Y}\subseteq\vec{Z}\,)\to u\ \mathbf{r}_{\vec{X}}^{\vec{Y^{\mathbf{r}}}}\ I\vec{x}\to(\mathcal{M}_\iota^{\vec{\beta}\to\vec{\gamma}}u\vec{f}\,)\ \mathbf{r}_{\vec{X}}^{\vec{Z^{\mathbf{r}}}}\ I\vec{x}.$$

By substitution we obtain

$$\vec{w}\ \mathbf{r}\ (\vec{P}(\vec{Y})\subseteq\vec{P}(\vec{Z}\,))\to u\ \mathbf{r}_{\vec{X}}^{\vec{P^{\mathbf{r}}}(\vec{Y^{\mathbf{r}}})}\ I\vec{x}\to(\mathcal{M}_\iota^{\vec{\pi}(\vec{\beta})\to\vec{\pi}(\vec{\gamma})}u\vec{w})\ \mathbf{r}_{\vec{X}}^{\vec{P^{\mathbf{r}}}(\vec{Z^{\mathbf{r}}})}\ I\vec{x}.$$

Our goal is

$$\vec{f}\ \mathbf{r}\ (\vec{Y}\subseteq\vec{Z}\,)\to u\ \mathbf{r}_{\vec{X}}^{\vec{P^{\mathbf{r}}}(\vec{Y^{\mathbf{r}}})}\ I\vec{x}\to(\mathcal{M}^{\vec{\beta}\to\vec{\gamma}}_{\lambda_{\vec{\alpha}}\iota(\vec{\pi}(\vec{\alpha}))}u\vec{f}\,)\ \mathbf{r}_{\vec{X}}^{\vec{P^{\mathbf{r}}}(\vec{Z^{\mathbf{r}}})}\ I\vec{x}.$$

By induction hypothesis for $P_i(\vec{X})$

$$\vec{f}\ \mathbf{r}\ (\vec{Y}\subseteq\vec{Z}\,)\to(\mathcal{M}^{\vec{\beta}\to\vec{\gamma}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha})}\cdot\vec{f}\,)\ \mathbf{r}\ (P_i(\vec{Y})\subseteq P_i(\vec{Z}\,)).$$

Let $w_i := \mathcal{M}^{\vec{\beta}\to\vec{\gamma}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha})}\cdot\vec{f}$. The claim follows from

$$\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\iota(\vec{\pi}(\vec{\alpha}))}u\vec{f}:=\mathcal{M}_\iota^{\vec{\pi}(\vec{\sigma}\,)\to\vec{\pi}(\vec{\tau}\,)}u(\mathcal{M}^{\vec{\sigma}\to\vec{\tau}}_{\lambda_{\vec{\alpha}}\pi_i(\vec{\alpha})}\cdot\vec{f}\,)_{i<|\vec{\pi}\,|}$$

which is part of the definition of $\mathcal{M}$. The other cases are left as exercises. □

4.2.3. Extracted terms. For a derivation M of a c.r. formula A we define its *extracted term* $\mathrm{et}(M)$, of type $\tau(A)$. It will be a term in our term language of Section 2.2. This definition is relative to a fixed assignment of object variables to assumption variables: to every assumption variable u^A for a c.r. formula A we assign an object variable z_u of type $\tau(A)$.

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

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

It remains to define extracted terms for the axioms. Consider a (c.r.) inductively defined predicate I. For its introduction axioms (15) and elimination axiom (16) define $\mathrm{et}(I_i^+) := \mathrm{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where both the constructor C_i and the recursion operator $\mathcal{R}$ refer to the algebra ι_I associated with I. For the invariance axioms we take identities.

4.3. Soundness

We prove that the term extracted from a proof in TCF + Inv + $\mathrm{Ax}^{\mathrm{nc}}$ is a solution of the problem posed by the proven formula. Here $\mathrm{Ax}^{\mathrm{nc}}$ is an arbitrary set of non-computational formulas viewed as axioms.

Theorem (Soundness). *Let M be a derivation of a c.r. formula A from assumptions $u_i\colon C_i$ $(i < n)$. Then we can derive* $\mathrm{et}(M)$ **r** *A from assumptions z_{u_i}* **r** *C_i in case C_i is c.r. and C_i otherwise.*

If not stated otherwise, all derivations are in TCF + Inv + $\mathrm{Ax}^{\mathrm{nc}}$. The proof is by induction on M.

PROOF FOR THE LOGICAL RULES. *Case* $u\colon A$. Then $\mathrm{et}(u) = z_u$.

Case $(\lambda_{u^A} M^B)^{A\to B}$. We must find a derivation of $\mathrm{et}(\lambda_u M)\ \mathbf{r}\ (A \to B)$. *Subcase* A c.r. Then the goal is

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

Recall that $\mathrm{et}(\lambda_u M) = \lambda_{z_u}\mathrm{et}(M)$. Renaming z into z_u, our goal is to find a derivation of

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

since we identify terms with the same β-normal form. But by induction hypothesis we have a derivation of $\mathrm{et}(M)\ \mathbf{r}\ B$ from $z_u\ \mathbf{r}\ A$. An $\to$ and $\forall$ introduction then give the desired result. *Subcase* A n.c. Then the goal is

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

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

Case $M^{A\to B} N^A$. We must find a derivation of $\mathrm{et}(MN)\ \mathbf{r}\ B$. *Subcase* A c.r. Then $\mathrm{et}(MN) = \mathrm{et}(M)\mathrm{et}(N)$. By induction hypothesis we have derivations of

$$\mathrm{et}(M)\ \mathbf{r}\ (A \to B), \quad \text{which is} \quad \forall_z(z\ \mathbf{r}\ A\ \to\ \mathrm{et}(M) z\ \mathbf{r}\ B)$$

and of $\mathrm{et}(N)\ \mathbf{r}\ A$. Hence, again by logic, the claim follows. *Subcase* A n.c. Then $\mathrm{et}(MN) = \mathrm{et}(M)$. By induction hypothesis we have a derivation of

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

Using the derivation N^A we obtain $\mathrm{et}(M)\ \mathbf{r}\ B$.

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

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

Since we identify terms with the same β-normal form, the claim follows from the induction hypothesis.

Case $M^{\forall_x A(x)} t$. We must find a derivation of $\mathrm{et}(Mt)\ \mathbf{r}\ A(t)$. By definition $\mathrm{et}(Mt) = \mathrm{et}(M)t$, and by induction hypothesis we have a derivation of

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

Hence the claim.

Case $(\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}} B}$. Recall that $\mathrm{et}(\lambda_u M) = \mathrm{et}(M)$. We must find a derivation of

$$\mathrm{et}(M)\ \mathbf{r}\ (A \to^{\mathrm{nc}} B), \quad \text{which is} \quad A \to \mathrm{et}(M)\ \mathbf{r}\ B.$$

Assume A. *Subcase* A c.r. By induction hypothesis we have a derivation of $\mathrm{et}(M)\ \mathbf{r}\ B$ from $z\ \mathbf{r}\ A$. But by the invariance axioms $A \to \exists^{\mathrm{I}}_z(z\ \mathbf{r}\ A)$,

whence the claim follows logically. *Subcase* A n.c. By induction hypothesis we have a derivation of $\mathrm{et}(M)\ \mathbf{r}\ B$ from A.

Case $M^{A\to^{\mathrm{nc}}B}N^A$. Recall that $\mathrm{et}(MN) = \mathrm{et}(M)$. We must find a derivation of $\mathrm{et}(M)\ \mathbf{r}\ B$. By induction hypothesis we have a derivation of

$$\mathrm{et}(M)\ \mathbf{r}\ (A \to^{\mathrm{nc}} B), \quad \text{which is} \quad A\ \to\ \mathrm{et}(M)\ \mathbf{r}\ B.$$

Using the derivation N^A we obtain $\mathrm{et}(M)\ \mathbf{r}\ B$.

Case $(\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}$. Recall that $\mathrm{et}(\lambda_x M) = \mathrm{et}(M)$. Thus we must find a derivation of

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

But this follows from the induction hypothesis.

Case $M^{\forall^{\mathrm{nc}}_x A(x)}t$. Recall that $\mathrm{et}(Mt) = \mathrm{et}(M)$. We must find a derivation of $\mathrm{et}(Mt)\ \mathbf{r}\ A(t)$. By induction hypothesis we have a derivation of

$$\mathrm{et}(M)\ \mathbf{r}\ \forall^{\mathrm{nc}}_x A(x), \quad \text{which is} \quad \forall_x(\mathrm{et}(M)\ \mathbf{r}\ A(x)). \qquad \square$$

It remains to prove the soundness theorem for the axioms, i.e., that their extracted terms are realizers. Before doing anything general let us first look at an example. Totality for $\mathbf{N}$ has been inductively defined by the clauses

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

Its elimination axiom is

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

We show that their extracted terms 0, S and $\mathcal{R}$ are realizers. For the proof recall from the examples in 4.2.2 that the witnessing predicate $T^{\mathbf{r}}_{\mathbf{N}}$ is defined by the clauses

$$T^{\mathbf{r}}_{\mathbf{N}}00, \qquad \forall_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to T^{\mathbf{r}}_{\mathbf{N}}(Sm, Sn)),$$

and it has as its elimination axiom

$$\begin{aligned}\forall_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to X^{\mathrm{nc}}00 \to\ & \\ & \forall_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to X^{\mathrm{nc}}mn \to X^{\mathrm{nc}}(Sm, Sn)) \to \\ & X^{\mathrm{nc}}mn).\end{aligned}$$

Lemma. (a) $0\ \mathbf{r}\ T_{\mathbf{N}}0$ *and* $S\ \mathbf{r}\ \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to T_{\mathbf{N}}(Sn))$.
(b) $\mathcal{R}\ \mathbf{r}\ \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to X0 \to \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to Xn \to X(Sn)) \to Xn)$.

Proof. (a) $0\ \mathbf{r}\ T_{\mathbf{N}}0$ is defined to be $T^{\mathbf{r}}_{\mathbf{N}}00$. Moreover, by definition $S\ \mathbf{r}\ \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to T_{\mathbf{N}}(Sn))$ unfolds into $\forall_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to T^{\mathbf{r}}_{\mathbf{N}}(Sm, Sn))$.

(b) Let n, m be given and assume $m\ \mathbf{r}\ T_{\mathbf{N}}n$. Let further w_0, w_1 be such that $w_0\ \mathbf{r}\ X0$ and $w_1\ \mathbf{r}\ \forall^{\mathrm{nc}}_n(T_{\mathbf{N}}n \to Xn \to X(Sn))$, i.e.,

$$\forall_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to \forall_z(z\ \mathbf{r}\ Xn \to w_1mz\ \mathbf{r}\ X(Sn))).$$

Our goal is

$$\mathcal{R}mw_0w_1\ \mathbf{r}\ Xn =: Qmn.$$

To this end we use the elimination axiom for $T^{\mathbf{r}}_{\mathbf{N}}$ above. Hence it suffices to prove its premises $Q00$ and $\forall^{\mathrm{nc}}_{n,m}(T^{\mathbf{r}}_{\mathbf{N}}mn \to Qmn \to Q(Sm,Sn))$. By a conversion rule for $\mathcal{R}$ (cf. 2.2.2) the former is the same as $w_0\ \mathbf{r}\ P0$, which we have. For the latter assume n,m and its premises. We show $Q(Sm,Sn)$, i.e., $\mathcal{R}(Sm)w_0w_1\ \mathbf{r}\ X(Sn)$. By a conversion rule for $\mathcal{R}$ this is the same as

$$w_1m(\mathcal{R}mw_0w_1)\ \mathbf{r}\ X(Sn).$$

But with $z := \mathcal{R}mw_0w_1$ this follows from what we have. □

Proof for the axioms. We first prove soundness for introduction and elimination axioms of c.r. inductively defined predicates, and show that the extracted terms defined above are realizers. The proof uses the definition of $I^{\mathbf{r}}$ in 4.2.2.

By the clauses for $I^{\mathbf{r}}$ we clearly have $\mathrm{C}_i\ \mathbf{r}\ I^+_i$. For the elimination axiom we have to prove $\mathcal{R}\ \mathbf{r}\ I^-$, that is,

$$\mathcal{R}\ \mathbf{r}\ \forall^{\mathrm{nc}}_{\vec{x}}(I\vec{x} \to (K_i(I,P))_{i<k} \to P\vec{x}\,).$$

Let $\vec{x}, w$ be given and assume $w\ \mathbf{r}\ I\vec{x}$. Let further $w_0,\dots,w_{k-1}$ be such that $w_i\ \mathbf{r}\ K_i(I,P)$, i.e.,

$$\forall_{\vec{x}_i,\vec{v}_i}((v_{i\nu}^{\rho_{i\nu}(\iota\times\tau)}\ \mathbf{r}^{\{p,\vec{x}\,|\mathrm{lft}(p)\mathbf{r}I\vec{x}\wedge\mathrm{rht}(p)\mathbf{r}P\vec{x}\,\}}_{X}\ A_{i\nu})_{\nu<n_i} \to w_i\vec{x}_i\vec{v}_i\ \mathbf{r}\ P\vec{r}_i) \tag{26}$$

with $\rho_{i\nu}(\alpha) := \tau(A_{i\nu}(X))$ and τ the type of P. Here $\mathrm{lft}(p)$, $\mathrm{rht}(p)$ are shorthand for the two projections of p of type $\iota\times\tau$. Our goal is

$$\mathcal{R}w\vec{w}\ \mathbf{r}\ P\vec{x} =: Qw\vec{x}.$$

We use the elimination axiom (16) for $I^{\mathbf{r}}$ with the n.c. Q, i.e.,

$$(\forall_{\vec{x}_i,\vec{u}_i}((u_{i\nu}\ \mathbf{r}^{I^{\mathbf{r}}\cap Q}_X\ A_{i\nu})_{\nu<n_i} \to Q(\mathrm{C}_i\vec{x}_i\vec{u}_i,\vec{r}_i)))_{i<k} \to I^{\mathbf{r}} \subseteq Q.$$

Hence it suffices to prove its premises. Fix $i<k$. Assume $\vec{x}_i,\vec{u}_i$ and for every $\nu<n_i$ the premise

$$u_{i\nu}\ \mathbf{r}^{I^{\mathbf{r}}\cap Q}_X\ A_{i\nu}.$$

We show $Q(\mathrm{C}_i\vec{x}_i\vec{u}_i,\vec{r}_i)$, i.e.,

$$\mathcal{R}(\mathrm{C}_i\vec{x}_i\vec{u}_i)\vec{w}\ \mathbf{r}\ P\vec{r}_i.$$

By the conversion rules for $\mathcal{R}$ this is the same as

$$w_i\vec{x}_i(\underbrace{\mathcal{M}^{\iota\to\iota\times\tau}_{\lambda_\alpha\rho_{i\nu}(\alpha)}u_{i\nu}\lambda_w\langle w,\mathcal{R}w\vec{w}\,\rangle}_{v_{i\nu}})_{\nu<n_i}\ \mathbf{r}\ P\vec{r}_i.$$

To this end we apply (26) to $\vec{x}_i$ and $\vec{v}_i$. Its conclusion is what we want, and for its premises use

$$u_{i\nu}\ \mathbf{r}_X^{I^{\mathbf{r}}\cap Q}\ A_{i\nu} \to (\mathcal{M}^{\iota\to\iota\times\tau}_{\lambda_\alpha\rho_{i\nu}(\alpha)}u_{i\nu}\lambda_w\langle w,\mathcal{R}w\vec{w}\,\rangle)\ \mathbf{r}_X^{\{\,p,\vec{x}\,|\mathrm{lft}(p)\mathbf{r}I\vec{x}\wedge\mathrm{rht}(p)\mathbf{r}P\vec{x}\,\}}\ A_{i\nu}$$

which follows from an instance of the monotonicity lemma in 4.2.2

$$f\ \mathbf{r}\ (Y\subseteq Z)\to u\ \mathbf{r}_X^{Y^{\mathbf{r}}}\ A\to(\mathcal{M}^{\beta\to\gamma}_{\lambda_\alpha\rho(\alpha)}uf\,)\ \mathbf{r}_X^{Z^{\mathbf{r}}}\ A$$

where $\beta:=\tau(Y)$, $\gamma:=\tau(Z)$. To see this unfold the premise

$$\forall_{y,\vec{x}}(Y^{\mathbf{r}}y\vec{x}\to Z^{\mathbf{r}}(fy)\vec{x})\to u\ \mathbf{r}_X^{Y^{\mathbf{r}}}\ A\to(\mathcal{M}^{\beta\to\gamma}_{\lambda_{\vec{\alpha}}\rho(\vec{\alpha}\,)}uf\,)\ \mathbf{r}_X^{Z^{\mathbf{r}}}\ A$$

and substitute $\beta\mapsto\iota$, $\gamma\mapsto\iota\times\tau$, $y\mapsto w$, $u\mapsto u_{i\nu}$, $A\mapsto A_{i\nu}$, $Y^{\mathbf{r}}\mapsto I^{\mathbf{r}}\cap Q$, $Z^{\mathbf{r}}\mapsto\{\,p,\vec{x}\mid\mathrm{lft}(p)\mathbf{r}I\vec{x}\wedge\mathrm{rht}(p)\mathbf{r}P\vec{x}\,\}$, $f\mapsto\lambda_w\langle w,\mathcal{R}w\vec{w}\,\rangle$. Then the conclusion is what we want, and we have to prove the premise

$$(I^{\mathbf{r}}\cap Q)w\vec{x}\to\{\,p,\vec{x}\mid\mathrm{lft}(p)\mathbf{r}I\vec{x}\wedge\mathrm{rht}(p)\mathbf{r}P\vec{x}\,\}\langle w,\mathcal{R}w\vec{w}\,\rangle\vec{x}.$$

But this follows from the definition of Q.

