Indexed Induction-Recursion

Peter Dybjer [a,⋆]

[a] *Department of Computer Science and Engineering, Chalmers University of Technology, 412 96 Göteborg, Sweden*
Email: peterd@cs.chalmers.se, http://www.cs.chalmers.se/∼peterd/,
Tel: +46 31 772 1035, Fax: +46 31 772 3663.

Anton Setzer [b,1]

[b] *Department of Computer Science, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, UK,*
Email: a.g.setzer@swan.ac.uk, http://www.cs.swan.ac.uk/∼csetzer/,
Tel: +44 1792 513368, Fax: +44 1792 295651.

Abstract

An indexed inductive definition (IID) is a simultaneous inductive definition of an indexed family of sets. An inductive-recursive definition (IRD) is a simultaneous inductive definition of a set and a recursive definition of a function on that set. An indexed inductive-recursive definition (IIRD) is a combination of both.

We present a closed theory which allows us to introduce all IIRD in a natural way without much encoding. By specialising it we also get a closed theory of IID. Our theory of IIRD includes essentially all definitions of sets which occur in Martin-Löf type theory. We show in particular that Martin-Löf's computability predicates for dependent types and Palmgren's higher order universes are special kinds of IIRD and thereby clarify why they are constructively acceptable notions.

We give two axiomatisations. The first formalises a principle for introducing meaningful IIRD by using the data-construct in the original version of the proof assistant Agda for Martin-Löf type theory. The second one admits a more general form of introduction rule, including the introduction rule for the intensional identity relation, which is not covered by the first axiomatisation. If we add an extensional identity relation to our logical framework, we show that the theories of restricted and general IIRD are equivalent by interpreting them in each other.

Finally, we show the consistency of our theories by constructing a model in classical set theory extended by a Mahlo cardinal.

Key words: Dependent type theory, Martin-Löf Type Theory, inductive definitions, inductive-recursive definitions, inductive families, initial algebras, normalisation proofs, generic programming.

1 Introduction

Martin-Löf type theory is a foundational framework for constructive mathematics. It is also a functional programming language with a powerful type theory. In this theory, induction is one of the two principles for constructing sets; the other is function space formation. For this reason, it has been important to spell out the principles of inductive definability underlying Martin-Löf type theory [11,12]. A similar notion of inductive definability is a core concept of the Calculus of Inductive Constructions [10,28], the impredicative type theory underlying the Coq-system.

Inductive-recursive definitions. Simple inductive definitions and function spaces alone do not suffice to define all sets in Martin-Löf type theory. Consider for example the universe à la Tarski [22]. It consists of a set U of codes for small sets, and a decoding function T, which maps a code to the set it denotes. This definition is simultaneously inductive and recursive: U is inductively generated at the same time as T is defined by recursion on the way the elements of U are generated. To see why they are simultaneously defined we look at the closure of small sets under Σ stating that the disjoint union of a small family of small sets is small. This is expressed by an introduction rule for U: if $a : \mathrm{U}$ and $b(x) : \mathrm{U}$ for $x : \mathrm{T}(a)$, then $\widehat{\Sigma}(a, b) : \mathrm{U}$, that is,

$$\widehat{\Sigma} : (a : \mathrm{U}) \to (\mathrm{T}(a) \to \mathrm{U}) \to \mathrm{U}$$

is a constructor for U. Here $(x : A) \to B$ is the dependent function space, that is, the set of functions f, which map an element $a : A$ to an element $f(a)$ of $B[x := a]$. There is also a recursive equation for T:

$$\mathrm{T}(\widehat{\Sigma}(a, b)) = \Sigma x : \mathrm{T}(a).\mathrm{T}(b(x)).$$

Observe that the introduction rule for U refers to T, something which is not allowed in an inductive definition. Therefore, T has to be defined simultaneously with U. To capture this we need the more general notion of an inductive-recursive definition [13], where we simultaneously define a set U and a decoding $\mathrm{T} : \mathrm{U} \to D$ into an arbitrary type D.

Other examples of inductive-recursive definitions were known before (larger universes, computability predicates for dependent types), but the new idea [13] was that these are instances of a general notion, equally natural as that of an inductive definition. It is a constructively acceptable notion: its rules can

⋆ Supported by Vetenskapsrådet, grant Typed Lambda Calculus and Applications.
[1] Supported by Nuffield Foundation, grant ref. NAL/00303/G and EPSRC grant GR/S30450/01.

be justified by meaning explanations in the sense of Martin-Löf in a similar way as the rules for inductive definitions.