For the introduction and elimination axioms for n.c. inductive predicates there is nothing to show since these axioms are n.c. as well. The only exception are one-clause-nc inductive predicates (for instance ExNc, CapNc and Leibniz equality) where the competitor predicate in the elimination axiom can be c.r. (cf. 4.1.3). But for these the identity is a realizer:

$$(\lambda_z z)\ \mathbf{r}\ \forall^{\mathrm{nc}}_{\vec{x}}(I^{\mathbf{r}}\vec{x}\to\forall^{\mathrm{nc}}_{\vec{y}}(\vec{A}\to^{\mathrm{nc}}X\vec{r}\,)\to X\vec{x}\,)$$
$$\forall_{\vec{x}}(I^{\mathbf{r}}\vec{x}\to(\lambda_z z)\ \mathbf{r}\ (\forall^{\mathrm{nc}}_{\vec{y}}(\vec{A}\to^{\mathrm{nc}}X\vec{r}\,)\to X\vec{x}\,))$$
$$\forall_{\vec{x}}(I^{\mathbf{r}}\vec{x}\to\forall_z(z\ \mathbf{r}\ \forall^{\mathrm{nc}}_{\vec{y}}(\vec{A}\to^{\mathrm{nc}}X\vec{r}\,)\to z\ \mathbf{r}\ X\vec{x}\,))$$
$$\forall_{\vec{x}}(I^{\mathbf{r}}\vec{x}\to\forall_z(\forall_{\vec{y}}(\vec{A}\to z\ \mathbf{r}\ X\vec{r}\,)\to z\ \mathbf{r}\ X\vec{x}\,))$$

which is an instance of the same elimination axiom.

We have already seen (in 4.2.2) that identities are realizers for the invariance axioms. □

Remark. We finally show that general recursion provides a realizer for general induction. Recall that according to (18) general induction is the schema

$$\forall_{\mu\in T}\forall_{x\in T}(\mathrm{Prog}^{\mu}_{x}Px\to Px)$$

where $\mathrm{Prog}^{\mu}_{x}Px$ expresses "progressiveness" w.r.t. the measure function μ and the order $<$:

$$\mathrm{Prog}^{\mu}_{x}Px:=\forall_{x\in T}(\forall_{y\in T}(\mu y<\mu x\to Py)\to Px).$$

We need to show

$$\mathcal{F}\ \mathbf{r}\ \forall_{\mu,x\in T}(\mathrm{Prog}^{\mu}_{x}Px\to Px),$$

that is,

$$\forall_{\mu,x\in T}\forall_g(g\ \mathbf{r}\ \forall_{x\in T}(\forall_{y\in T;\mu y<\mu x}Py\to Px)\to \mathcal{F}\mu xg\ \mathbf{r}\ Px).$$

Fix μ, x, g and assume the premise, which unfolds into

$$\forall_{x\in T,f}(\forall_{y\in T;\mu y<\mu x}(fy\ \mathbf{r}\ Py)\to gxf\ \mathbf{r}\ Px). \tag{27}$$

We have to show $\mathcal{F}\mu xg\ \mathbf{r}\ Px$. To this end we use an instance of general induction with the formula $\mathcal{F}\mu xg\ \mathbf{r}\ Px$, that is,

$$\forall_{\mu,x\in T}(\forall_{x\in T}(\forall_{y\in T;\mu y<\mu x}(\mathcal{F}\mu yg\ \mathbf{r}\ Py)\to \mathcal{F}\mu xg\ \mathbf{r}\ Px)\to \mathcal{F}\mu xg\ \mathbf{r}\ Px).$$

It suffices to prove the premise. Assume $\forall_{y\in T;\mu y<\mu x}(\mathcal{F}\mu yg\ \mathbf{r}\ Py)$ for a fixed $x\in T$. We must show $\mathcal{F}\mu xg\ \mathbf{r}\ Px$. Recall that by definition (10)

$$\mathcal{F}\mu xg = gxf_0 \quad \text{with } f_0 := \lambda_y[\mathbf{if}\ \mu y<\mu x\ \mathbf{then}\ \mathcal{F}\mu yg\ \mathbf{else}\ \varepsilon].$$

Hence we can apply (27) to x, f_0, and it remains to show

$$\forall_{y\in T;\mu y<\mu x}(f_0y\ \mathbf{r}\ Py).$$

Fix $y\in T$ with $\mu y<\mu x$. Then $f_0y=\mathcal{F}\mu yg$, and by the last assumption we have $\mathcal{F}\mu yg\ \mathbf{r}\ Py$.

CHAPTER 5

Computational content of classical proofs

Often in mathematics existence proofs are indirect, i.e., assuming that there is no solution of a problem one derives a contradiction.

Examples. 1. The first one is Fürstenberg's (1955) topological proof of the infinity of primes; it is the fifth of "Six proofs of the infinity of primes" in Aigner and Ziegler's "Proofs from THE BOOK" (2004, Problem 1). Call a set X of natural numbers *open* if every $x \in X$ starts an arithmetic progression contained in X, i.e., $\forall_{x\in X}\exists_{b>0}\forall_n(x + bn \in X)$. Clearly arbitrary unions and finite intersections of open sets are open, i.e., we have a topology. Let Np be the set of multiples of a positive number p. Then Np is not only open but also closed, since it is the complement is a finite union of open sets. The only thing we assume on primes is that every number > 1 has a prime divisor. Now assume that there would be only finitely many primes $p_0, p_1, \ldots, p_{m-1}$. Since $\bigcup_{l<m} Np_l$ is closed, its complement is open and therefore infinite, a contradiction.

2. The second example (suggested by Yiannis Moschovakis) is Euclid's result that the greatest common divisor of two integers is a linear combination of the two. The usual proof considers the ideal (a_1, a_2) generated by the two numbers and uses the fact the every ideal in the integers is a principal ideal. Its least positive element has a representation $|k_1a_1 - k_2a_2|$ with non-negative integers k_1, k_2. It is a common divisor of a_1 and a_2 (since otherwise the remainder of its division by a_i would be a smaller positive element of the ideal), and it is the greatest common divisor (since any common divisor of a_1 and a_2 must also be a divisor of $|k_1a_1 - k_2a_2|$).

Clearly such proofs only implicitly provide a solution, and it is a challenge to access the hidden computational content. In the first example this can be done[1] by slightly modifying the argument. Call X *uniformly open* if $\exists_{b>0}\forall_{x\in X}\forall_n(x + bn \in X)$; the number b is a *witness* of uniform openness. This time only finite unions and (as before) finite intersections of uniformly open sets are uniformly open; witnesses in both cases are finite products of the given witnesses. Since $\bigcup_{l<m} Np_l$ is uniformly closed with witness

[1] `git/minlog/examples/classical/fuerst.scm`

$b := \prod_{l<m} p_l$, its complement is uniformly open, and since it contains 1, it also contains $1 + b$. — This is the standard constructive proof of the infinity of primes.

For the second example[2] one can use general tools from proof theory, the main ones being the Dialectica interpretation of Gödel (1958), and a refined form of the so-called A-translation of Friedman (1978) and Dragalin (1979). Minlog implements both; however, here we only deal with the latter. Before going into the technicalities, let us first discuss what the A-translation is, and why there is a need to refine it.

5.1. A-translation

It is well known that from a derivation of a classical existential formula $\tilde{\exists}_y A := \forall_y(A \to \bot) \to \bot$ one generally cannot read off an instance. A simple example has been given by Kreisel: let R be a primitive recursive relation such that $\tilde{\exists}_z Rxz$ is undecidable. Clearly – even logically –

$$\vdash \forall_x \tilde{\exists}_y \forall_z (Rxz \to Rxy)$$

but there is no computable f satisfying

$$\forall_x \forall_z (Rxz \to R(x, f(x))),$$

for then $\tilde{\exists}_z Rxz$ would be decidable: it would be true if and only if $R(x, f(x))$ holds.

However, it is well known that in case $\tilde{\exists}_y G$ with G quantifier-free one *can* read off an instance. Here is a simple idea of how to prove this. Given a proof of $\tilde{\exists}_y G$ from D, replace $\bot$ anywhere in the proof by $\exists_y G$. Then the end formula $\forall_y(G \to \bot) \to \bot$ is turned into $\forall_y(G \to \exists_y G) \to \exists_y G$, and since the premise is trivially provable, we have the claim.

Unfortunately this simple argument is not quite correct. The problem is that both D and G may contain $\bot$ and hence are changed by the substitution. To obtain a constructive proof which can be used for extraction we should employ the arithmetical falsity $\mathbf{F}$ rather than the logical one, $\bot$. To repair this failure we require the following DG-property. Let $A^{\mathbf{F}}$ denote the result of substituting $\bot$ by $\mathbf{F}$ in A.

$$\begin{aligned} &D^{\mathbf{F}} \to D, \\ &(G^{\mathbf{F}} \to \bot) \to G \to \bot. \end{aligned} \tag{28}$$

Using (28) we can now correct the argument: from the given derivation of $D \to \forall_y(G \to \bot) \to \bot$ we obtain

$$D^{\mathbf{F}} \to \forall_y(G^{\mathbf{F}} \to \bot) \to \bot,$$

[2] `git/minlog/examples/classical/euclid.scm`

since $D^{\mathbf{F}} \to D$ and $(G^{\mathbf{F}} \to \bot) \to G \to \bot$. Substituting $\bot$ by $\exists_y G^{\mathbf{F}}$ gives

$$D^{\mathbf{F}} \to \forall_y(G^{\mathbf{F}} \to \exists_y G^{\mathbf{F}}) \to \exists_y G^{\mathbf{F}}.$$

Since $\forall_y(G^{\mathbf{F}} \to \exists_y G^{\mathbf{F}})$ is derivable we obtain $D^{\mathbf{F}} \to \exists_y G^{\mathbf{F}}$ as desired.

Therefore we need to pick our assumptions D and goal formulas G from appropriately chosen sets $\mathcal{D}$ and $\mathcal{G}$ which guarantee the DG-property (28).

An easy way to achieve this is to replace in D and G every atomic formula P different from $\bot$ by its double negation $(P \to \bot) \to \bot$. This corresponds to the original A-translation of Friedman (1978). However, then the computational content of the resulting constructive proof is unnecessarily complex, since each occurrence of $\bot$ gets replaced by the c.r. formula $\exists_y G^{\mathbf{F}}$.

Let us now see how we can eliminate unnecessary double negations. To this end we define certain sets $\mathcal{D}$ and $\mathcal{G}$ of formulas which ensure that their elements $D \in \mathcal{D}$ and $G \in \mathcal{G}$ satisfy the DG-property (28).

5.2. Definite and goal formulas

We simultaneously inductively define the classes $\mathcal{D}$ of definite formulas, $\mathcal{G}$ of goal formulas, $\mathcal{R}$ of relevant definite formulas and $\mathcal{I}$ of irrelevant goal formulas. Let D, G, R, I range over $\mathcal{D}$, $\mathcal{G}$, $\mathcal{R}$, $\mathcal{I}$, respectively, P over prime formulas distinct from $\bot$, and D_0 over quantifier-free formulas in $\mathcal{D}$.

$\mathcal{D}$, $\mathcal{G}$, $\mathcal{R}$ and $\mathcal{I}$ are generated by the clauses

(a) $R,\ P,\ I \to D,\ \forall_x D \in \mathcal{D}$.
(b) $I,\ \bot,\ R \to G,\ D_0 \to G \in \mathcal{G}$.
(c) $\bot,\ G \to R,\ \forall_x R \in \mathcal{R}$.
(d) $P,\ D \to I,\ \forall_x I \in \mathcal{I}$.

Let $A^{\mathbf{F}}$ denote $A[\bot := \mathbf{F}]$, and $\neg A$, $\neg_\bot A$ abbreviate $A \to \mathbf{F}$, $A \to \bot$.

Lemma. *We have derivations from* $\mathbf{F} \to \bot$ *of*

$$D^{\mathbf{F}} \to D, \tag{29}$$

$$G \to \neg_\bot \neg_\bot G^{\mathbf{F}}, \tag{30}$$

$$\neg_\bot \neg R^{\mathbf{F}} \to R, \tag{31}$$

$$I \to I^{\mathbf{F}}. \tag{32}$$

This result is due to Ishihara (2000) and has been (independently) rediscovered in Berger et al. (2002).

Proof. We prove (29)–(32) simultaneously by induction on formulas, starting with (31) and (32).

(31). *Case* $\bot$. Clearly $\neg_\bot \neg_\bot(\mathbf{F} \to \mathbf{F})$ is derivable.

Case $\forall_x R$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg_\perp\neg R^{\mathbf{F}} \to R}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg_\perp\neg\forall_x R^{\mathbf{F}}}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg R^{\mathbf{F}}}\quad \dfrac{}{R^{\mathbf{F}}}}{\mathbf{F}}}{\neg\forall_x R^{\mathbf{F}}}}{\bot}}{\neg_\perp\neg R^{\mathbf{F}}}}{R}}{\forall_x R}}{\neg_\perp\neg\forall_x R^{\mathbf{F}} \to \forall_x R}$$

Here we have used the induction hypothesis (31) for R.

Case $G \to R$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg_\perp\neg R^{\mathbf{F}} \to R}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg_\perp\neg(G^{\mathbf{F}} \to R^{\mathbf{F}})}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{G \to \neg_\perp\neg_\perp G^{\mathbf{F}}}\quad G}{\neg_\perp\neg_\perp G^{\mathbf{F}}}}{\neg_\perp G^{\mathbf{F}}}}{\bot}}{\neg_\perp\neg R^{\mathbf{F}}}}{R}}{\neg_\perp\neg(G^{\mathbf{F}} \to R^{\mathbf{F}}) \to G \to R}$$

Here we have used the induction hypotheses (31) for R and (30) for G.

(32). *Case* P. Clear, since $P^{\mathbf{F}}$ is P. *Case* $\forall_x I$. This is clear again, using the induction hypothesis (32) for I.

Case $D \to I$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{I \to I^{\mathbf{F}}}\quad \dfrac{\dfrac{\dfrac{}{D \to I}\quad \dfrac{\dfrac{}{D^{\mathbf{F}} \to D}\quad D^{\mathbf{F}}}{D}}{I}}{I^{\mathbf{F}}}}{(D \to I) \to D^{\mathbf{F}} \to I^{\mathbf{F}}}$$

Here we have used the induction hypotheses (32) for I and (29) for D.

(29). *Case* $\perp$. Then $\perp^{\mathbf{F}} = \mathbf{F}$ and the claim follows from our assumption $\mathbf{F} \to \perp$. *Case* P. Clear, since $P^{\mathbf{F}}$ is P. *Case* $\forall_x D$. By induction hypothesis (29) for D we have $D^{\mathbf{F}} \to D$, which clearly implies $\forall_x D^{\mathbf{F}} \to \forall_x D$.

Case R.

$$\dfrac{\dfrac{\dfrac{}{\neg_\perp\neg R^{\mathbf{F}} \to R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg R^{\mathbf{F}}} \quad \dfrac{}{R^{\mathbf{F}}}}{\mathbf{F}}}{\mathbf{F} \to \perp}}{\perp}}{\neg_\perp\neg R^{\mathbf{F}}}}{R}$$
$$R^{\mathbf{F}} \to R$$

Here we have used (31) (already proved) and $\mathbf{F} \to \perp$.

Case $I \to D$.

$$\dfrac{\dfrac{}{D^{\mathbf{F}} \to D} \qquad \dfrac{\dfrac{\dfrac{}{I \to I^{\mathbf{F}}} \quad \dfrac{}{I}}{I^{\mathbf{F}}}}{I^{\mathbf{F}} \to D^{\mathbf{F}}} \qquad \dfrac{}{D^{\mathbf{F}}}}{D}$$
$$(I^{\mathbf{F}} \to D^{\mathbf{F}}) \to I \to D$$

Here we have used the induction hypotheses (32) for I and (29) for D.

(30). *Case* $\perp$. Clear. *Case* P. Clear, since $P^{\mathbf{F}}$ is P. *Case* I. This is clear again, using (32) (already proved).