In the original presentation [13] induction-recursion is described as an external schema spelling out criteria for correct formation, introduction, elimination, and equality rules. However, this schema is not fully rigorous and the authors therefore presented a closed axiomatisation of inductive-recursive definitions in Martin-Löf type theory [14,16].

Indexed inductive-recursive definitions. There are many examples (see Section 3) where we want to define a whole family of sets simultaneously, but the previous articles [14,16] only consider the case of an inductive-recursive definition of *one* set at a time. It is the objective of this paper to extend this to *indexed* inductive-recursive definitions (IIRD), that is, inductive-recursive definitions of indexed families of sets $\mathrm{U}(i)$ and functions $\mathrm{T}(i) : \mathrm{U}(i) \to D[i]$ for $i : I$, where I is a set and $D[i]$ is an I-indexed family of types. (We must write $D[i]$ rather than $D(i)$, since the typing $D : I \to \mathrm{type}$ is not expressible in our logical framework. See Section 2 for more information about the notation.)

Our theories for IIRD are the most general existing versions of Martin-Löf type theory. They encompass virtually all sets that have been used in Martin-Löf type theory before, with the exception of some notions of even larger universes (like the Mahlo universe) of proof-theoretic interest considered by the second author [31,32].

Indexed inductive definitions (IID). IID is the principle of defining an indexed family of sets by a simultaneous inductive definition. IID are ubiquitous when using Martin-Löf type theory for formalising mathematics or programming problems.

IID appear as special cases of IIRD where the recursively defined function is degenerate (has codomain $\mathbf{1}$). Therefore, a side effect of our paper is to provide the first closed axiomatisation of IID.

Several examples of IID can be found in Section 3.1 and of proper IIRD in Section 3.2.

General and restricted IIRD. We consider two classes of indexed inductive-recursive definitions: *general IIRD* (as in [13]) and *restricted IIRD* as introduced by T. Coquand for use in the Half and Agda [7] systems. Coquand's restricted IIRD have not been spelled out in detail in the literature, although they are supported by the Half and Agda systems. To illustrate the difference

between restricted and general IIRD, we consider the inductive definition of the even number predicate $\mathrm{Even} : \mathrm{N} \to \mathrm{set}$. As a general IID, it is inductively generated by the two rules

$$\mathrm{C}_0 : \mathrm{Even}(0) \ ,$$
$$\mathrm{C}_1 : (n : \mathrm{N}) \to \mathrm{Even}(n) \to \mathrm{Even}(\mathrm{S}(\mathrm{S}(n))) \ .$$

As a restricted IIRD, the constructors instead have the following types:

$$\mathrm{C}'_0 : (m : \mathrm{N}) \to (m =_{\mathrm{N}} 0) \to \mathrm{Even}(m) \ ,$$
$$\mathrm{C}'_1 : (m : \mathrm{N}) \to (n : \mathrm{N}) \to \mathrm{Even}(n) \to (m =_{\mathrm{N}} \mathrm{S}(\mathrm{S}(n))) \to \mathrm{Even}(m) \ ,$$

where $=_{\mathrm{N}}$ is equality of natural numbers. In restricted IIRD, we can determine the constructors (and their arguments) of a particular set $\mathrm{U}(i)$ in the family without analysing i. For example, $\mathrm{Even}(m)$ has constructors $\mathrm{C}'_0(m)$ and $\mathrm{C}'_1(m)$. More formally, the first argument of each constructor is the index of the element introduced by the constructor. In an implementation which allows full recursion (like Agda), we can use case-distinction for one element as elimination principle for restricted IIRD. For instance, an element of $\mathrm{Even}(m)$ either has the form $\mathrm{C}'_0(m, p)$ or the form $\mathrm{C}'_1(m, n, p, q)$. In Agda, the argument m of the constructor is omitted (instead there is some notation which makes the whole type $\mathrm{Even}(m)$ part of the name of the constructor – we omit this here). The notation for case distinction is therefore as follows:

$$\text{case } m \text{ of } \{\mathrm{C}'_0(p) \to \cdots ; \mathrm{C}'_1(n, p, q) \to \cdots\}$$

In general IIRD we do not have the above restriction: if C is a constructor of an indexed set U and we apply it to the arguments $\vec{a}$, then the index i such that $\mathrm{C}(\vec{a}) : \mathrm{U}(i)$ may depend on the arguments $\vec{a}$ in an arbitrary way. Case-distinction is no longer possible for one individual element of $\mathrm{U}(i)$. For example, in the definition of Even we have to define functions simultaneously for all pairs $\langle n, p\rangle$ such that $n : \mathrm{N}$ and $p : \mathrm{Even}(n)$, and we need to use pattern matching in order to distinguish between the cases $\langle n, p\rangle = \langle 0, \mathrm{C}_0\rangle$ and $\langle n, p\rangle = \langle \mathrm{S}(\mathrm{S}(m)), \mathrm{C}_1(m, q)\rangle$.