Case $R \to G$. We have to prove $(R \to G) \to \neg_\perp\neg_\perp(R^{\mathbf{F}} \to G^{\mathbf{F}})$. Let $\mathcal{D}_1[R \to G, \neg_\perp(R^{\mathbf{F}} \to G^{\mathbf{F}})]\colon \neg_\perp R$ be

$$\dfrac{\dfrac{\dfrac{}{G \to \neg_\perp\neg_\perp G^{\mathbf{F}}} \qquad \dfrac{\dfrac{}{R \to G} \quad \dfrac{}{R}}{G}}{\neg_\perp\neg_\perp G^{\mathbf{F}}} \qquad \dfrac{\dfrac{}{\neg_\perp(R^{\mathbf{F}} \to G^{\mathbf{F}})} \qquad \dfrac{\dfrac{}{G^{\mathbf{F}}}}{R^{\mathbf{F}} \to G^{\mathbf{F}}}}{\perp}}{\neg_\perp G^{\mathbf{F}}}}{\perp}$$
$$\neg_\perp R$$

(by induction hypothesis (30) for G) and $\mathcal{D}_2[\neg_\perp(R^{\mathbf{F}} \to G^{\mathbf{F}})]\colon \neg_\perp\neg R^{\mathbf{F}}$ be

$$\dfrac{\dfrac{}{\neg_\perp(R^{\mathbf{F}} \to G^{\mathbf{F}})} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg R^{\mathbf{F}}} \quad \dfrac{}{R^{\mathbf{F}}}}{\mathbf{F}}}{\vdots}}{G^{\mathbf{F}}}}{R^{\mathbf{F}} \to G^{\mathbf{F}}}}{\perp}$$
$$\neg_\perp\neg R^{\mathbf{F}}$$

Note that $G^{\mathbf{F}}$ is easily derivable from $\mathbf{F}$, since G can be assumed to be relevant (the case I was already done).

$$\dfrac{\dfrac{\mathcal{D}_1[R\to G,\neg_\perp(R^{\mathbf{F}}\to G^{\mathbf{F}})]}{\neg_\perp R}\qquad \dfrac{\mathcal{D}_2[\neg_\perp(R^{\mathbf{F}}\to G^{\mathbf{F}})]}{\neg_\perp\neg R^{\mathbf{F}}\to R}}{\dfrac{\bot}{(R\to G)\to\neg_\perp\neg_\perp(R^{\mathbf{F}}\to G^{\mathbf{F}})}}$$

Here we have used the induction hypothesis (31) for R.

Case $D_0\to G$. We have to prove $(D_0\to G)\to\neg_\perp\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})$. Let $\mathcal{D}_1[D_0\to G,\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})]\colon\neg_\perp D_0$ and $\mathcal{D}_2[\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})]\colon\neg_\perp\neg D_0^{\mathbf{F}}$ be as above. We use $(D_0^{\mathbf{F}}\to\bot)\to(\neg D_0^{\mathbf{F}}\to\bot)\to\bot$, i.e., case distinction on $D_0^{\mathbf{F}}$. Hence it suffices to derive from $D_0\to G$ and $\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})$ both $\neg_\perp D_0^{\mathbf{F}}$ and $\neg_\perp\neg D_0^{\mathbf{F}}$; recall that our goal is $(D_0\to G)\to\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})\to\bot$. The negative case is provided by $\mathcal{D}_2[\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})]$, and the positive case by

$$\dfrac{\dfrac{\mathcal{D}_1[D_0\to G,\neg_\perp(D_0^{\mathbf{F}}\to G^{\mathbf{F}})]}{\neg_\perp D_0}\qquad \dfrac{D_0^{\mathbf{F}}\to D_0\qquad D_0^{\mathbf{F}}}{D_0}}{\dfrac{\bot}{\neg_\perp D_0^{\mathbf{F}}}}$$

Here we have used the induction hypothesis (29) for D_0. □

Remark. Is $\mathcal{D}$ the largest class of formulas such that $D^{\mathbf{F}}\to D$ is provable intuitionistically? This is not the case, as the following example shows.

$$S:=\forall_x(((Qx\to\mathbf{F})\to\mathbf{F})\to Qx),$$
$$D:=(\forall_x Qx\to\bot)\to\bot.$$

One can easily derive $(S\to D)^{\mathbf{F}}\to S\to D$, since $S^{\mathbf{F}}$ is S and a derivation of $D^{\mathbf{F}}\to S\to D$ can be found easily.

However, $S\to D\notin\mathcal{D}$, since $D\notin\mathcal{D}$. This is because D is neither (i) in $\mathcal{R}$ nor (ii) of the form $I\to D_1$. For (i), observe that if D were in $\mathcal{R}$, then its premise $\forall_x Qx\to\bot$ would be in $\mathcal{G}$, hence $\forall_x Qx$ in $\mathcal{R}$, which is not the case. For (ii), observe that $\forall_x Qx\to\bot$ is not in $\mathcal{I}$ bcause $\bot\notin\mathcal{I}$.

It is an open problem to find a useful characterization of the class of formulas such that $D^{\mathbf{F}}\to D$ is provable intuitionistically.

We give some examples of definite and goal formulas. Keep in mind that $\mathcal{R}\subseteq\mathcal{D}$ and $\mathcal{I}\subseteq\mathcal{G}$.

- $P\in\mathcal{D}\cap\mathcal{I}$.

- $\bot \in \mathcal{R} \cap \mathcal{G}$.
- $P \to \bot \in \mathcal{R} \cap \mathcal{G}$.
- $(P \to \bot) \to \bot \in \mathcal{R} \cap \mathcal{G}$.

LEMMA. *$C \in \mathcal{D} \cap \mathcal{G}$ for C quantifier-free such that no implication in C has $\bot$ as its final conclusion, and $C \in \mathcal{R}$ $(\in \mathcal{I})$ if and only if $\bot$ is (is not) the final conclusion of C.*

PROOF. The cases P and $\bot$ are clear. *Case* $C \to R$. Since $C \in \mathcal{G}$ we have $C \to R \in \mathcal{R}$, and since $C \in \mathcal{D}$ and $R \in \mathcal{G}$ we have $C \to R \in \mathcal{G}$.

Case $C \to I$. Since $C \in \mathcal{I}$ (because $\bot$ is not the final conclusion of C) and $I \in \mathcal{D}$ we have $C \to I \in \mathcal{D}$, and since $C \in \mathcal{D}$ we have $C \to I \in \mathcal{I}$. □

Note that a further condition on C except being quantifier-free is needed since for instance $\bot \to P \notin \mathcal{D}$.

LEMMA. *For goal formulas $\vec{G} = G_1, \dots, G_n$ we have a derivation from $\mathbf{F} \to \bot$ of*

$$(\vec{G}^{\mathbf{F}} \to \bot) \to \vec{G} \to \bot. \tag{33}$$

PROOF. Assume $\mathbf{F} \to \bot$. By (30)

$$G_i \to (G_i^{\mathbf{F}} \to \bot) \to \bot$$

for all $i = 1, \dots, n$. Now the assertion follows by minimal logic: assume $\vec{G}^{\mathbf{F}} \to \bot$ and $\vec{G}$; we must show $\bot$. By $G_1 \to (G_1^{\mathbf{F}} \to \bot) \to \bot$ it suffices to prove $G_1^{\mathbf{F}} \to \bot$. Assume $G_1^{\mathbf{F}}$. By $G_2 \to (G_2^{\mathbf{F}} \to \bot) \to \bot$ it suffices to prove $G_2^{\mathbf{F}} \to \bot$. Assume $G_2^{\mathbf{F}}$. Repeating this pattern, we finally have assumptions $G_1^{\mathbf{F}}, \dots, G_n^{\mathbf{F}}$ available, and obtain $\bot$ from $\vec{G}^{\mathbf{F}} \to \bot$. □

5.3. Extraction from weak existence proofs

THEOREM (Strong from weak existence proofs). *Assume that for arbitrary formulas $\vec{A}$, definite formulas $\vec{D}$ and goal formulas $\vec{G}$ we have a derivation $M_{\vec{\exists}}$ of*

$$\vec{A} \to \vec{D} \to \forall_y(\vec{G} \to \bot) \to \bot. \tag{34}$$

Then from $\mathbf{F} \to \bot$ and $\mathbf{F} \to P$ for all prime formulas P in $\vec{D}, \vec{G}$ we can derive

$$\vec{A} \to \vec{D}^{\mathbf{F}} \to \forall_y(\vec{G}^{\mathbf{F}} \to \bot) \to \bot.$$

In particular, substitution of the formula

$$\exists_y \vec{G}^{\mathbf{F}} := \exists_y(G_1^{\mathbf{F}} \wedge \dots \wedge G_n^{\mathbf{F}})$$

for $\bot$ yields a derivation $M_\exists$ from the $\mathbf{F} \to P$ of

$$\vec{A}[\bot := \exists_y \vec{G}^{\mathbf{F}}] \to \vec{D}^{\mathbf{F}} \to \exists_y \vec{G}^{\mathbf{F}}. \tag{35}$$

PROOF. The first assertion follows from (29) (to infer $\vec{D}$ from $\vec{D}^{\mathbf{F}}$) and (33) (to infer $\vec{G} \to \bot$ from $\vec{G}^{\mathbf{F}} \to \bot$). The second assertion is a simple consequence since $\forall_y(\vec{G}^{\mathbf{F}} \to \exists_y \vec{G}^{\mathbf{F}})$ and $\mathbf{F} \to \exists_y \vec{G}^{\mathbf{F}}$ are both derivable. □

We shall apply the method of realizability to extract computational content from the resulting strong existence proof $M_\exists$. Recall that this proof essentially follows the given weak existence proof $M_{\tilde{\exists}}$. The only difference is that proofs of (29) (to infer $\vec{D}$ from $\vec{D}^{\mathbf{F}}$) and (33) (to infer $\vec{G} \to \bot$ from $\vec{G}^{\mathbf{F}} \to \bot$) have been inserted. Therefore the extracted term can be structured in a similar way, with one part determined solely by $M_{\tilde{\exists}}$ and another part depending only on the definite formulas $\vec{D}$ and and goal formulas $\vec{G}$. – For simplicity let $\vec{G}$ consist of a single goal formula G.

To make the method work we need to assume that all prime formulas P appearing in $\vec{D}^{\mathbf{F}}, G^{\mathbf{F}}$ are n.c. (for instance, equalities).

LEMMA. *Let D be a definite and G a goal formula. Assume that all prime formulas P in $D^{\mathbf{F}}, G^{\mathbf{F}}$ are n.c.*

(a) *We have a term t_D such that*

$$D^{\mathbf{F}} \to t_D\ \mathbf{r}\ D$$

is derivable from $\forall_y(\mathbf{F} \to y\ \mathbf{r}\ \bot)$ and $\mathbf{F} \to P$.

(b) *We have a term s_G such that*

$$(G^{\mathbf{F}} \to v\ \mathbf{r}\ \bot) \to w\ \mathbf{r}\ G \to s_G v w\ \mathbf{r}\ \bot$$

is derivable from $\forall_y(\mathbf{F} \to y\ \mathbf{r}\ \bot)$ and $\mathbf{F} \to P$.

PROOF. By the assumption all formulas $D^{\mathbf{F}}, G^{\mathbf{F}}$ are n.c. as well.

(a) By (29) we have a derivation N_D of $D^{\mathbf{F}} \to D$ from assumptions $\mathbf{F} \to \bot$ and $\mathbf{F} \to P$. By the soundness theorem we can take $t_D := \mathrm{et}(N_D)$.

(b) By (30) we have a derivation H_G of $(G^{\mathbf{F}} \to \bot) \to G \to \bot$ from assumptions $\mathbf{F} \to \bot$ and $\mathbf{F} \to P$. Observe that the following are equivalent:

$$\mathrm{et}(H_G)\ \mathbf{r}\ ((G^{\mathbf{F}} \to \bot) \to G \to \bot),$$
$$\forall_{v,w}(v\ \mathbf{r}\ (G^{\mathbf{F}} \to \bot) \to w\ \mathbf{r}\ G \to \mathrm{et}(H_G) v w\ \mathbf{r}\ \bot),$$
$$\forall_{v,w}((G^{\mathbf{F}} \to v\ \mathbf{r}\ \bot) \to w\ \mathbf{r}\ G \to \mathrm{et}(H_G) v w\ \mathbf{r}\ \bot).$$

Hence we can take $s_G := \mathrm{et}(H_G)$. □

THEOREM (Extraction from weak existence proofs). *Assume that for definite formulas $\vec{D}$ and a goal formula $G(y)$ we have a derivation $M_{\tilde{\exists}}$ of*

$$\vec{D} \to \forall_y(G(y) \to \bot) \to \bot.$$

Assume that all prime formulas P in $\vec{D}^{\mathbf{F}}, G^{\mathbf{F}}(y)$ are n.c. Let $t_1, \dots, t_n$ and s be terms for $D_1, \dots, D_n$ and G according to parts (a) and (b) of the lemma above. Then from assumptions $\mathbf{F} \to P$ we can derive

$$\vec{D}^{\mathbf{F}} \to G^{\mathbf{F}}(\mathrm{et}(M'_{\vec{\exists}})t_1 \dots t_n s),$$

where $M'_{\vec{\exists}}$ is the result of substituting $\exists_y G^{\mathbf{F}}(y)$ for $\bot$ in $M_{\vec{\exists}}$.

Proof. By the soundness theorem we have

$$\mathrm{et}(M_{\vec{\exists}})\ \mathbf{r}\ (\vec{D} \to \forall_y(G(y) \to \bot) \to \bot),$$
$$\forall_{\vec{u},x}(\vec{u}\ \mathbf{r}\ \vec{D} \to x\ \mathbf{r}\ \forall_y(G(y) \to \bot) \to \mathrm{et}(M_{\vec{\exists}})\vec{u}x\ \mathbf{r}\ \bot),$$
$$\forall_{\vec{u},x}(\vec{u}\ \mathbf{r}\ \vec{D} \to \forall_{y,w}(w\ \mathbf{r}\ G(y) \to xyw\ \mathbf{r}\ \bot) \to \mathrm{et}(M_{\vec{\exists}})\vec{u}x\ \mathbf{r}\ \bot).$$

Instantiating $\vec{u}, x$ by $\vec{t}, s$, respectively, we obtain

$$\vec{t}\ \mathbf{r}\ \vec{D} \to \forall_{y,w}(w\ \mathbf{r}\ G(y) \to syw\ \mathbf{r}\ \bot) \to \mathrm{et}(M_{\vec{\exists}})\vec{t}s\ \mathbf{r}\ \bot.$$

Hence by part (a) of the lemma above we have a derivation of

$$\vec{D}^{\mathbf{F}} \to \forall_{y,w}(w\ \mathbf{r}\ G(y) \to syw\ \mathbf{r}\ \bot) \to \mathrm{et}(M_{\vec{\exists}})\vec{t}s\ \mathbf{r}\ \bot$$

from $\forall_y(\mathbf{F} \to y\ \mathbf{r}\ \bot)$ and $\mathbf{F} \to P$. Substituting $\bot$ by $\exists_y G^{\mathbf{F}}(y)$ gives

$$\vec{D}^{\mathbf{F}} \to \forall_{y,w}((w\ \mathbf{r}\ G(y))[\bot := \exists_y G^{\mathbf{F}}(y)] \to G^{\mathbf{F}}(syw)) \to G^{\mathbf{F}}(\mathrm{et}(M'_{\vec{\exists}})\vec{t}s)$$

from $\mathbf{F} \to P$. Substituting $\bot$ by $\exists_y G^{\mathbf{F}}(y)$ in the formula derived in part (b) of the lemma above gives

$$(G^{\mathbf{F}}(y) \to G^{\mathbf{F}}(v)) \to (w\ \mathbf{r}\ G(y))[\bot := \exists_y G^{\mathbf{F}}(y)] \to G^{\mathbf{F}}(svw)$$

from $\mathbf{F} \to P$. Instantiating this with $v := y$ we obtain a derivation of

$$\vec{D}^{\mathbf{F}} \to G^{\mathbf{F}}(\mathrm{et}(M'_{\vec{\exists}})\vec{t}s)$$

from $\mathbf{F} \to P$, as required. □

5.4. Applications

5.4.1. List reversal. We first give an informal weak existence proof for list reversal. Write vw for the result $v * w$ of appending the list w to the list v, vx for the result $v * x$: of appending the one element list x: to the list v, and xv for the result $x :: v$ of constructing a list by writing an element x in front of a list v, and omit the parentheses in $R(v, w)$ for (typographically) simple arguments. Assume

$$\mathrm{InitR}\colon R([], []),$$
$$\mathrm{GenR}\colon \forall_{v,w,x}(Rvw \to R(vx, xw)).$$

We view R as a predicate variable without computational content. The reader should not be confused: of course these formulas involving R do

express how a computation of the reverted list should proceed. But the predicate R itself only represents the graph of the list reversal function.