Martin-Löf's definition of the equality relation as inductively generated by the reflexivity rule is a key example of a general IID with no corresponding restricted version (unless we assume that the framework already contains an equality $=_A$ in which case it would be pointless to introduce a second equality relation).

The proof assistant Alf [19,2] supports the use of general IIRD by using Coquand's pattern matching with dependent types [9]. Recently, a construct for general IIRD has also been added to Agda.

Generic dependent type theory. Our axiomatisations of IIRD are related to generic programming. In generic functional programming [18,17], generic functions are defined by induction on the code of a data type. Our axiomatisation provides a type of codes for all IIRD, and therefore we can write programs by induction on the codes for IIRD. In this sense, we here provide a "generic dependent type theory", a version of Martin-Löf type theory with generic formation, introduction, elimination, and equality rules. Benke, Dybjer, and Jansson [5] further develop an approach to generic programs and proofs which is based on ideas from the present paper. Other references to generic programs and proofs in dependent type theory are Pfeifer and Rueß [29] and Altenkirch and McBride [3].

Alternative axiomatisations. We emphasise that our objective is to axiomatise IIRD (and IID) as they are naturally presented in terms of rules for generating new elements of a set. It is thus not only a question of presenting a theory with a certain proof-theoretic strength: we would have reached equal proof-theoretic strength using a version of Martin-Löf type theory with well-orderings and a Mahlo universe. However, working with IIRD in such a theory would require elaborate encodings. Instead, we achieve a close correspondence between our codes for IIRD and the syntax for corresponding definitions in the Agda-system (using the `data`-construct). The latter can be viewed as a sugared version of the former.

By formalising a concrete theory of IIRD we give a rigorous definition of the concept of IIRD which makes metamathematical analysis possible. For example, in Section 7 we show how to interpret the theories of general and restricted IIRD in each other. In future work we plan to show further reductions of theories axiomatised in this way, for example, we plan to show how to interpret the theory of IIRD in the theory of IRD and the theory of "small" general IIRD in the theory of IID.

When explaining the idea behind our axiomatisations, we use some categorical notions. In particular, we consider algebras of certain endofunctors on the slice category $\mathbf{Fam}(I)/D$, where $\mathbf{Fam}(I)$ is the category of I-indexed types (see Subsection 4.1). From the point of view of category theory it would be natural to use two more ideas from categorical semantics: that I-indexed types can be represented as fibrations, and that we expect our algebras to be initial. However, neither of these ideas taken literally gives rise to rules which are close to the usual type-theoretic rules for IIRD. The goal of this paper is to introduce and analyse theories with good intensional properties. The reduction of our theory to initial algebras requires extensional equality. Moreover, even in an extensional setting working directly with initial algebras would add an overhead which may make it impractical for use in proof assistants.

Nevertheless, we use categorical ideas in a limited way and show for example that $\mathbf{Fam}(I)/D$ is equivalent to the category $\mathbf{Type}/((i : I) \times D[i])$. This equivalence suggests that it is possible to reduce IIRD to IRD (non-indexed inductive-recursive definitions). As already mentioned we plan to show this formally in a future publication, where we also plan to discuss the relationship between IIRD and initial algebras.

2 The Logical Framework

Before giving the rules for IIRD we need to introduce the basic Logical Framework of dependent types. This is essentially Martin-Löf's Logical Framework [25] extended with rules for dependent product types $(x : A) \times B$ and the types $\mathbf{0}$, $\mathbf{1}$, and $\mathbf{2}$. The complete set of rules for the logical framework can be found in Appendix A.1. Note that we will work in intensional type theory, except for Sect. 7 and when explicitly stated.

The Logical Framework has the following forms of judgements: Γ context (where Γ is of the form $x_1 : A_1, \ldots, x_n : A_n$); and $A : \text{type}$, $A = B : \text{type}$, $a : A$, $a = b : A$, depending on contexts Γ (written as $\Gamma \Rightarrow A : \text{type}$, etc.). We have $\text{set} : \text{type}$ and if $A : \text{set}$, then $A : \text{type}$. The collection of types is closed under the formation of dependent function types written as $(x : A) \to B$. ($\Pi x : A.B$ is a common alternative notation for this construction, but it is used here for dependent function *sets*, that is, the inductively defined sets with constructor $\lambda : ((x : A) \to B) \to (\Pi x : A.B)$.) The elements of $(x : A) \to B$ are denoted by $(x : A)a$ (abstraction of x in a – this is often denoted by $\lambda x : A.a$ in the literature). Application is written as $a(b)$. We follow Martin-Löf and have η-rules (note that we are working in intensional type theory) as well as β-rules in the logical framework. Types are also closed under the formation of dependent products written as $(x : A) \times B$. (A common alternative notation is $\Sigma x : A.B$, but this is here used for the inductively defined set with introduction rule $\mathrm{p} : ((x : A) \times B) \to (\Sigma x : A.B)$.) The elements of $(x : A) \times B$ are denoted by $\langle a, b \rangle$, the projection functions by π_0 and π_1 and again we have β and η-rule (surjective pairing). There is the type $\mathbf{1}$, with unique element $\star : \mathbf{1}$ and η-rule expressing that, if $a : \mathbf{1}$, then $a = \star : \mathbf{1}$. Furthermore, we have the empty type $\mathbf{0}$ with elimination rule case_0.

Moreover, in our version of the logical framework we include the type $\mathbf{2}$ with two elements $\star_0 : \mathbf{2}$ and $\star_1 : \mathbf{2}$, ordinary elimination rule $\mathrm{case}_2 : (i : \mathbf{2}, A[\star_0], A[\star_1]) \to A[i]$ (where $i : \mathbf{2} \Rightarrow A[i] : \text{type}$) and elimination into type, expressed as $\mathrm{case}_2^{\mathrm{type}}(i, A, B) : \text{type}$ for $i : \mathbf{2}$, $A : \text{type}$, $B : \text{type}$. We need elimination into type, since we want to inductively define indexed families of sets $U : I \to \text{set}$ and functions $T : (i : I) \to U(i) \to D[i]$ where $D[i]$ depends non-trivially on i, as in the definition of Palmgren's higher-order universe (where

for instance $D[0] := \text{set}$, $D[1] := ((X : \text{set}) \times (X \to \text{set})) \to ((X : \text{set}) \times (X \to \text{set}))$, etc; see Subsect. 3.2 for details).

We can now define the disjoint union of two types $A + B := (i : \mathbf{2}) \times \text{case}_2^{\text{type}}(i, A, B)$, and prove the usual formation, introduction, elimination and equality rules (see Def. A.2 for details).

We also add a level between set and type, which we call stype for small types: stype : type. (The reason for the need for stype is discussed in [13].) If $a : \text{set}$ then $a : \text{stype}$. Moreover, stype is also closed under dependent function types, dependent products and includes $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$. However, set itself will not be in stype. The logical framework does not have any rules for introducing elements of set. They will be introduced by IIRD later and set will therefore consist exactly of the sets introduced by IIRD.

In Section 7 and when explicitly stated, we will use the rules of extensional equality (where $a =_A b$ is the equality type for $a, b : A$) which can be found in Appendix A.2. Note that we introduce $a =_A b$ for $A : \text{type}$.

We also use some standard abbreviations, see Definition A.1 in Appendix A.1 for a complete list. We just mention the following: We omit the type in an abstraction and write $(x)a$ instead of $(x : A)a$; we write repeated function spaces as $(x_1 : A_1, \ldots, x_n : A_n) \to A$ instead of $(x_1 : A_1) \to \cdots (x_n : A_n) \to A$, repeated application as $a(b_1, \ldots, b_n)$ instead of $a(b_1) \cdots (b_n)$. We sometimes put arguments in subscript position and thus also write $a_{b_1,\ldots,b_n}$ for (repeated) application. Repeated abstraction as is written as $(x_1 : A_1, \ldots, x_n : A_n)a$ or $(x_1, \ldots, x_n)a$ instead of $(x_1 : A_1) \cdots (x_n : A_n)a$; and we write $(-)$ for the abstraction (x) of a variable x, which is not used later.

In the following, we will sometimes refer to a type depending on a variable x. We want to use the notation $D[t]$ for $D[x := t]$ for some fixed variable x and D for $(x)D[x]$. Note that we cannot simply introduce $D : I \to \text{type}$, since this goes beyond the logical framework. Instead, we introduce the notion of an abstracted expression, which is an expression together with one or several designated free variables. For an abstracted expression E, $E[t_1, \ldots, t_n]$ means the substitution of the variables by $t_1, \ldots, t_n$. If we let D above be an abstracted expression of the form $(x)E$, then $D[t]$ denotes $D[x := t]$ and we can write D as parameter for $(x)E$. More formally:

Definition 2.1 *(a) An n-times abstracted expression is an expression $(x_1, \ldots, x_n)E$ where $x_1, \ldots, x_n$ are distinct variables and E an expression of the language of type theory. An abstracted expression is a 1-times abstracted expression.*

(b) $((x_1, \ldots, x_n)E)[t_1, \ldots, t_n] := E[x_1 := t_1, \ldots, x_n := t_n]$.