Let us now prove

$$\forall_v \tilde{\exists}_w Rvw \qquad (:= \forall_v(\forall_w(Rvw \to \bot) \to \bot)). \tag{36}$$

Fix R, v and assume `InitR`, `GenR` and the "false" assumption $u\colon \forall_w \neg Rvw$; we need to derive a contradiction. To this end we prove that all initial segments of v are non-revertible, which contradicts `InitR`. More precisely, from u and `GenR` we prove

$$\forall_{v_2} A(v_2) \quad \text{with } A(v_2) := \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg Rv_1 w)$$

by induction on v_2. For $v_2 = []$ this follows from $u_0\colon v_1\ [] = v$ and our ("false") assumption u. For the step case, assume $u_1\colon v_1(xv_2) = v$, fix w and assume further $u_2\colon Rv_1 w$. We must derive a contradiction. We use the induction hypotheses with $v_1 x$ and xw to obtain the desidered contradiction. This requires us to prove (i) $(v_1 x)v_2 = v$ and (ii) $R(v_1 x, xw)$. But (i) follows from u_1 using properties of the append function, and (ii) follows from u_2 using `GenR`.

We formalize this proof, to prepare it for the refined A-translation. The following lemmata will be used:

$$\begin{aligned}
&\mathrm{Compat}'\colon \ \forall^{\mathrm{nc}}_{v,w}(v =^{\mathrm{d}} w \to Xw \to Xv),\\
&\mathrm{EqToEqD}\colon \forall_{v,w}(v = w \to v =^{\mathrm{d}} w).
\end{aligned}$$

The proof term is

$$\begin{aligned}
M :=& \lambda_{R,v}\lambda_{u_{\mathrm{InitR}}}\lambda_{u_{\mathrm{GenR}}}\lambda_u^{\forall_w \neg Rvw}(\\
&\mathrm{Ind}_{v_2,A(v_2)} v R v M_{\mathrm{Base}} M_{\mathrm{Step}}\ []\ \mathrm{Truth}^{[]\, v=v}\ []\ u_{\mathrm{InitR}})
\end{aligned}$$

with

$$\begin{aligned}
M_{\mathrm{Base}} :=& \lambda_{v_1}\lambda_{u_0}^{v_1[]=v}(\\
&\mathrm{Compat}'\,\{\, v \mid \forall_w \neg Rvw \,\}\, R\, v\, v_1\, v\, (\mathrm{EqToEqD}\, v_1 v u_0) u),\\
M_{\mathrm{Step}} :=& \lambda_{x,v_2}\lambda_{u_0}^{A(v_2)}\lambda_{v_1}\lambda_{u_1}^{v_1(xv_2)=v}\lambda_w\lambda_{u_2}^{Rv_1 w}(\\
&u_0(v_1 x)u_1(xw)(u_{\mathrm{GenR}} v_1 w x u_2)).
\end{aligned}$$

We now have a proof M of $\forall_v \tilde{\exists}_w Rvw$ from $\mathrm{InitR}\colon D_1$ and $\mathrm{GenR}\colon D_2$, with $D_1 := R([],[])$ and $D_2 := \forall_{v,w,x}(Rvw \to R(vx, xw))$. Using the refined A-translation we can replace $\bot$ throughout by $\exists_w Rvw$. The end formula $\tilde{\exists}_w Rvw := \neg\forall_w \neg Rvw := \forall_w(Rvw \to \bot) \to \bot$ is turned into $\forall_w(Rvw \to \exists_w Rvw) \to \exists_w Rvw$. Since its premise is an instance of existence introduction we obtain a derivation $M^{\exists}$ of $\exists_w Rvw$. Moreover, in this case neither the D_i nor any of the axioms used involves $\bot$ in its uninstantiated formulas, and

hence the correctness of the proof is not affected by the substitution. The term `neterm` extracted in Minlog from a formalization of the proof above is

```
[R,v](Rec list nat=>list nat=>list nat=>list nat)v([v0,v1]v1)
 ([x,v0,g,v1,v2]g(v1++x:)(x::v2)) (Nil nat) (Nil nat)
```

with `g` a variable for binary functions on lists. In fact, the underlying algorithm defines an auxiliary function h by

$$h([], v_1, v_2) := v_2, \qquad h(xv, v_1, v_2) := h(v, v_1 x, x v_2)$$

and gives the result by applying h to the original list and twice $[]$.

Notice that the second argument of h is not needed. However, its presence makes the algorithm quadratic rather than linear, because in each recursion step $v_1 x$ is computed, and the list append function is defined by recursion on its first argument. We will be able to get rid of this superfluous second argument by decorating the proof. It will turn out that in the proof (by induction on v_2) of the formula $A(v_2) := \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg R v_1 w))$, the variable v_1 is not used computationally. Hence, in the decorated version of the proof, we can use $\forall^{\mathrm{nc}}_{v_1}$.

5.4.2. Integer square roots. For an unbounded function $f\colon \mathbb{N} \to \mathbb{N}$ with $f(0) = 0$ we prove

$$\forall_n \tilde{\exists}_m (f(m) \le n < f(m+1)).$$

If e.g. $f(m) = m^2$, then this formula expresses the existence of an integer square root $m := [\sqrt{n}]$ for any n. More formally, we prove

$$\forall_n \tilde{\exists}_m (\neg(n < f(m)) \land n < f(m+1)) \tag{37}$$

from the assumptions

$$v_1\colon \forall_n \neg(n < f(0)), \qquad v_2\colon \forall_n (n < f(g(n))).$$

Here $<^{\mathbb{N}\to\mathbb{N}\to\mathbb{B}}$ is the characteristic function of the natural ordering of the natural numbers. We expressed $f(m) \le n$ by $\neg(n < f(m))$ and $f(0) = 0$ by $\forall_n \neg(n < f(0))$ to keep the formal proof as simple as possible. In order to have purely universal assumptions we had to express the unboundedness of f by a witnessing function g.

Now let us prove (37). Let n be given and assume

$$u\colon \forall_m (\neg(n < f(m)) \to n < f(m+1) \to \bot).$$

We have to show $\bot$. From v_1 and u we inductively get $\forall_m \neg(n < f(m))$. For $m := g(n)$ this yields a contradiction to v_2.

In Minlog, this proof is implemented as follows, after loading `nat.scm`:

```
(add-var-name "f" "g" (py "nat=>nat"))

(set-goal "all f,g,n(
 all n(n<f 0 -> bot) -> all n n<f(g n) ->
 excl m((n<f m -> bot) ! n<f(m+1)))")
(assume "f" "g" "n" "v1" "v2" "u")
(assert "all m(n<f m -> bot)")
 (ind)
 (use "v1")
 (use "u")
(assume "Assertion")
(use-with "Assertion" (pt "g n") "?")
(use "v2")
;; Proof finished.
(save "IntSqRt")
```

We have saved the proof and now normalize it.

```
(define proof (theorem-name-to-proof "IntSqRt"))
(define nproof (np proof))
```

Now we can apply the refined A-translation followed by term extraction and normalization:

```
(define eterm
 (atr-min-excl-proof-to-structured-extracted-term nproof))
(define neterm (nt eterm))
(pp (rename-variables neterm))
```

and obtain

```
[f,f0,n](Rec nat=>nat)(f0 n)0([n0,n1][if (n<f n0) n1 n0])
```

Informally, the result is $h(g(n))$ where $h\colon \mathbf{N} \to \mathbf{N}$ is defined by

$$h(0) = 0, \qquad h(m+1) = \begin{cases} h(m) & \text{if } n{<}f(m) \\ m & \text{else.} \end{cases}$$

CHAPTER 6

Decorating proofs

In this chapter we are interested in "fine-tuning" the computational content of proofs, by inserting decorations. Here is an example (due to Constable) of why this is of interest. Suppose that in a proof M of a formula C we have made use of a case distinction based on an auxiliary lemma stating a disjunction, say $L\colon A \vee B$. Then the extract $\mathrm{et}(M)$ will contain the extract $\mathrm{et}(L)$ of the proof of the auxiliary lemma, which may be large. Now suppose further that in the proof M of C the only computationally relevant use of the lemma was which one of the two alternatives holds true, A or B. We can express this fact by using a weakened form of the lemma instead: $L'\colon A \vee^{\mathrm{u}} B$. Since the extract $\mathrm{et}(L')$ is a boolean, the extract of the modified proof has been "purified" in the sense that the (possibly large) extract $\mathrm{et}(L)$ has disappeared.

In Section 4.1 we consider the question of "optimal" decorations of proofs: suppose we are given an undecorated proof, and a decoration of its end formula. The task then is to find a decoration of the whole proof (including a further decoration of its end formula) in such a way that any other decoration "extends" this one. Here "extends" just means that some connectives have been changed into their more informative versions, disregarding polarities. We show that such an optimal decoration exists, and give an algorithm to construct it.

We then consider some applications: list reversal, computing the Fibonacci numbers in continuation passing style, and finally the Maximal Scoring Segment (MSS) algorithm. For instance in the latter case, directly deriving such an algorithm from a proof leads to quadratic complexity. We will see that the (automatically found) optimal decoration of this proof results in a linear extracted algorithm.

6.1. Decoration algorithm

We denote the *sequent* of a proof M by $\mathrm{Seq}(M)$; it consists of its *context* and *end formula*.

The *proof pattern* $\mathrm{P}(M)$ of a proof M is the result of marking in c.r. parts of M (i.e., not above a n.c. formula) all occurrences of implications

and universal quantifiers as non-computational, except the "uninstantiated" formulas of axioms and theorems. For instance, the induction axiom for $\mathbf{N}$ consists of the uninstantiated formula $\forall_n(X0 \to \forall_n(Xn \to X(Sn)) \to Xn^{\mathbf{N}})$ with a predicate variable X and a predicate substitution $X \mapsto \{\, x \mid A(x) \,\}$. Notice that a proof pattern in most cases is not a correct proof, because at axioms formulas may not fit.

We say that a formula D *extends* C if D is obtained from C by changing some (possibly zero) of its occurrences of non-computational implications and universal quantifiers into their computational variants $\to$ and $\forall$.

A proof N *extends* M if (i) N and M are the same up to variants of implications and universal quantifiers in their formulas, and (ii) every formula in c.r. parts of M is extended by the corresponding one in N. Every proof M whose proof pattern $\mathrm{P}(M)$ is U is called a *decoration* of U.

Notice that if a proof N extends another one M, then $\mathrm{FV}(\mathrm{et}(N))$ is essentially (that is, up to extensions of assumption formulas) a superset of $\mathrm{FV}(\mathrm{et}(M))$. This can be proven by induction on N.

In the sequel we assume that every axiom has the property that for every extension of its formula we can find a further extension which is an instance of an axiom, and which is the least one under all further extensions that are instances of axioms. This property clearly holds for axioms whose uninstantiated formula only has $\to$ and $\forall$, for instance induction. However, in $\forall_n(A(0) \to \forall_n(A(n) \to A(Sn)) \to A(n^{\mathbf{N}}))$ the given extension of the four A's might be different. One needs to pick their "least upper bound" as further extension.

We will define a *decoration algorithm*, assigning to every proof pattern U and every extension of its sequent an "optimal" decoration M_∞ of U, which further extends the given extension of its sequent.

THEOREM. *Under the assumption above, for every proof pattern U and every extension of its sequent* $\mathrm{Seq}(U)$ *we can find a decoration M_∞ of U such that*

(a) $\mathrm{Seq}(M_\infty)$ *extends the given extension of* $\mathrm{Seq}(U)$*, and*
(b) M_∞ *is* optimal *in the sense that any other decoration M of U whose sequent* $\mathrm{Seq}(M)$ *extends the given extension of* $\mathrm{Seq}(U)$ *has the property that M also extends M_∞.*

PROOF. By induction on derivations. It suffices to consider derivations with a c.r. endformula. For axioms the validity of the claim was assumed, and for assumption variables it is clear.

Case $(\to^{\mathrm{nc}})^+$. Consider the proof pattern

$$\dfrac{\begin{array}{c}\Gamma, u\colon A\\ \vert\, U\\ B\end{array}}{A \to^{\mathrm{nc}} B}\,(\to^{\mathrm{nc}})^+, u$$

with a given extension $\Delta \Rightarrow C \to^{\mathrm{nc}} D$ or $\Delta \Rightarrow C \to D$ of its sequent $\Gamma \Rightarrow A \to^{\mathrm{nc}} B$. Applying the induction hypothesis for U with sequent $\Delta, C \Rightarrow D$, one obtains a decoration M_∞ of U whose sequent $\Delta_1, C_1 \Rightarrow D_1$ extends $\Delta, C \Rightarrow D$. Now apply $(\to^{\mathrm{nc}})^+$ in case the given extension is $\Delta \Rightarrow C \to^{\mathrm{nc}} D$ and $x_u \notin \mathrm{FV}(\mathrm{et}(M_\infty))$, and $\to^+$ otherwise.

For (b) consider a decoration $\lambda_u M$ of $\lambda_u U$ whose sequent extends the given extended sequent $\Delta \Rightarrow C \to^{\mathrm{nc}} D$ or $\Delta \Rightarrow C \to D$. Clearly the sequent $\mathrm{Seq}(M)$ of its premise extends $\Delta, C \Rightarrow D$. Then M extends M_∞ by induction hypothesis for U. If $\lambda_u M$ derives a non-computational implication then the given extended sequent must be of the form $\Delta \Rightarrow C \to^{\mathrm{nc}} D$ and $x_u \notin \mathrm{FV}(\mathrm{et}(M))$, hence $x_u \notin \mathrm{FV}(\mathrm{et}(M_\infty))$. But then by construction we have applied $(\to^{\mathrm{nc}})^+$ to obtain $\lambda_u M_\infty$. Hence $\lambda_u M$ extends $\lambda_u M_\infty$. If $\lambda_u M$ does not derive a non-computational implication, the claim follows immediately.

Case $(\to^{\mathrm{nc}})^-$. Consider a proof pattern

$$\dfrac{\begin{array}{c}\Phi, \Gamma\\ \vert\, U\\ A \to^{\mathrm{nc}} B\end{array}\qquad\begin{array}{c}\Gamma, \Psi\\ \vert\, V\\ A\end{array}}{B}\,(\to^{\mathrm{nc}})^-$$

We are given an extension $\Pi, \Delta, \Sigma \Rightarrow D$ of $\Phi, \Gamma, \Psi \Rightarrow B$. Then we proceed in alternating steps, applying the induction hypothesis to U and V.

(1) The induction hypothesis for U for the extension $\Pi, \Delta \Rightarrow A \to^{\mathrm{nc}} D$ of its sequent gives a decoration M_1 of U whose sequent $\Pi_1, \Delta_1 \Rightarrow C_1 \to^{\mathrm{c/nc}} D_1$ extends $\Pi, \Delta \Rightarrow A \to^{\mathrm{nc}} D$, where $\to^{\mathrm{c/nc}}$ means $\to$ or $\to^{\mathrm{nc}}$. This already suffices if A is n.c., since then the extension $\Delta_1, \Sigma \Rightarrow C_1$ of V is a correct proof (recall that in n.c. parts of a proof decorations of implications and universal quantifiers can be ignored). If A is c.r.:

(2) The induction hypothesis for V for the extension $\Delta_1, \Sigma \Rightarrow C_1$ of its sequent gives a decoration N_2 of V whose sequent $\Delta_2, \Sigma_2 \Rightarrow C_2$ extends $\Delta_1, \Sigma \Rightarrow C_1$.

(3) The induction hypothesis for U for the extension $\Pi_1, \Delta_2 \Rightarrow C_2 \to^{\mathrm{c/nc}} D_1$ of its sequent gives a decoration M_3 of U whose sequent $\Pi_3, \Delta_3 \Rightarrow C_3 \to^{\mathrm{c/nc}} D_3$ extends $\Pi_1, \Delta_2 \Rightarrow C_2 \to^{\mathrm{c/nc}} D_1$.

(4) The induction hypothesis for V for the extension $\Delta_3, \Sigma_2 \Rightarrow C_3$ of its sequent gives a decoration N_4 of V whose sequent $\Delta_4, \Sigma_4 \Rightarrow C_4$ extends

$\Delta_3, \Sigma_2 \Rightarrow C_3$. This process is repeated until no further proper extension of Δ_i, C_i is returned. Such a situation will always be reached since there is a maximal extension, where all connectives are maximally decorated. But then we easily obtain (a): Assume that in (4) we have $\Delta_4 = \Delta_3$ and $C_4 = C_3$. Then the decoration

$$\dfrac{\begin{array}{cc}\Pi_3, \Delta_3 & \Delta_4, \Sigma_4 \\ \mid M_3 & \mid N_4 \\ C_3 \to^{\mathrm{c/nc}} D_3 & C_4\end{array}}{D_3} \to^{-}$$

of UV derives a sequent $\Pi_3, \Delta_3, \Sigma_4 \Rightarrow D_3$ extending $\Pi, \Delta, \Sigma \Rightarrow D$.

For (b) we need to consider a decoration MN of UV whose sequent $\mathrm{Seq}(MN)$ extends the given extension $\Pi, \Delta, \Sigma \Rightarrow D$ of $\Phi, \Gamma, \Psi \Rightarrow B$. We must show that MN extends M_3N_4. To this end we go through the alternating steps again.

(1) Since the sequent $\mathrm{Seq}(M)$ extends $\Pi, \Delta \Rightarrow A \to^{\mathrm{nc}} D$, the induction hypothesis for U for the extension $\Delta \Rightarrow A \to^{\mathrm{nc}} D$ of its sequent ensures that M extends M_1.