(c) Whenever we write $s[a_1, \ldots, a_n]$, s is to be understood as an n-times abstracted expression.

(d) If $U : A \to B$, we identify U with the abstracted expression $(a)U(a)$.

3 Some Examples

3.1 Examples of Indexed Inductive Definitions

Trees and forests. Many IID occur as the simultaneous inductive definition of finitely many sets, each of which has a different name. One example is the set of well-founded trees Tree with finite branching degrees, which is defined together with the set Forest of finite lists of such trees. The constructors are:

$$\begin{aligned} \text{tree} &: \text{Forest} \to \text{Tree} \ , \\ \text{nil} &: \text{Forest} \ , \\ \text{cons} &: \text{Tree} \to \text{Forest} \to \text{Forest} \ . \end{aligned}$$

If we instead of Forest and Tree introduce $\text{Tree}' : \mathbf{2} \to \text{set}$ and replace Tree by $\text{Tree}'(\star_0)$ and Forest by $\text{Tree}'(\star_1)$ in the types of tree, nil, cons above, then we obtain an IID with index set $\mathbf{2}$.

The accessible part of a relation. Let I be a set and $<: I \to I \to \text{set}$ be a binary relation on it. We define the accessible part (the largest well-founded initial segment) of $<$ as a predicate $\text{Acc} : I \to \text{set}$ by a generalised indexed inductive definition with one introduction rule:

$$\text{acc} : (i : I) \to ((x : I) \to (x < i) \to \text{Acc}(x)) \to \text{Acc}(i) \ .$$

Note that acc introduces elements of $\text{Acc}(i)$ while referring to possibly infinitely many elements of the sets $\text{Acc}(x)$ (for each $x : I$ and each proof of $x < i$).

The identity relation. This is the only example of an IID in Martin-Löf's original formulation of type theory with a non-trivial index type. For historical reasons it is often referred to as the "intensional identity type" – a more appropriate name would be "identity set". Assume $A : \text{set}$. The rules for the intensional identity on A express that it is the least reflexive relation on A. The formation rule is $\text{I} : A \to A \to \text{set}$. The introduction rule expresses that it is reflexive:

$$\text{r} : (a : A) \to \text{I}(A, a, a) \ .$$

The elimination rule expresses that the only elements of an identity set are those which are constructed by the introduction rule. So to define a function $a : A, b : A, p : \mathrm{I}(A, a, b) \Rightarrow C[a, b, p]$ it is sufficient to define the step-function s such that for every $a : A$ we have $s[a] : C[a, a, \mathrm{r}(a)]$. Thus the elimination rule states that for every $a, b : A$ and $p : \mathrm{I}(A, a, b)$ we have $\mathrm{J}(A, C, a, b, p, (x)s[x]) : C[a, b, p]$. Usually, in Martin-Löf type theory one assumes that $C[a, b, p]$ is a set. However, in this paper the elimination rule is strengthened so that $C[a, b, p]$ can be a type, that is, we have a so called *large* elimination rule. We will in general consider large elimination rules in this paper.

Context free grammars. IID occur very frequently in applications in computer science. For example a context free grammar over a finite alphabet Σ and a finite set of nonterminals NT is an $\mathrm{NT} \times \Sigma^*$-indexed inductive definition, where each production corresponds to an introduction rule.

As an example, consider the context free grammar with $\Sigma = \{a, b\}$, $\mathrm{NT} = \{A, B\}$ and productions $A \longrightarrow a$, $A \longrightarrow BB$, $B \longrightarrow AA$, $B \longrightarrow b$. This corresponds to an inductive definition of a family of sets L indexed over $\mathrm{NT} \times \Sigma^*$, where $\mathrm{L}(A, \alpha)$ is the set of derivation trees of the string α from the start symbol A. So α is in the language generated by the grammar with start symbol A iff $\mathrm{L}(A, \alpha)$ is inhabited. L has one constructor for each production: $C_0 : \mathrm{L}(A, a)$, $C_1 : \mathrm{L}(B, \alpha) \to \mathrm{L}(B, \beta) \to \mathrm{L}(A, \alpha * \beta)$, and $C_2 : \mathrm{L}(A, \alpha) \to \mathrm{L}(A, \beta) \to \mathrm{L}(B, \alpha * \beta)$, $C_3 : \mathrm{L}(B, b)$, where $*$ denotes concatenation.