(2) Since then the sequent $\mathrm{Seq}(N)$ extends $\Delta_1, \Sigma \Rightarrow C_1$, the induction hypothesis for V for the extension $\Delta_1, \Sigma \Rightarrow C_1$ of its sequent ensures that N extends N_2.

(3) Therefore $\mathrm{Seq}(M)$ extends the sequent $\Pi_1, \Delta_2 \Rightarrow C_2 \to^{\mathrm{c/nc}} D_1$, and the induction hypothesis for U for the extension $\Pi_1, \Delta_2 \Rightarrow C_2 \to^{\mathrm{c/nc}} D_1$ of U's sequent ensures that M extends M_3.

(4) Therefore $\mathrm{Seq}(N)$ extends $\Delta_3, \Sigma_2 \Rightarrow C_3$, and induction hypothesis for V for the extension $\Delta_3, \Sigma_2 \Rightarrow C_3$ of V's sequent ensures that N also extends N_4.

But since $\Delta_4 = \Delta_3$ and $C_4 = C_3$ by assumption, MN extends the decoration M_3N_4 of UV constructed above.

Case $(\forall^{\mathrm{nc}})^+$. Consider a proof pattern

$$\dfrac{\begin{array}{c}\Gamma \\ \mid U \\ A\end{array}}{\forall^{\mathrm{nc}}_x A} (\forall^{\mathrm{nc}})^+$$

with a given extension $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C$ or $\Delta \Rightarrow \forall_x C$ of its sequent. Applying the induction hypothesis for U with sequent $\Delta \Rightarrow C$, one obtains a decoration M_∞ of U whose sequent $\Delta_1 \Rightarrow C_1$ extends $\Delta \Rightarrow C$. Now apply $(\forall^{\mathrm{nc}})^+$ in case the given extension is $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C$ and $x \notin \mathrm{FV}(\mathrm{et}(M_\infty))$, and $\forall^+$ otherwise.

For (b) consider a decoration $\lambda_x M$ of $\lambda_x U$ whose sequent extends the given extended sequent $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C$ or $\Delta \Rightarrow \forall_x C$. Clearly the sequent $\mathrm{Seq}(M)$

of its premise extends $\Delta \Rightarrow C$. Then M extends M_∞ by induction hypothesis for U. If $\lambda_x M$ derives a non-computational generalization, then the given extended sequent must be of the form $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C$ and $x \notin \mathrm{FV}(\mathrm{et}(M))$, hence $x \notin \mathrm{FV}(\mathrm{et}(M_\infty))$ (by the remark above). But then by construction we have applied $(\forall^{\mathrm{nc}})^+$ to obtain $\lambda_x M_\infty$. Hence $\lambda_x M$ extends $\lambda_x M_\infty$. If $\lambda_x M$ does not derive a non-computational generalization, the claim follows immediately.

Case $(\forall^{\mathrm{nc}})^-$. Consider a proof pattern

$$\dfrac{\begin{array}{c}\Gamma\\ \mid U\\ \forall^{\mathrm{nc}}_x A(x) \qquad r\end{array}}{A(r)}\,(\forall^{\mathrm{nc}})^-$$

and let $\Delta \Rightarrow C(r)$ be any extension of its sequent $\Gamma \Rightarrow A(r)$. The induction hypothesis for U for the extension $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C(x)$ produces a decoration M_∞ of U whose sequent extends $\Delta \Rightarrow \forall^{\mathrm{nc}}_x C(x)$. Then apply $(\forall^{\mathrm{nc}})^-$ or $\forall^-$, whichever is appropriate, to obtain the required $M_\infty r$.

For (b) consider a decoration Mr of Ur whose sequent $\mathrm{Seq}(Mr)$ extends the given extension $\Delta \Rightarrow C(r)$ of $\Gamma \Rightarrow A(r)$. Then M extends M_∞ by induction hypothesis for U, and hence Mr extends $M_\infty r$. □

We illustrate the effects of decoration on a simple example involving implications. Consider $A \to B \to A$ with the trivial proof $M := \lambda^A_{u_1}\lambda^B_{u_2} u_1$. Clearly only the first implication must transport possible computational content. To "discover" this by means of the decoration algorithm we specify as extension of $\mathrm{Seq}(\mathrm{P}(M))$ the formula $A \to^{\mathrm{nc}} B \to^{\mathrm{nc}} A$. The algorithm then returns a proof of $A \to B \to^{\mathrm{nc}} A$.

6.2. Applications

6.2.1. List reversal: decoration of the weak existence proof. Recall the weak (or classical) existence proof for list reversal in 5.4.1. We apply the general method of decorating proofs in this example. First we present our proof in more detail, particularly by writing proof trees with formulas. Recall that we essentially use list induction. The full derivation M is obtained from

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{A([]) \to \forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(v))} \quad \begin{array}{l}\mid M_B\\ A([])\end{array}}{\forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(v)} \quad \begin{array}{l}\mid M_S\\ \forall_{x,v_2}(A(v_2) \to A(xv_2))\end{array}}{\forall_{v_1}(v_1 v = v \to \forall_w \neg^{\exists} R v_1 w) \quad (= A(v))}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{CompatRev}\quad R\quad v\quad v_1\quad v}}{v_1 =^{\mathrm{d}} v \to \forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w} \qquad \dfrac{[u_1\colon v_1\,[] = v]}{\mid N_1}}{v_1\,[] =^{\mathrm{d}} v}}{\forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w} \qquad \dfrac{\dfrac{}{\exists^+\quad R\quad v}}{\forall_w \neg^{\exists} Rvw}}{\dfrac{\dfrac{\forall_w \neg^{\exists} Rv_1 w}{v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w} \to^+ u_1}{\forall_{v_1}(v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w) \quad (= A([]))}}$$

FIGURE 1. Base derivation M_B

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[u_0\colon A(v_2)] \quad v_1 x}}{(v_1 x)v_2 = v \to \forall_w \neg^{\exists} R(v_1 x, w)} \quad \dfrac{}{[u_1\colon v_1(xv_2) = v]}}{\forall_w \neg^{\exists} R(v_1 x, w)} \quad \dfrac{}{xw} \qquad \dfrac{}{[u_2\colon Rv_1 w]}}{\neg^{\exists} R(v_1 x, xw)} \qquad \dfrac{}{\mid N_2}}{R(v_1 x, xw)}}{\dfrac{\exists_w Rvw}{\neg^{\exists} Rv_1 w} \to^+ u_2}}{\dfrac{\forall_w \neg^{\exists} Rv_1 w}{v_1(xv_2) = v \to \forall_w \neg^{\exists} Rv_1 w} \to^+ u_1}}{\dfrac{\forall_{v_1}(v_1(xv_2) = v \to \forall_w \neg^{\exists} Rv_1 w) \;\; (= A(xv_2))}{A(v_2) \to A(xv_2)}}}{\forall_{x,v_2}(A(v_2) \to A(xv_2))} \to^+ u_0$$

FIGURE 2. Step derivation M_S

by applying it to [], Truth, [] and InitR and finally introducing $\forall_v$. Here

$$\begin{aligned}
&\text{Ind}\colon && \forall^{\mathrm{nc}}_{v,R} \forall_w (A([]) \to \forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(w)),\\
&A(v_2) := && \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg^{\exists} Rv_1 w),\\
&\neg^{\exists} B := && B \to \exists_w Rvw.
\end{aligned}$$

The end formula then is $\forall_v \exists_w Rvw$. We have used the base derivation M_B in Figure 1 with N_1 involving EqToEqD: $\forall_{v,w}(v = w \to v =^{\mathrm{d}} w)$, and

$$\begin{aligned}
&\text{CompatRev}\colon && \forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall_w \neg^{\exists} Rv_2 w \to \forall_w \neg^{\exists} Rv_1 w)\\
&\exists^+\colon && \forall^{\mathrm{nc}}_{R,v} \forall_w \neg^{\exists} Rvw.
\end{aligned}$$

We have also used the step derivation M_S in Figure 2 with N_2 involving the

assumption GenR: $\forall_{v,w,x}(Rvw \to R(vx, xw))$.

We now apply the decoration algorithm. Notice that the sequent or our derivation consists of the context

$$\text{InitR: } R([],[]) \qquad \text{GenR: } \forall_{v,w,x}(Rvw \to R(vx, xw))$$

and the end formula $\forall_v \exists_w Rvw$. Among the axioms used, the only ones in c.r. parts are list induction, CompatRev and $\exists^+$. If we now form the proof pattern as defined above, we obtain a clash at the list induction axiom. Recall that it is given by its uninstantiated formula

$$\forall_v(X([]) \to \forall_{x,v_2}(X(v_2) \to X(xv_2)) \to X(v))$$

and the predicate substitution $X \mapsto \{\, v \mid A(v) \,\}$. When forming the proof pattern, $A(v_2)$ is changed into $\hat{\mathrm{A}}(v_2) := \forall^{\mathrm{nc}}_{v_1}(v_1v_2 = v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w)$, but the uninstantiated formula is not touched. The clash then consists in the fact that the conclusion of the decorated induction axiom

$$\forall^{\mathrm{nc}}_{v,R}\forall_w(\hat{\mathrm{A}}([]) \to \forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to \hat{\mathrm{A}}(xv_2)) \to \hat{\mathrm{A}}(w)),$$

is a proper extension of what is in the proof pattern:

$$\forall^{\mathrm{nc}}_{v,R,w}(\hat{\mathrm{A}}([]) \to^{\mathrm{nc}} \forall^{\mathrm{nc}}_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2)) \to^{\mathrm{nc}} \hat{\mathrm{A}}(w)).$$

The decoration algorithm now replaces the latter by the former. Similarly in M_B the conclusions of the decorated axioms CompatRev and $\exists^+$

$$\forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_2w \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w)$$
$$\forall^{\mathrm{nc}}_{R,v}\forall_w(Rvw \to \exists_w Rvw)$$

are proper extensions of what is in the proof pattern

$$\forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_2w \to^{\mathrm{nc}} \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w)$$
$$\forall^{\mathrm{nc}}_{R,v,w}(Rvw \to \exists_w Rvw)$$

and the decoration algorithm replaces the latter by the former. But now the end formula $\forall_w \neg^{\exists} Rvw$ of the $\exists^+$-derivation is a proper extension of the premise of the conclusion $\forall^{\mathrm{nc}}_w \neg^{\exists} Rv_2w \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w)$ of the CompatRev-derivation. This requires us to go back into the CompatRev-derivation and change the predicate substitution $X \mapsto \{\, v \mid \hat{\mathrm{A}}(v) \,\}$ to $X \mapsto \{\, v \mid A'(v) \,\}$ with $A'(v_2) := \forall^{\mathrm{nc}}_{v_1}(v_1v_2 = v \to \forall_w \neg^{\exists} Rv_1w)$. Thus we obtain the derivation in Figure 3.

But now we have a clash where M'_B is used:

$$\dfrac{\dfrac{\text{Ind}\quad v\quad R\quad v}{\hat{\mathrm{A}}([]) \to \forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to \hat{\mathrm{A}}(xv_2)) \to \hat{\mathrm{A}}(v)} \quad |M'_B \qquad A'([])}{\forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2)) \to^{\mathrm{nc}} \hat{\mathrm{A}}(v)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{v_1 =^{\mathrm{d}} v \to \forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w}\,\mathrm{CompatRev}\quad R\quad v\quad v_1\quad v \qquad \dfrac{[u_1 \colon v_1\,[] = v]}{\mid N_1}}{\forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w}\; v_1\,[] =^{\mathrm{d}} v \qquad \dfrac{\exists^{+}\quad R\quad v}{\forall_w \neg^{\exists} Rvw}}{\dfrac{\forall_w \neg^{\exists} Rv_1 w}{v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w} \to^{+} u_1}}{\forall^{\mathrm{nc}}_{v_1}(v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w) \quad (= A'([]))}$$

FIGURE 3. Base derivation M'_B

Thus we have to go again into the left hand derivation and change the predicate substitution $X \mapsto \{\, v \mid \hat{\mathrm{A}}(v) \,\}$ used in the induction axiom into $X \mapsto \{\, v \mid A'(v) \,\}$. This gives us $\forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)$.

Now the next clash appears where we used M_S: we have to change $\mathrm{P}(M_S)$ with end formula $\forall^{\mathrm{nc}}_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2))$ into a derivation M'_S of $\forall_{x,v_2}(A'(v_2) \to A'(xv_2))$. But this is easy, since no c.r. axioms are involved: just change $\forall^{\mathrm{nc}}_x$, $\forall^{\mathrm{nc}}_w$ everywhere into $\forall_x$, $\forall_w$.

Finally we obtain

$$\dfrac{\dfrac{\dfrac{}{A'([]) \to \forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v))}\,\mathrm{Ind}\quad v\quad R\quad v \qquad \dfrac{\mid M'_B}{A'([])}}{\forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)} \qquad \dfrac{\mid M'_S}{\forall_{x,v_2}(A'(v_2) \to A'(xv_2))}}{\forall^{\mathrm{nc}}_{v_1}(v_1 v = v \to \forall_w \neg^{\exists} Rv_1 w) \quad (= A'(v))}$$

Applying this it to [], Truth, [] and InitR and finally introducing $\forall_v$ gives us a decorated derivation of formula $\forall_v \exists_w Rvw$. The difference is that induction is now used w.r.t. the formula $A'(v_2) := \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall_w \neg^{\exists} Rv_1 w)$ with $\forall^{\mathrm{nc}}_{v_1}$ rather than $\forall_{v_1}$.

The extracted term `neterm` then is

```
[R,v](Rec list nat=>list nat=>list nat)v([v0]v0)
  ([x,v0,f,v1]f(x::v1))(Nil nat)
```

with `f` a variable for unary functions on lists. To run this algorithm one has to normalize the term obtained by applying `neterm` to a list:

```
(pp (nt (mk-term-in-app-form neterm (pt "1::2::3::4:"))))
```

The returned value is the reverted list `4::3::2::1:`. This time, the underlying algorithm defines an auxiliary function g by

$$g([], w) := w, \qquad g(x :: v, w) := g(v, x :: w)$$

and gives the result by applying g to the original list and []. In conclusion, we have obtained (by machine extraction from an automated decoration of a weak existence proof) the standard linear algorithm for list reversal, with its use of an accumulator.

6.2.2. Fibonacci numbers. An application of decoration occurs when one derives double induction

$$\forall_n(Qn \to Q(Sn) \to Q(S(Sn))) \to \forall_n(Q0 \to Q1 \to Qn)$$

in *continuation passing style*, i.e., not directly, but using as an intermediate assertion (proved by induction)

$$\forall_{n,m}((Qn \to Q(Sn) \to Q(n+m)) \to Q0 \to Q1 \to Q(n+m)).$$

After decoration, the formula becomes

$$\forall_n \forall_m^{\mathrm{nc}}((Qn \to Q(Sn) \to Q(n+m)) \to Q0 \to Q1 \to Q(n+m)).$$

This can be applied to obtain a continuation based tail recursive definition of the Fibonacci function, from a proof of its totality. Let G be the (n.c.) graph of the Fibonacci function, defined by the clauses

$$G(0,0), \quad G(1,1),$$
$$\forall_{n,v,w}(G(n,v) \to G(Sn,w) \to G(S(Sn), v+w)).$$

From these assumptions one can easily derive

$$\forall_n \exists_v G(n,v),$$

using double induction (proved in continuation passing style). The term extracted from this proof is

```
[n](Rec nat=>nat=>(nat=>nat=>nat)=>nat=>nat=>nat)n([n0,k]k)
 ([n0,p,n1,k]p(Succ n1)([n2,n3]k n3(n2+n3)))
```

applied to `0`, `([n0,n1]n0)`, `0` and `1`.

An unclean aspect of this term is that the recursion operator has value type

```
nat=>(nat=>nat=>nat)=>nat=>nat=>nat
```

rather than `(nat=>nat=>nat)=>nat=>nat=>nat`, which would correspond to an iteration. However, we can repair this by decoration. After (automatic) decoration of the proof, the extracted term becomes

```
[n](Rec nat=>(nat=>nat=>nat)=>nat=>nat=>nat)n([k]k)
 ([n0,p,k]p([n1,n2]k n2(n1+n2)))
```

applied to `([n0,n1]n0)`, `0` and `1` (`k`, `p` are variables of type `nat=>nat=>nat` and `(nat=>nat=>nat)=>nat=>nat=>nat`, respectively.) This is iteration in continuation passing style: the functional F recursively defined by

$$F(0,k) := k$$

$$F(n+1,k) := F(n, \lambda_{n,n'}(k(n', n+n')))$$

is applied to n, the left projection $\lambda_{n_0,n_1} n_0$ and $0, 1$.

6.2.3. Proof transformations. In the next two examples we allow the decoration algorithm to substitute an auxiliary lemma used in the proof by a lemma that we specify explicitly. The algorithm will verify if the lemma passed to it as an argument is fitting and if this is the case, it will replace the lemma used in the original proof by the specified one. If not, the initial lemma is kept. This will allow for a certain control over the computational content, as shown by the following examples.