Alternatively, we can inductively define an NT-indexed set D of "abstract syntax trees" for the grammar, and then recursively define the string $d(A, p)$ ("concrete syntax") corresponding to the abstract syntax tree $p : \mathrm{D}(A)$. In the example above we get $C_0 : \mathrm{D}(A)$, $C_1 : \mathrm{D}(B) \to \mathrm{D}(B) \to \mathrm{D}(A)$, $C_2 : \mathrm{D}(A) \to \mathrm{D}(A) \to \mathrm{D}(B)$, $C_3 : \mathrm{D}(B)$. Furthermore,
$d(A, C_0) = a$, $d(A, C_1(p, q)) = d(B, p) * d(B, q)$,
$d(B, C_2(p, q)) = d(A, p) * d(A, q)$, $d(B, C_3) = b$.

The simply typed lambda calculus. The traditional way of introducing the simply typed lambda calculus is to first define inductively the set of lambda types Ltype, the set of lambda contexts Lcontext, and the set of "raw" lambda terms Lraw. In typed lambda calculus à la Curry Lraw is just the set of untyped lambda terms, whereas in the typed lambda calculus à la Church the lambda terms have type labels associated with each binding occurrence of a variable. Then we define the typing relation inductively by writing down the typing rules for the simply typed lambda calculus. The typing relation is a ternary relation on Lcontext $\times$ Lterm $\times$ Ltype.

We here show an alternative approach, which is to directly give an IID of the

well-typed terms $\mathrm{Lterm}(\Gamma, \sigma)$ of type σ in context Γ (see also Altenkirch and Reus [4] and Qiao [30]).

We first define the type Ltype of lambda types, constructed from the basic type o and function types (where $\mathrm{ar}(\sigma, \tau)$ denotes the type $\sigma \to \tau$) as a simple inductive definition, having constructors

$$\mathrm{o} : \mathrm{Ltype}$$
$$\mathrm{ar} : \mathrm{Ltype} \to \mathrm{Ltype} \to \mathrm{Ltype}$$

We define the type Lcontext of contexts as a list of types:

$$\mathrm{empty} : \mathrm{Lcontext}$$
$$\mathrm{cons} : \mathrm{Lcontext} \to \mathrm{Ltype} \to \mathrm{Lcontext}$$

A variable v of type σ in context Γ will be an element of a set $\mathrm{Lvar}(\Gamma, \sigma)$, which is given as an IID indexed over $\Gamma : \mathrm{Lcontext}$ and $\sigma : \mathrm{Ltype}$. In de Bruijn notation, the variable v_0 (denoted by $\mathrm{zVar}(\cdots)$ below) refers to the last element bound in the context, and the variable v_{n+1} (denoted by $\mathrm{sVar}(\cdots, y)$, if y denotes v_n) in context Γ, σ is the result of lifting the variable v_n in context Γ to context Γ, σ. So, we get the following IID:

$$\mathrm{zVar} : (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}) \to \mathrm{Lvar}(\mathrm{cons}(\Gamma, \sigma), \sigma)$$
$$\mathrm{sVar} : (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \tau : \mathrm{Ltype}) \to \mathrm{Lvar}(\Gamma, \tau)$$
$$\to \mathrm{Lvar}(\mathrm{cons}(\Gamma, \sigma), \tau)$$

The set of λ-terms is given as follows: Variables are λ-terms; if s is a term of type $\sigma \to \tau$ and t a term of type σ then $\mathrm{ap}(s, t)$ is a term of type τ; if s is a term of type τ in context Γ, σ, then $\lambda(s)$ is a term of type $\sigma \to \tau$ in context Γ. So, the sets $\mathrm{Lterm}(\Gamma, \sigma)$ of λ-terms of type σ in context Γ are given by the following IID:

$$\mathrm{var} : (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}) \to \mathrm{Lvar}(\Gamma, \sigma) \to \mathrm{Lterm}(\Gamma, \sigma)$$
$$\mathrm{ap} : (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \tau : \mathrm{Ltype})$$
$$\to \mathrm{Lterm}(\Gamma, \mathrm{ar}(\sigma, \tau)) \to \mathrm{Lterm}(\Gamma, \sigma) \to \mathrm{Lterm}(\Gamma, \tau)$$
$$\mathrm{lam} : (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \tau : \mathrm{Ltype}) \to \mathrm{Lterm}(\mathrm{cons}(\Gamma, \sigma), \tau)$$
$$\to \mathrm{Lterm}(\Gamma, \mathrm{ar}(\sigma, \tau))$$