6.2.3.1. *Avoiding factorization.* Our first example is an elaboration of Constable's idea described in the introduction. Let Pn mean "n is prime". Consider

$$\begin{aligned} &\forall_n(Pn \vee^{\mathrm{r}} \exists^{\mathrm{d}}_{m,k>1}(n = mk)) \quad \text{factorization,} \\ &\forall_n(Pn \vee^{\mathrm{u}} \exists^{\mathrm{d}}_{m,k>1}(n = mk)) \quad \text{prime number test.} \end{aligned}$$

Euler's φ-function has the properties

$$\begin{cases} \varphi(n) = n-1 & \text{if } Pn, \\ \varphi(n) < n-1 & \text{if } n \text{ is composed.} \end{cases}$$

Suppose that somewhat foolishly we have used factorization and these properties to obtain a proof of

$$\forall_n(\varphi(n) = n-1 \vee^{\mathrm{u}} \varphi(n) < n-1).$$

Our goal is to get rid of the expensive factorization algorithm in the computational content, via decoration.

The decoration algorithm arrives at the factorization theorem

$$\forall_n(Pn \vee^{\mathrm{r}} \exists^{\mathrm{d}}_{m,k>1}(n = mk))$$

with the decorated formula

$$\forall_n(Pn \vee^{\mathrm{u}} \exists^{\mathrm{d}}_{m,k>1}(n = mk)).$$

Since the prime number test can be considered instead of the factorization lemma, we can specify that the decoration algorithm should try to replace the former by the latter. In case this is possible, a new proof is constructed, using the the prime number test lemma. Should this fail, the factorization lemma is kept. As it turns out in this case, the replacement is possible.

In the Minlog implementation the difference is clearly visible. `cFact` denotes the computational content of the factorization lemma `Fact` (i.e., the factorization algorithm), and `cPTest` the computational content of the

lemma `PTest` expressing the prime number test. The extract from the original proof involves computing `(cFact n)`, i.e., factorizing the argument, whereas after decoration the prime number test `cPTest` suffices.

```
(define eterm (proof-to-extracted-term nproof))
(define neterm (rename-variables (nt eterm)))
(pp neterm)
;; [n][if (cFact n) True ([algC]False)]

(define decnproof (fully-decorate nproof "Fact" "PTest"))
(pp (nt (proof-to-extracted-term decnproof)))
;; cPTest
```

6.2.3.2. *Maximal scoring segment.* The second example is due to Bates and Constable (1985), and deals with the "maximal scoring segment" (MSS) problem. Let X be a set with a linear ordering $\preceq$, and consider an infinite sequence $f\colon \mathbf{N} \to X$ of elements of X. Assume further that we have a function $M\colon (\mathbf{N} \to X) \to \mathbf{N} \to \mathbf{N} \to X$ such that $M(f,i,k)$ "measures" the segment $f(i),\dots,f(k)$. The task is to find a segment determined by $i \le k \le n$ such that its measure is maximal. To simplify the formalization let us consider M and f fixed and define $\mathrm{seg}(i,k) := M(f,i,k)$.

Such a problem appears e.g. in computational biology, when one wants to compute regions with high G,C content in DNA. Let

$$
\begin{aligned}
&X := \{G,C,A,T\},\\
&g\colon \mathbf{N} \to X \quad \text{(gene)},\\
&f\colon \mathbf{N} \to \mathbf{Z}, \quad f(i) := \begin{cases} 1 & \text{if } g(i) \in \{G,C\},\\ -1 & \text{if } g(i) \in \{A,T\}, \end{cases}\\
&\mathrm{seg}(i,k) = f(i) + \cdots + f(k).
\end{aligned}
$$

Of course we can simply solve this problem by trying all possibilities; these are $O(n^2)$ many. The first proof to be given below corresponds to this general claim. Then we will show that for a more concrete problem with the sum $x_i + \cdots + x_k$ as measure the proof can be simplified, using monotonicity of the sum at an appropriate place. From this simplified proof one can extract a better algorithm, which is linear rather than quadratic. Our goal is to achieve this effect by decoration.

Let us be more concrete. The original specification is to find a maximal segment $x_i,\dots,x_k$, i.e.,

$$\forall_n \exists_{i\le k\le n} \forall_{i'\le k'\le n} (\mathrm{seg}(i',k') \preceq \mathrm{seg}(i,k)).$$

A special case is to find a maximal *end* segment

$$\forall_n \exists_{j\le n} \forall_{j'\le n} (\mathrm{seg}(j',n) \preceq \mathrm{seg}(j,n)).$$

We provide two lemmata proving the existence of a maximal end segment for $n+1$. The first one is

$$\mathtt{L}\colon \forall_n \exists_{j\le n+1} \forall_{j'\le n+1}(\mathrm{seg}(j',n+1) \preceq \mathrm{seg}(j,n+1)).$$

Its proof introduces an auxiliary variable m and proceeds by induction on m, with n a parameter:

$$\forall_n^{\mathrm{nc}} \forall_{m\le n+1} \exists_{j\le n+1} \forall_{j'\le m}(\mathrm{seg}(j',n+1) \preceq \mathrm{seg}(j,n+1)).$$

The second one is

$$\mathtt{LMon}\colon \forall_n^{\mathrm{nc}}(\mathtt{ES}_n \to \mathtt{Mon} \to \exists_{j\le n+1} \forall_{j'\le n+1}(\mathrm{seg}(j',n+1) \preceq \mathrm{seg}(j,n+1))).$$

It has as additional assumptions the existence $\mathtt{ES}_n$ of a maximal end segment for n

$$\mathtt{ES}_n\colon \exists_{j\le n} \forall_{j'\le n}(\mathrm{seg}(j',n) \preceq \mathrm{seg}(j,n))$$

and the assumption Mon of monotonicity of seg

$$\mathtt{Mon}\colon \mathrm{seg}(i,k) \preceq \mathrm{seg}(j,k) \to \mathrm{seg}(i,k+1) \preceq \mathrm{seg}(j,k+1).$$

The proof proceeds by cases on $\mathrm{seg}(j,n+1) \preceq \mathrm{seg}(n+1,n+1)$. If $\preceq$ holds, take $n+1$, else the previous j.

We now prove the existence of a maximal segment by induction on n, simultaneously with the existence of a maximal end segment.

$$\mathtt{MaxSegMon}\colon \forall_n(\exists_{i\le k\le n} \forall_{i'\le k'\le n}(\mathrm{seg}(i',k') \preceq \mathrm{seg}(i,k)) \land \exists_{j\le n} \forall_{j'\le n}(\mathrm{seg}(j',n) \preceq \mathrm{seg}(j,n)))$$

In the step, we compare the maximal segment i,k for n with the maximal end segment for $j,n+1$ provided separately. If $\preceq$ holds, take the new i,k to be $j,n+1$. Else take the old i,k.

Depending on how the existence of a maximal end segment was proved, we obtain a quadratic or a linear algorithm. If we consider the first proof involving induction on the auxiliary variable m, we obtain a quadratic algorithm as follows.

```
(define eterm (proof-to-extracted-term
                (theorem-name-to-proof "MaxSegMon")))
(add-var-name "ijk" (py "nat@@nat@@nat"))
(define neterm (rename-variables (nt eterm)))
(pp neterm)
```

The result is

```
[le,seg,n](Rec nat=>nat@@nat@@nat)n(0@0@0)
 ([n0,ijk]
   [if (le(seg left ijk right right ijk)
         (seg((cL alpha)le seg n0(Succ n0))(Succ n0)))
     ((cL alpha)le seg n0(Succ n0))
```

```
    (left ijk)]@
  (cL alpha)le seg n0(Succ n0)@
  [if (le(seg left ijk right right ijk)
        (seg((cL alpha)le seg n0(Succ n0))(Succ n0)))
    (Succ n0)
    (right right ijk)])
```

The computational content of L involves an additional recursion, since L was proved by induction on m.

```
(pp (rename-variables (nt
 (proof-to-extracted-term (theorem-name-to-proof "L")))))

[le,seg,n,n0](Rec nat=>nat)n0 0
 ([n1,n2][if (le(seg n2(Succ n))(seg(Succ n1)(Succ n)))
             (Succ n1)
             n2])
```

The two nested recursions give a quadratic algorithm.

Now how could the better proof be found by decoration? We have

$\mathtt{L}\colon \forall_n \exists_{j\le n+1} \forall_{j'\le n+1}(\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1))$,

$\mathtt{LMon}\colon \forall^{\mathrm{nc}}_n(\mathtt{ES}_n \to \mathtt{Mon} \to \exists_{j\le n+1} \forall_{j'\le n+1}(\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1)))$.

The decoration algorithm arrives at L with

$$\exists_{j\le n+1} \forall_{j'\le n+1}(\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1)).$$

LMon fits as well, its assumptions $\mathtt{ES}_n$ and Mon are in the context, and it has the less extended $\forall^{\mathrm{nc}}_n$ rather than $\forall_n$, hence is preferred.

In Minlog we can do the decoration by executing

```
(define decproof
 (decorate (theorem-name-to-proof "MaxSegMon") "L" "LMon"))
(define eterm (proof-to-extracted-term decproof))
(pp (rename-variables (nt eterm)))

[le,seg,n](Rec nat=>nat@@nat@@nat)n(0@0@0)
 ([n0,ijk]
   [if (le(seg left ijk right right ijk)
          (seg((cLMon alpha)le seg n0 left right ijk)
              (Succ n0)))
     ((cLMon alpha)le seg n0 left right ijk)
     (left ijk)]@
   (cLMon alpha)le seg n0 left right ijk@
   [if (le(seg left ijk right right ijk)
          (seg((cLMon alpha)le seg n0 left right ijk)
```

```
               (Succ n0)))
     (Succ n0)
     (right right ijk)])
```

The computational content `cLMon` of `LMon` is

```
(pp (rename-variables
     (nt (proof-to-extracted-term
          (theorem-name-to-proof "LMon")))))

[le,seg,n,n0][if (le(seg n0(Succ n))(seg(Succ n)(Succ n)))
                 (Succ n)
                 n0]
```

which does not involve recursion any more. Hence after decoration we have a linear algorithm.

APPENDIX A

Denotational semantics: proofs

We show that every closed term M has a computable functional $[\![M]\!]$ as its denotation.

A.1. 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 P, Q (term for short) we mean a term built from variables x, y, z and constructors C by application. A *substitution* is a finite set $\vartheta = \{P_1/x_1, \ldots, P_n/x_n\}$ of pairs of variables and terms, such that $x_i \neq x_j$ for $i \neq j$, and $P_i \neq x_i$ for all i. An element P_i/x_i of ϑ is called a *binding* (of x_i to P_i). By $P\vartheta$ we denote the result of simultaneously replacing each variable x_i in P by P_i, and call $P\vartheta$ the *instance* of P induced by ϑ. We shall use ϑ, η, ζ for substitutions. Let ε be the empty substitution. For given substitutions

$$\vartheta = \{P_1/x_1, \ldots, P_n/x_n\}$$
$$\eta = \{Q_1/y_1, \ldots, Q_m/y_m\},$$

the *composition* $\vartheta\eta$ of ϑ and η is the substitution obtained by deleting in the set

$$\{P_1\eta/x_1, \ldots, P_n\eta/x_n, Q_1/y_1, \ldots, Q_m/y_m\}$$

all bindings $P_i\eta/x_i$ such that $P_i\eta = x_i$, and also all bindings Q_j/y_j such that $y_j \in \{x_1, \ldots, x_n\}$. 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 $(P\vartheta)\eta = P(\vartheta\eta)$, and that composition is associative.

We now come to the unification problem. By this we mean the question whether for two given terms P, Q there is a substitution ϑ "unifying" the two terms, i.e., with the property $P\vartheta = Q\vartheta$.

Let E denote finite equation systems, i.e., multisets

$$\{P_1 = Q_1, \ldots, P_n = Q_n\}$$

of equations between terms (more precisely pairs of terms). Consider $\{\bot\}$ as a (contradictory) equation system. A substitution ϑ *unifies* E, if for every equation $P = Q$ in E we have $P\vartheta = Q\vartheta$; no ϑ unifies $\{\bot\}$. 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 (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. Assume that ϑ is a unifier of E.

$\rightarrow$. Let ϑ be an idempotent mgu of E, and assume that η is a unifier of E. Since ϑ is a mgu of E, we have $\eta = \vartheta\zeta$ for some substitution ζ. Hence $\eta = \vartheta\zeta = \vartheta\vartheta\zeta = \vartheta\eta$.

$\leftarrow$. Assume that $\eta = \vartheta\eta$ for all unifiers η of E. Now let η be a unifier of E. Then $\eta \leq \vartheta$; therefore ϑ is a mgu. Since ϑ is a unifier, by assumption we have $\vartheta = \vartheta\vartheta$. □

DEFINITION (Unification algorithm). $E \mapsto_\vartheta E'$ is defined by

(a) $\{P = x\} \cup E \mapsto_\varepsilon \{x = P\} \cup E$, if P is not a variable.
(b) $\{x = x\} \cup E \mapsto_\varepsilon E$.
(c) $\{\mathrm{C}P_1 \dots P_n = \mathrm{C}Q_1 \dots Q_n\} \cup E \mapsto_\varepsilon \{P_1 = Q_1, \dots P_n = Q_n\} \cup E$.
(d) $\{\mathrm{C}P_1 \dots P_n = \mathrm{C}'Q_1 \dots Q_n\} \cup E \mapsto_\varepsilon \{\bot\}$ if $\mathrm{C} \neq \mathrm{C}'$.
(e) $\{x = P, P_1(x) = Q_1(x), \dots, P_n(x) = Q_n(x)\} \mapsto_{\{P/x\}}$
$\{P_1(P) = Q_1(P), \dots P_n(P) = Q_n(P)\}$ if $x \notin \mathrm{FV}(P)$.
(f) $\{x = P\} \cup E \mapsto_\varepsilon \{\bot\}$, if $x \in \mathrm{FV}(P)$ and $P \neq x$.

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 $\{P/x\}\eta'$ is a unifier of E.

Let η be a unifier of E. Then $x\eta = P\eta$, hence $\eta = \{P/x\}\eta$ (since both substitutions coincide on all variables), and moreover

$$P_i\{P/x\}\eta = P_i\eta = Q_i\eta = Q_i\{P/x\}\eta.$$

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 $\eta = \vartheta_2 \cdots \vartheta_n \eta$ 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 $E_{n+1} = \{\bot\}$. 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. We first show termination using the lexicographic ordering of $\mathbf{N}^3$. To every $E = \{P_1 = Q_1, \dots, P_n = Q_n\}$ assign a triple $(n_1, n_2, n_3) \in \mathbf{N}^3$ by

$n_1 :=$ number of variables in E,

$n_2 :=$ number of occurrences of variables and constructors in E,

$n_3 :=$ number of equations $P = x$ in E such that P is not a variable.

In every step $E \mapsto_\vartheta E'$ the assigned triple decreases w.r.t. the lexicographic ordering of $\mathbf{N}^3$. This can be verified easily by considering the different cases: For (a), n_1, n_2 remain unchanged, and n_3 decreases. For (b), (c), (d) and (f), n_2 decreases, and n_1 does not increase. For (e), n_1 decreases. Hence our given sequence $E_1 \mapsto_{\vartheta_1} E_2 \mapsto_{\vartheta_2} \dots$ terminates with $E_n \mapsto_{\vartheta_n} E_{n+1}$. Then it is easy to see that either $E_{n+1} = \emptyset$ or $E_{n+1} = \{\bot\}$.

Case $E_{n+1} = \emptyset$. By the Corollary $\vartheta_1 \dots \vartheta_n$ is a unifier of E, and by the Proposition we have $\eta = \vartheta_1 \dots \vartheta_n \eta$ for every unifier η of E. Hence by the characterization of idempotent mgu's $\vartheta_1 \dots \vartheta_n$ is an idempotent mgu of E.

Case $E_{n+1} = \{\bot\}$. Then by the proposition E is not unifiable. □

A.2. Ideals as denotations of terms

Recall the definition of the relation $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$ in Section 2.3

The *height* of a derivation of $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$ 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. The following transformations preserve D-height:

$$\vec{V} \vdash \vec{U} \to (\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!] \to (\vec{V}, a) \in [\![\lambda_{\vec{x}} M]\!], \tag{38}$$

$$(\vec{U}, V, a) \in [\![\lambda_{\vec{x},y} M]\!] \leftrightarrow (\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!] \quad \text{if } y \notin \mathrm{FV}(M), \tag{39}$$

$$(\vec{U}, V, a) \in [\![\lambda_{\vec{x},y}(My)]\!] \leftrightarrow (\vec{U}, V, a) \in [\![\lambda_{\vec{x}} M]\!] \quad \text{if } y \notin \mathrm{FV}(M), \tag{40}$$

$$(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}(M(\vec{P}(\vec{y})))]\!] \leftrightarrow (\vec{U}, \vec{P}(\vec{V}), a) \in [\![\lambda_{\vec{x},\vec{z}}(M(\vec{z}))]\!]. \tag{41}$$

Proof. (38) and (39) are both proved by easy inductions on the respective derivations.