The definitions of Lvar and Lterm above correspond to the following four typing rules for the simply typed lambda calculus, where we have omitted the implicit fifth rule (corresponding to the constructor var) which expresses that a variable of type σ is also a term of type σ.

$$\Gamma, \sigma \Rightarrow v_0 : \sigma \qquad\qquad \frac{\Gamma \Rightarrow v_n : \tau}{\Gamma, \sigma \Rightarrow v_{n+1} : \tau}$$

$$\frac{\Gamma \Rightarrow s : \sigma \to \tau \qquad \Gamma \Rightarrow t : \sigma}{\Gamma \Rightarrow s\ t : \tau} \qquad\qquad \frac{\Gamma, \sigma \Rightarrow s : \tau}{\Gamma \Rightarrow \lambda(s) : \sigma \to \tau}$$

It is easy to translate the above into a restricted IID (without the need for a built-in equality): First, we define equalities $=_{\mathrm{Ltype}}$ and $=_{\mathrm{Lcontext}}$ in a straightforward way by case analysis on Ltype and Lcontext respectively. The constructors Lvar and Lterm then have the following types:

$$\begin{aligned}
\mathrm{zVar} &: (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \Delta : \mathrm{Lcontext}, p : \mathrm{cons}(\Delta, \sigma) =_{\mathrm{Lcontext}} \Gamma) \\
&\quad \to \mathrm{Lvar}(\Gamma, \sigma) \\
\mathrm{sVar} &: (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \Delta : \mathrm{Lcontext}, \tau : \mathrm{Ltype}) \\
&\quad \to (\mathrm{cons}(\Delta, \tau) =_{\mathrm{Lcontext}} \Gamma) \to \mathrm{Lvar}(\Delta, \sigma) \to \mathrm{Lvar}(\Gamma, \sigma) \\
\mathrm{lam} &: (\Gamma : \mathrm{Lcontext}, \sigma : \mathrm{Ltype}, \tau : \mathrm{Ltype}, \rho : \mathrm{Ltype}, p : \mathrm{ar}(\tau, \rho) =_{\mathrm{Lterm}} \sigma) \\
&\quad \to \mathrm{Lterm}(\mathrm{cons}(\Gamma, \tau), \rho) \to \mathrm{Lterm}(\Gamma, \sigma)
\end{aligned}$$

var has the same type as before, and ap has the same type as before, but with arguments σ and τ interchanged.

More examples. There are many more examples of a similar kind. For example, if Formula is the set of formulas of a formal system, then to be a theorem can often be given by a Formula-indexed inductive definition of

$$\mathrm{Theorem} : \mathrm{Formula} \to \mathrm{set}$$

where the axioms and inference rules correspond to introduction rules. An element $d : \mathrm{Theorem}(\phi)$ is a derivation (proof tree) with conclusion ϕ. Further examples are provided by the computation rules in the definition of the operational semantics of a programming language.

Proofs by induction on the structure of an indexed inductive definition are

often called proofs by *rule induction*. Thus, the general form of rule induction is captured by the elimination rule for unrestricted IIRD to be given later.

3.2 Examples of Indexed Inductive-Recursive Definitions

Martin-Löf's computability predicates for dependent types. We shall now turn to proper IIRD. As a first example, we shall formalise the Tait-style computability predicates for dependent types introduced by Martin-Löf [23]. This example was crucial for the historical development of IIRD, since it may be viewed as an early occurrence of the informal notion of an IIRD. In [23] Martin-Löf presents a version of his intuitionistic type theory and proves a normalisation theorem using such Tait-style computability predicates. He works in an informal intuitionistic metalanguage but gives no explicit justification for the meaningfulness of these computability predicates. (Later Aczel [1] has shown how to model a similar construction in classical set theory.) Since the metalanguage is informal the inductive-recursive nature of this definition is implicit. One of the objectives of the current work is indeed to present an extension of Martin-Löf type theory where the inductive-recursive nature of this and other definitions is formalised. In this way, we hope to clarify the reason why it is an acceptable notion from the point of view of intuitionistic meaning explanations in the sense of Martin-Löf [20,22,21].

First, recall that for the case of the simply typed lambda calculus the Tait-computability predicates ϕ_A are predicates on terms of type A which are defined by *recursion* on the structure of A. We read $\phi_A(a)$ as "a is a computable term of type A". To match Martin-Löf's definition [23] we consider here a version where the clause for function types is

- If $\phi_B(b[a])$ for all closed terms a such that $\phi_A(a)$ then $\phi_{A\to B}(\lambda x.b[x])$.