(40). Assume $(\vec{U}, V, a) \in [\![\lambda_{\vec{x},y}(My)]\!]$. By (A) we then have W such that $(\vec{U}, V, W) \subseteq [\![\lambda_{\vec{x},y} y]\!]$ (i.e., $V \vdash W$) and $(\vec{U}, V, W, a) \in [\![\lambda_{\vec{x},y} M]\!]$. By (38) from the latter we obtain $(\vec{U}, V, V, a) \in [\![\lambda_{\vec{x},y} M]\!]$. Now since $y \notin \mathrm{FV}(M)$, (39) yields $(\vec{U}, V, a) \in [\![\lambda_{\vec{x}} M]\!]$, as required. Conversely, assume $(\vec{U}, V, a) \in [\![\lambda_{\vec{x}} M]\!]$. Since $y \notin \mathrm{FV}(M)$, (39) yields $(\vec{U}, V, V, a) \in [\![\lambda_{\vec{x}} M]\!]$. Clearly we have $(\vec{U}, V, V) \subseteq [\![\lambda_{\vec{x},y} y]\!]$. Hence by (A) $(\vec{U}, V, a) \in [\![\lambda_{\vec{x},y}(My)]\!]$, as required. Notice that the D-height did not change in these transformations.

(41). By induction on $\vec{P}$, with a side induction on M. We distinguish cases on M. The cases x_i, C and D are follow immediately from (39). In case MN the following are equivalent by induction hypothesis:

$$(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}((MN)(\vec{P}(\vec{y})))]\!]$$
$$\exists_W((\vec{U}, \vec{V}, W) \subseteq [\![\lambda_{\vec{x},\vec{y}}(N(\vec{P}(\vec{y})))]\!] \wedge (\vec{U}, \vec{V}, W, a) \in [\![\lambda_{\vec{x},\vec{y}}(M(\vec{P}(\vec{y})))]\!])$$
$$\exists_W((\vec{U}, \vec{P}(\vec{V}), W) \subseteq [\![\lambda_{\vec{x},\vec{y}}(N(\vec{z}))]\!] \wedge (\vec{U}, \vec{P}(\vec{V}), W, a) \in [\![\lambda_{\vec{x},\vec{y}}(M(\vec{z}))]\!])$$
$$(\vec{U}, \vec{P}(\vec{V}), a) \in [\![\lambda_{\vec{x},\vec{y}}((MN)(\vec{z}))]\!].$$

The final case is where M is z_i. Then we have to show

$$(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}(P(\vec{y}))]\!] \leftrightarrow P(\vec{V}) \vdash a.$$

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 $(\mathrm{C}\vec{Q})(\vec{y})$ the following are equivalent, using the induction hypothesis for $\vec{Q}(\vec{y})$

$$(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}((\mathrm{C}\vec{Q})(\vec{y}))]\!]$$
$$(\vec{U}, \vec{V}, a) \in [\![\lambda_{\vec{x},\vec{y}}(\mathrm{C}\vec{Q}(\vec{y}))]\!]$$
$$(\vec{U}, \vec{Q}(\vec{V}), a) \in [\![\lambda_{\vec{x},\vec{u}}(\mathrm{C}\vec{u})]\!]$$
$$(\vec{U}, \vec{Q}(\vec{V}), a) \in [\![\lambda_{\vec{x}}\mathrm{C}]\!] \quad \text{by (40)}$$
$$\exists_{\vec{a^*}}(a = \mathrm{C}\vec{a^*} \wedge \vec{Q}(\vec{V}) \vdash \vec{a^*})$$

$$\mathrm{C}\vec{Q}(\vec{V}) \vdash a. \qquad \square$$

Let $\sim$ denote the equivalence relation on formal neighborhoods generated by entailment, i.e., $U \sim V$ means $(U \vdash V) \wedge (V \vdash U)$.

(42) 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 $\mathrm{C}\vec{P}(\vec{x}\,)$. Since $U \vdash \mathrm{C}\vec{P}(\vec{V})$ there is a $\vec{b_0^*}$ such that $\mathrm{C}\vec{b_0^*} \in U$. Let

$$U_i := \{\, a \mid \exists_{\vec{a^*}}(\mathrm{C}\vec{a^*} \in U \wedge a = a_i^*) \,\}.$$

For the constructor pattern $\mathrm{C}\vec{x}$ consider $\mathrm{C}\vec{U}$. By definition

$$\mathrm{C}\vec{U} = \{\, \mathrm{C}\vec{a^*} \mid a_i^* \in U_i \text{ if } U_i \neq \emptyset, \text{ and } a_i^* = * \text{ otherwise} \,\}.$$

We first show $U \sim \mathrm{C}\vec{U}$. Assume $\mathrm{C}\vec{a^*} \in \mathrm{C}\vec{U}$. For each i, if $U_i \neq \emptyset$, then there is an $\vec{a_i^*}$ such that $\mathrm{C}\vec{a_i^*} \in U$ and $a_{ii}^* = a_i^*$, and if $U_i = \emptyset$ then $a_i^* = *$. Hence

$$U \supseteq \{\, \mathrm{C}\vec{a_i^*} \mid U_i \neq \emptyset \,\} \cup \{\mathrm{C}\vec{b_0^*}\} \vdash \mathrm{C}\vec{a^*}.$$

Conversely assume $\mathrm{C}\vec{a^*} \in U$. We define $\mathrm{C}\vec{b^*} \in \mathrm{C}\vec{U}$ 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 $\{\mathrm{C}\vec{b^*}\} \vdash \mathrm{C}\vec{a^*}$.

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 $U \sim \mathrm{C}\vec{U} \sim \mathrm{C}\vec{P}(\vec{W})$. $\square$

Lemma (Unification). *If $\vec{P}_1(\vec{V}_1) \sim \cdots \sim \vec{P}_n(\vec{V}_n)$, then $\vec{P}_1, \ldots, \vec{P}_n$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that*

$$(\vec{P}_1\vartheta)(\vec{W}) = \cdots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1) \sim \cdots \sim \vec{P}_n(\vec{V}_n).$$

Proof. Assume $\vec{P}_1(\vec{V}_1) \sim \cdots \sim \vec{P}_n(\vec{V}_n)$. Then $\vec{P}_1(\vec{V}_1), \ldots, \vec{P}_n(\vec{V}_n)$ are componentwise consistent and hence $\vec{P}_1, \ldots, \vec{P}_n$ are unifiable with a most general unifier ϑ. We now proceed by induction on $\vec{P}_1, \ldots, \vec{P}_n$. If they are either all empty or all variables the claim is trivial. In the case $(\vec{P}_1, P_1), \ldots, (\vec{P}_n, P_n)$ it follows from the linearity condition on variables that a most general unifier of $(\vec{P}_1, P_1), \ldots, (\vec{P}_n, P_n)$ is the union of most general unifiers of $\vec{P}_1, \ldots, \vec{P}_n$ and of $P_1, \ldots, P_n$. Hence the induction hypothesis applies. In the case $\mathrm{C}\vec{P}_1, \ldots, \mathrm{C}\vec{P}_n$ the assumption $\mathrm{C}\vec{P}_1(\vec{V}_1) \sim \cdots \sim \mathrm{C}\vec{P}_n(\vec{V}_n)$ implies $\vec{P}_1(\vec{V}_1) \sim \cdots \sim \vec{P}_n(\vec{V}_n)$ and hence again the induction hypothesis applies. The remaining case is when some are variables and the other ones of the form $\mathrm{C}\vec{P}_i$, say $x, \mathrm{C}\vec{P}_2, \ldots, \mathrm{C}\vec{P}_n$. By assumption

$$V_1 \sim \mathrm{C}\vec{P}_2(\vec{V}_2) \sim \cdots \sim \mathrm{C}\vec{P}_n(\vec{V}_n).$$

By induction hypothesis we obtain the required $\vec{W}$ such that

$$(\vec{P}_2\vartheta)(\vec{W}) = \cdots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_2(\vec{V}_2) \sim \cdots \sim \vec{P}_n(\vec{V}_n). \qquad \square$$

We need a final preparation before we can tackle consistency of $[\![\lambda_{\vec{x}}M]\!]$. The information systems $\boldsymbol{C}_\rho$ 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 $\boldsymbol{A} = (A, \mathrm{Con}, \vdash)$ is *coherent* if it satisfies: $U \subseteq A$ is consistent if and only if all of its two-element subsets are.

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

PROOF. Let $\boldsymbol{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\boldsymbol{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems, and consider $\{(U_1, b_1), \ldots, (U_n, b_n)\} \subseteq \mathrm{Con}_A \times B$. Assume

$$\forall_{1\le i<j\le n}(\{(U_i, b_i), (U_j, b_j)\} \in \mathrm{Con}).$$

We have to show $\{(U_1, b_1), \ldots, (U_n, b_n)\} \in \mathrm{Con}$. Let $I \subseteq \{1, \ldots, n\}$ and $\bigcup_{i\in I} U_i \in \mathrm{Con}_A$. We must show $\{\, b_i \mid i \in I \,\} \in \mathrm{Con}_B$. Now since $\boldsymbol{B}$ is coherent by assumption, it suffices to show that $\{b_i, b_j\} \in \mathrm{Con}_B$ for all $i, j \in I$. So let $i, j \in I$. By assumption we have $U_i \cup U_j \in \mathrm{Con}_A$, and hence $\{b_i, b_j\} \in \mathrm{Con}_B$. $\square$

By a similar argument we can prove

LEMMA (Coherence). *The information systems $\boldsymbol{C}_\rho$ are all coherent.*

PROOF. By induction of the height $|U|$ of consistent finite sets of tokens in $\boldsymbol{C}_\rho$, as defined in parts (c) and (d) of the definition in 2.1.5. $\square$

LEMMA (Consistency). *$[\![\lambda_{\vec{x}}M]\!]$ is consistent.*

PROOF. Let $(\vec{U}_i, a_i) \in [\![\lambda_{\vec{x}}M]\!]$ 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 $(\vec{U}_1, a_1), (\vec{U}_2, a_2) \in [\![\lambda_{\vec{x}}x_i]\!]$, 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

$$\frac{\vec{V}_i \vdash \vec{a_i^*}}{(\vec{U}_i, \vec{V}_i, \mathrm{C}\vec{a_i^*}) \in [\![\lambda_{\vec{x}}\mathrm{C}]\!]}.$$

Assume $\vec{U}_1, \vec{V}_1$ and $\vec{U}_2, \vec{V}_2$ are componentwise consistent. The consistency of $\mathrm{C}\vec{a_1^*}$ and $\mathrm{C}\vec{a_2^*}$ follows from $\vec{V}_i \vdash \vec{a_i^*}$ and the consistency of $\vec{V}_1$ and $\vec{V}_2$.

Case (A). For $i = 1, 2$ we have

$$\frac{(\vec{U}_i, V_i, a_i) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}_i, V_i) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}_i, a_i) \in [\![\lambda_{\vec{x}}(MN)]\!]}.$$

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

$$\frac{(\vec{U}_i, \vec{V}_i, a_i) \in [\![\lambda_{\vec{x},\vec{y}_i} M_i(\vec{y}_i)]\!] \qquad \vec{W}_i \vdash \vec{P}_i(\vec{V}_i)}{(\vec{U}_i, \vec{W}_i, a_i) \in [\![\lambda_{\vec{x}} D]\!]}(D)$$

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 (42) 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 the unification lemma 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 (38) we have

$$(\vec{U}_i, (\vec{y}_i\vartheta)(\vec{W}), a_i) \in [\![\lambda_{\vec{x},\vec{y}_i} M_i(\vec{y}_i)]\!]$$

with lesser D-height. Now (41) gives

$$(\vec{U}_i, \vec{W}, a_i) \in [\![\lambda_{\vec{x},\vec{z}} M_i(\vec{y}_i)\vartheta]\!]$$

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 $(\vec{U}_1, \vec{W}, a_1), (\vec{U}_2, \vec{W}, a_2) \in [\![\lambda_{\vec{x},\vec{z}} M_1(\vec{y}_1)\vartheta]\!]$ implies the consistency of a_1 and a_2, as required. □

Lemma (Deductive closure). $[\![\lambda_{\vec{x}} M]\!]$ *is deductively closed, i.e., if* $W \subseteq [\![\lambda_{\vec{x}} M]\!]$ *and* $W \vdash (\vec{V}, b)$*, then* $(\vec{V}, b) \in [\![\lambda_{\vec{x}} M]\!]$.

Proof. By induction on the maximum of the D-heights and a side induction on the maximum of the heights of $W \subseteq [\![\lambda_{\vec{x}} M]\!]$. 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

$$\frac{U_i \vdash a}{(\vec{U}, a) \in [\![\lambda_{\vec{x}} x_i]\!]}.$$

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 $W = \{(\vec{U}_1, a_1), \ldots, (\vec{U}_n, a_n)\}$. For each $(\vec{U}_i, a_i) \in W$ there is U_i such that

$$\frac{(\vec{U}_i, U_i, a_i) \in [\![\lambda_{\vec{x}} M]\!] \qquad (\vec{U}_i, U_i) \subseteq [\![\lambda_{\vec{x}} N]\!]}{(\vec{U}_i, a_i) \in [\![\lambda_{\vec{x}}(MN)]\!]}.$$

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 by the consistency of $[\![\lambda_{\vec{x}} N]\!]$ proved above a and b are consistent as well.

Next we show $(\vec{V}, U) \subseteq [\![\lambda_{\vec{x}} N]\!]$. Let $a \in U$; we show $(\vec{V}, a) \in [\![\lambda_{\vec{x}} N]\!]$. Fix i such that $a \in U_i$ and $\vec{V} \vdash U_i$, and let $W_i := \{\, (\vec{U}_i, b) \mid b \in U_i \,\} \subseteq [\![\lambda_{\vec{x}} N]\!]$. Since by the side induction hypothesis $[\![\lambda_{\vec{x}} N]\!]$ 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 $(\vec{V}, U, b) \subseteq [\![\lambda_{\vec{x}} M]\!]$. Let

$$W' := \{(\vec{U}_1, U_1, a_1), \ldots, (\vec{U}_n, U_n, a_n)\} \subseteq [\![\lambda_{\vec{x}} M]\!].$$

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 $(\vec{V}, b) \in [\![\lambda_{\vec{x}} M]\!]$, as required.

Case (C). Assume $W \subseteq [\![\lambda_{\vec{x}} C]\!]$. Then W consists of $(\vec{U}, \vec{U}', \mathrm{C}\vec{a^*})$ such that $\vec{U}' \vdash \vec{a^*}$. Assume further $W \vdash (\vec{V}, \vec{V}', b)$. Then

$$\{\, \mathrm{C}\vec{a^*} \mid \exists_{\vec{U}, \vec{U}'}((\vec{U}, \vec{U}', \mathrm{C}\vec{a^*}) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}') \,\} \vdash b.$$

By definition of entailment b has the form $\mathrm{C}\vec{b^*}$ such that

$$W_i := \{\, a \mid \exists_{\vec{U}, \vec{U}', \vec{a^*}}(a = a_i^* \wedge (\vec{U}, \vec{U}', \mathrm{C}\vec{a^*}) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}') \,\} \vdash b_i^*.$$

We must show $(\vec{V}, \vec{V}', \mathrm{C}\vec{b^*}) \in [\![\lambda_{\vec{x}} \mathrm{C}]\!]$, i.e., $\vec{V}' \vdash \vec{b^*}$. 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 $a = a_i^*$, $(\vec{U}, \vec{U}', \mathrm{C}\vec{a^*}) \in W$ and $\vec{V}' \vdash \vec{U}'$. Hence $V_i' \vdash U_i' \vdash a_i^* = a$.

Case (D). Let $W = \{(\vec{U}_1, \vec{U}_1'', a_1), \ldots, (\vec{U}_n, \vec{U}_n'', a_n)\}$. For every i there is an $\vec{U}_i'$ such that

$$\frac{(\vec{U}_i, \vec{U}_i', a_i) \in [\![\lambda_{\vec{x}, \vec{y_i}} M_i(\vec{y_i})]\!] \qquad \vec{U}_i'' \vdash \vec{P}_i(\vec{U}_i')}{(\vec{U}_i, \vec{U}_i'', a_i) \in [\![\lambda_{\vec{x}} D]\!]}$$

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 $(\vec{V}, \vec{V}'', b) \in [\![\lambda_{\vec{x}} D]\!]$. 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 (42) 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 $I = \{1, \ldots, m\}$. Hence by the unification lemma $\vec{P}_1, \ldots, \vec{P}_m$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that

$$(\vec{P}_1\vartheta)(\vec{W}) = \cdots = (\vec{P}_m\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1') \sim \cdots \sim \vec{P}_m(\vec{V}_m').$$

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 (38)

$$(\vec{V}, (\vec{y_i}\vartheta)(\vec{W}), a_i) \in [\![\lambda_{\vec{x},\vec{y_i}} M_i(\vec{y_i})]\!]$$

and hence by (41)

$$(\vec{V}, \vec{W}, a_i) \in [\![\lambda_{\vec{x},\vec{y_i}} M_i(\vec{y_i}\vartheta)]\!].$$

But $M_i(\vec{y_i}\vartheta) = M_i\vartheta = M_1\vartheta = M_1(\vec{y_1}\vartheta)$ and hence for all $i \in I$

$$(\vec{V}, \vec{W}, a_i) \in [\![\lambda_{\vec{x},\vec{y_i}} M_1(\vec{y_1}\vartheta)]\!].$$

Therefore $X := \{\, (\vec{V}, \vec{W}, a_i) \mid i \in I \,\} \subseteq [\![\lambda_{\vec{x},\vec{y_i}} M_1(\vec{y_1}\vartheta)]\!]$. Since $\{\, a_i \mid i \in I \,\} \vdash b$, we have $X \vdash (\vec{V}, \vec{W}, b)$ and hence the induction hypothesis implies $(\vec{V}, \vec{W}, b) \in [\![\lambda_{\vec{x},\vec{y_i}} M_1(\vec{y_1}\vartheta)]\!]$. Using (41) again we obtain $(\vec{V}, (\vec{y_i}\vartheta)(\vec{W}), b) \in [\![\lambda_{\vec{x},\vec{y_i}} M_1(\vec{y_1})]\!]$. Since $\vec{V}'' \sim \vec{P}_1(\vec{V}_1') \sim \vec{P}_1((\vec{y_1}\vartheta)(\vec{W}))$ we obtain $(\vec{V}, \vec{V}'', b) \in [\![\lambda_{\vec{x}} D]\!]$, by (D). □

Corollary. $[\![\lambda_{\vec{x}} M]\!]$ *is an ideal.*

A.3. Preservation of values

We now prove that 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. 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 $[\![M]\!]_{\vec{x}}^{\vec{U}} := \{\, b \mid (\vec{U}, b) \in [\![\lambda_{\vec{x}} M]\!] \,\}$ and $[\![M]\!]_{\vec{x},\vec{y}}^{\vec{u},\vec{V}} := \bigcup_{\vec{U} \subseteq \vec{u}} [\![M]\!]_{\vec{x},\vec{y}}^{\vec{U},\vec{V}}$.

From (39) we obtain $[\![M]\!]^{\vec{U},V}_{\vec{x},y} = [\![M]\!]^{\vec{U}}_{\vec{x}}$ if $y \notin \mathrm{FV}(M)$, 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 $[\![\lambda_{\vec{x}} M]\!]$.

LEMMA. (a) *If* $\vec{V} \vdash \vec{U}$, $a \vdash b$ *and* $a \in [\![M]\!]^{\vec{U}}_{\vec{x}}$, *then* $b \in [\![M]\!]^{\vec{V}}_{\vec{x}}$.
(b) *If* $\vec{v} \supseteq \vec{u}$, $a \vdash b$ *and* $a \in [\![M]\!]^{\vec{u}}_{\vec{x}}$, *then* $b \in [\![M]\!]^{\vec{v}}_{\vec{x}}$.

PROOF. (a) $\vec{V} \vdash \vec{U}$, $a \vdash b$ and $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$ together imply $(\vec{V}, b) \in [\![\lambda_{\vec{x}} M]\!]$, by the deductive closure of $[\![\lambda_{\vec{x}} M]\!]$. (b) follows from (a). □

LEMMA. (a) $[\![x_i]\!]^{\vec{U}}_{\vec{x}} = \overline{U}_i$ *and* $[\![x_i]\!]^{\vec{u}}_{\vec{x}} = u_i$.
(b) $[\![\lambda_y M]\!]^{\vec{U}}_{\vec{x}} = \{\, (V,b) \mid b \in [\![M]\!]^{\vec{U},V}_{\vec{x},y} \,\}$ *and* $[\![\lambda_y M]\!]^{\vec{u}}_{\vec{x}} = \{\, (V,b) \mid b \in [\![M]\!]^{\vec{u},V}_{\vec{x},y} \,\}$.
(c) $[\![MN]\!]^{\vec{U}}_{\vec{x}} = [\![M]\!]^{\vec{U}}_{\vec{x}} [\![N]\!]^{\vec{U}}_{\vec{x}}$ *and* $[\![MN]\!]^{\vec{u}}_{\vec{x}} = [\![M]\!]^{\vec{u}}_{\vec{x}} [\![N]\!]^{\vec{u}}_{\vec{x}}$.

PROOF. (b) It suffices to prove the first part. But $(V,b) \in [\![\lambda_y M]\!]^{\vec{U}}_{\vec{x}}$ and $b \in [\![M]\!]^{\vec{U},V}_{\vec{x},y}$ are both equivalent to $(\vec{U}, V, b) \in [\![\lambda_{\vec{x},y} M]\!]$.

(c) For the first part we argue as follows.

$$\begin{aligned}
c \in [\![M]\!]^{\vec{U}}_{\vec{x}} [\![N]\!]^{\vec{U}}_{\vec{x}} &\leftrightarrow \exists_{V \subseteq [\![N]\!]^{\vec{U}}_{\vec{x}}} ((V,c) \in [\![M]\!]^{\vec{U}}_{\vec{x}}) \\
&\leftrightarrow \exists_V ((\vec{U}, V) \subseteq [\![\lambda_{\vec{x}} N]\!] \wedge (\vec{U}, V, c) \in [\![\lambda_{\vec{x}} M]\!]) \\
&\leftrightarrow (\vec{U}, c) \in [\![\lambda_{\vec{x}} (MN)]\!] \quad \text{by } (A) \\
&\leftrightarrow c \in [\![MN]\!]^{\vec{U}}_{\vec{x}}.
\end{aligned}$$

The second part is an easy consequence:

$$\begin{aligned}
c \in [\![M]\!]^{\vec{u}}_{\vec{x}} [\![N]\!]^{\vec{u}}_{\vec{x}} &\leftrightarrow \exists_{V \subseteq [\![N]\!]^{\vec{u}}_{\vec{x}}} ((V,c) \in [\![M]\!]^{\vec{u}}_{\vec{x}}) \\
&\leftrightarrow \exists_{V \subseteq [\![N]\!]^{\vec{u}}_{\vec{x}}} \exists_{\vec{U} \subseteq \vec{u}} ((V,c) \in [\![M]\!]^{\vec{U}}_{\vec{x}}) \\
&\leftrightarrow \exists_{\vec{U}_1 \subseteq \vec{u}} \exists_{V \subseteq [\![N]\!]^{\vec{U}_1}_{\vec{x}}} \exists_{\vec{U} \subseteq \vec{u}} ((V,c) \in [\![M]\!]^{\vec{U}}_{\vec{x}}) \\
&\leftrightarrow^{(*)} \exists_{\vec{U} \subseteq \vec{u}} \exists_{V \subseteq [\![N]\!]^{\vec{U}}_{\vec{x}}} ((V,c) \in [\![M]\!]^{\vec{U}}_{\vec{x}}) \\
&\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in [\![M]\!]^{\vec{U}}_{\vec{x}} [\![N]\!]^{\vec{U}}_{\vec{x}}) \\
&\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in [\![MN]\!]^{\vec{U}}_{\vec{x}}) \quad \text{by the first part} \\
&\leftrightarrow c \in [\![MN]\!]^{\vec{u}}_{\vec{x}}.
\end{aligned}$$

Here is the proof of the equivalence marked $(*)$. The upward direction is obvious. For the downward direction we use monotonicity. Assume $\vec{U}_1 \subseteq \vec{u}$, $V \subseteq [\![N]\!]^{\vec{U}_1}_{\vec{x}}$, $\vec{U} \subseteq \vec{u}$ and $(V,c) \in [\![M]\!]^{\vec{U}}_{\vec{x}}$. Let $\vec{U}_2 := \vec{U}_1 \cup \vec{U} \subseteq \vec{u}$. Then by monotonicity $V \subseteq [\![N]\!]^{\vec{U}_2}_{\vec{x}}$ and $(V,c) \in [\![M]\!]^{\vec{U}_2}_{\vec{x}}$. □

COROLLARY. $[\![\lambda_y M]\!]^{\vec{u}}_{\vec{x}} v = [\![M]\!]^{\vec{u},v}_{\vec{x},y}$.

PROOF.

$$\begin{aligned} b \in [\![\lambda_y M]\!]^{\vec{u}}_{\vec{x}} v &\leftrightarrow \exists_{V \subseteq v}((V,b) \in [\![\lambda_y M]\!]^{\vec{u}}_{\vec{x}}) \\ &\leftrightarrow \exists_{V \subseteq v}(b \in [\![M]\!]^{\vec{u},V}_{\vec{x},y}) \quad \text{by the lemma, part (b)} \\ &\leftrightarrow b \in [\![M]\!]^{\vec{u},v}_{\vec{x},y}. \end{aligned}$$ □

LEMMA (Substitution). $[\![M(z)]\!]^{\vec{u},[\![N]\!]^{\vec{u}}_{\vec{x}}}_{\vec{x},z} = [\![M(N)]\!]^{\vec{u}}_{\vec{x}}$.

PROOF. By induction on M, and cases on the form of M.
Case $\lambda_y M$. For readability we leave out $\vec{x}$ and $\vec{u}$.

$$\begin{aligned} [\![\lambda_y M(z)]\!]^{[\![N]\!]}_{z} &= \{\, (V,b) \mid b \in [\![M(z)]\!]^{[\![N]\!],V}_{z,y} \,\} \\ &= \{\, (V,b) \mid b \in [\![M(N)]\!]^{V}_{y} \,\} \quad \text{by induction hypothesis} \\ &= [\![\lambda_y M(N)]\!] \quad \text{by the last lemma, part (b)} \\ &= [\![(\lambda_y M)(N)]\!]. \end{aligned}$$

The other cases are easy. □

LEMMA (Preservation of values, β). $[\![(\lambda_y M(y))N]\!]^{\vec{u}}_{\vec{x}} = [\![M(N)]\!]^{\vec{u}}_{\vec{x}}$.

PROOF. Again we leave out $\vec{x}$, $\vec{u}$. By the last two lemmata and the corollary, $[\![(\lambda_y M(y))N]\!] = [\![\lambda_y M(y)]\!][\![N]\!] = [\![M(y)]\!]^{[\![N]\!]}_{y} = [\![M(N)]\!]$. □

LEMMA (Preservation of values, η). $[\![\lambda_y(My)]\!]^{\vec{u}}_{\vec{x}} = [\![M]\!]^{\vec{u}}_{\vec{x}}$ *if* $y \notin \mathrm{FV}(M)$.

PROOF.

$$\begin{aligned} (V,b) \in [\![\lambda_y(My)]\!]^{\vec{u}}_{\vec{x}} &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}}((\vec{U},V,b) \in [\![\lambda_{\vec{x},y}(My)]\!]) \\ &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}}((\vec{U},V,b) \in [\![\lambda_{\vec{x}} M]\!]) \quad \text{by (40)} \\ &\leftrightarrow (V,b) \in [\![M]\!]^{\vec{u}}_{\vec{x}}. \end{aligned}$$ □

Bibliography

Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer Verlag, Berlin, Heidelberg, New York, 3rd edition, 2004.

Joseph L. Bates and Robert L. Constable. Proofs as programs. *ACM Transactions on Programming Languages and Systems*, 7(1):113–136, January 1985.

Ulrich Berger. Program extraction from normalization proofs. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 91–106. Springer Verlag, Berlin, Heidelberg, New York, 1993.

Ulrich Berger. Uniform Heyting arithmetic. *Annals of Pure and Applied Logic*, 133:125–148, 2005.

Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.

Ulrich Berger, Matthias Eberl, and Helmut Schwichtenberg. Term rewriting for normalization by evaluation. *Information and Computation*, 183:19–42, 2003.

Albert Dragalin. New kinds of realizability. In *Abstracts of the 6th International Congress of Logic, Methodology and Philosophy of Sciences*, pages 20–24, Hannover, Germany, 1979.

Harvey Friedman. Classically and intuitionistically provably recursive functions. In D.S. Scott and G.H. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978.

Harry Fürstenberg. On the infinitude of primes. *Amer. Math. Monthly*, 62: 353, 1955.

Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

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

David Hilbert. Über das Unendliche. *Mathematische Annalen*, 95:161–190, 1925.

Hajime Ishihara. A note on the Gödel–Gentzen translation. *Mathematical Logic Quarterly*, 46:135–137, 2000.

Ingebrigt Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 4:119–136, 1937.

Andrey N. Kolmogorov. Zur Deutung der intuitionistischen Logik. *Math. Zeitschr.*, 35:58–65, 1932.

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

Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

Grigori Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978. Translated from: *Zap. Nauchn. Semin. LOMI* 49 (1975).

J. Alan Robinson. A machine–oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

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

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

Index

CV, 53
$\tilde{\exists}, \tilde{\forall}$, 2, 9
F, 43
$\bot$, 2, 72
$\tilde{\wedge}$, 9

Alexandrov condition, 23
algebra, 26
  explicit, 28
  finitary, 27
  nested, 26
  structure-finitary, 27
  unnested, 26
append, 34
application, 20, 24
approximable map, 21

Berger, 31, 50
binary number, 27
binary tree, 27
boolean, 26

case-construct, 32
$\mathcal{C}$-operator, 32
choice theorem, 62
clause, 42, 54
closure axiom, 48
coinduction, 48
coinductive definition
  of cototality, 47
companion, 48
composition, 97
comprehension term, 41
computation rule, 34, 36
conjunction, 6
consistent, 19
Constable, 83
constructor
  as continuous function, 29
constructor pattern, 36
constructor symbol, 27
constructor type, 26
continuation passing style, 91
conversion, 34, 37, 38, 105
  D-, 34, 37
  β-, 34
  η-, 34
  $\mathcal{R}$-, 34
corecursion
  operator, 36
cototality, 47
Curry-Howard correspondence, 1

decoration
  algorithm, 84
  of proofs, 84
derivable, 2
  classically, 10
  intuitionistically, 9
derivation, 27
destructor, 35
disjunction, 5, 6
drinker formula, 13
dual, 48
duplication, 31

Eberl, 31
effectivity principle, 18
elimination axiom, 42, 54
entailment, 19
equality
  decidable, 37, 41
  Leibniz, 41, 43, 53

even number, 41, 51
ex-falso-quodlibet, 2, 9, 43, 48
existential quantifier, ix, 6

falsity **F**, 43
falsum $\bot$, 2
final predicate, 51
finite support principle, 17, 23
formal neighborhood, 19
formula
  computationally relevant (c.r.), 51
  decorated, 51
  non-computational (n.c.), 51, 53
formula form, 40
function
  continuous, 22
  monotone, 23
  strict, 30
functional
  computable, 29
  cototal, 31
  partial continuous, 29
  total, 31

Gentzen, 1, 2
Gödel-Gentzen translation, 15
greatest-fixed-point axiom, 47, 48

Harrop formula, 50, 51
Harrop predicate, 51
height
  of syntactic expressions, 28

ideal, 19
  cototal, 30
  structure-cototal, 30
  structure-total, 30
  total, 30
idempotent, 97
identity, 27
independence of premise, 62
induction
  general, 47, 69
  strengthened form, 39
induction axiom, 47
inductive definition
  of totality, 39
information system, 19
  coherent, 102
  flat, 19
  of a type ρ, 28
introduction axiom, 42, 54
intuitionistic logic, 2
Inv, 62
invariance axiom, 50, 62

Johansson, 2

Kolmogorov, 2, 57

Larsen, 19
least-fixed-point axiom, 42
Leibniz, 41
Leibniz equality, 50, 53, 54
list, 27

map operator, 32
Markov, 56
mgu, 98
minimal logic, 2
Mints, 31
monotonicity principle, 18

n.c., 50
natural number, 26
nullary clause, 48
nulltype, 58

one-clause-nc, 54, 69
ordinal, 27

parameter argument type, 26
parameter premise, 41
positive number, 27
predicate
  coinductive, 48
  computationally relevant (c.r.), 51
  decorated, 51
  inductive, 40
  inductively defined, 40
  nested, 41
  non-computational (n.c.), 51
  one-clause n.c. defined, 50
  unitary, 50
  unnested, 41
predicate form, 40
product, 27
progressive, 47
projection, 34
proof
  pattern, 83

realizability, 58
recursion
  general, 35, 69
  operator, 31
recursive argument type, 26
recursive premise, 41
redex
  D-, 34
relation
  accessible part, 44

Schwichtenberg, 31
Scott, 19
Scott condition, 23
Scott topology, 22
set of tokens
  deductively closed, 19
soundness theorem
  for realizability, 65
stability, 10
strictly positive, 26
substitution, 97
  more general, 97
sum, 27

T, 33
TCF, 42
term
  extracted, 65
  of T^+, 36
  of Gödel's T, 33
theorem of choice, 62
token, 19
  extended, 28
totality, 39, 45
  absolute, 45
  relative, 45
  structural, 45
transitive closure, 41, 42
tree, 27
  binary, 27
type, 26
  base, 28
  higher, 28
  level of, 28
type form, 26

unary number, 26
unifier
  most general, 98
uninstantiated formula
  of an axiom, 84
unit, 26

variable
  computational, 53
  non-computational, 53
variable condition, vii, 6

Winskel, 19
witnessing predicate, 59