How can we generalise this to dependent types? First, we must assume that we have introduced the syntax of expressions for dependent types including Π-types, with lambda abstraction and application. Now we cannot define ϕ_A for all (type) expressions A but only for those which are "computable types". The definition of ϕ_A has several clauses, such as the following one for Π [23, p. 161]:

> 4.1.1.2. Suppose that ϕ_A has been defined and that $\phi_{B[a]}$ has been defined for all closed terms a of type A such that $\phi_A(a)$. We then define $\phi_{\Pi x:A.B[x]}$ by the following three clauses.
>
> 4.1.1.2.1. If $\lambda x.b[x]$ is a closed term of type $\Pi x : A.B[x]$ and $\phi_{B[a]}(b[a])$ for all closed terms a of type A such that $\phi_A(a)$, then $\phi_{\Pi x:A.B[x]}(\lambda x.b[x])$.
>
> 4.1.1.2.2. ...
>
> 4.1.1.2.3. ...

(We omit the cases 4.1.1.2.2 and 4.1.1.2.3, which express closure under reduction. They are not relevant for the present discussion. Note that the complete definition of the computability predicate also has one case for each of the other type formers of type theory.)

We also note that Martin-Löf does not use the term "A is a computable type" but only states "that ϕ_A has been defined". We can understand Martin-Löf's definition as an indexed inductive-recursive definition by introducing a predicate Φ on expressions, where $\Phi(A)$ stands for "ϕ_A is defined" or "A is a computable type". Moreover, we add a second argument to ϕ so that $\phi_A(p, a)$ means that a is a computable term of the computable type A, where p is a proof that A is computable. Now we observe that we define Φ inductively while we simultaneously recursively define ϕ.

It would be possible to formalise Martin-Löf's definition verbatim, but for simplicity, we shall follow a slightly different version due to C. Coquand [8]. Assume that we have inductively defined the set Exp of expressions and have an operation $\mathrm{Apl} : \mathrm{Exp} \to \mathrm{Exp} \to \mathrm{Exp}$ for the application of one expression to another. Apl is a constructor of Exp and we will write $A\ b$ for $\mathrm{Apl}(A, b)$. There are also additional reduction rules for expressions, like reduction of β-redexes. We then define

$$\begin{aligned}&\Psi : \mathrm{Exp} \to \mathrm{set}\ ,\\ &\psi : (A : \mathrm{Exp}) \to \Psi(A) \to \mathrm{Exp} \to \mathrm{set}\ .\end{aligned}$$

by an Exp-indexed IIRD. Ψ is inductively defined and plays the rôle of U, and ψ is recursively defined and plays the rôle of $T : (A : \mathrm{Exp}, U(A)) \to D[A]$, where $D[A] = \mathrm{Exp} \to \mathrm{set}$ for $A : \mathrm{Exp}$. ψ will depend negatively on Ψ, so this is not a simultaneous inductive definition.

Informally, C. Coquand's variant of 4.1.1.2 and 4.1.1.2.1 above reads as follows (note that the reference to $\lambda x.b[x]$ is replaced by the reference to arbitrary terms b):

- If $\Psi(A)$ and for all expressions a such that $\psi(A, a)$ we have $\Psi(B\ a)$, then $\Psi(\Pi(A, B))$.
- If $\Psi(\Pi(A, B))$ holds according to the previous definition, then $\psi(\Pi(A, B), b)$ holds iff $\psi(B\ a, b\ a)$ holds for all expressions a such that $\psi(A, a)$.

This can be formalised as follows:

$$\begin{aligned}\pi : &(A : \mathrm{Exp}, p : \Psi(A), B : \mathrm{Exp})\\ &\to (q : (a : \mathrm{Exp}, \psi(A, p, a)) \to \Psi(B\, a))\\ &\to \Psi(\Pi(A, B))\ .\end{aligned}$$

Note that q refers negatively to $\psi(A, p, a)$, which is short for $\psi(A, p)(a)$, where $\psi(A, p)$ is the result of the recursively defined function for the second argument p. The corresponding equality rule is

$$\psi(\Pi(A, B), \pi(A, p, B, q), b) = \forall a : \mathrm{Exp}.\forall x : \psi(A, p, a).\psi(B\, a, q(a, x), b\, a) \ .$$

Again, the reader should be aware that we have presented only one crucial case of the complete IIRD in [8]. For instance, there are clauses corresponding to closure under reductions.

Palmgren's higher order universes [27]. This construction generalises Palmgren's super universe [26], which is a universe which is closed under all the usual operators for forming small sets but is as well closed under the operator for universe formation, that is, an operator which accepts an arbitrary family of sets (a "universe") and builds a universe containing that family.

Palmgren [27] shows how to generalise this idea to a universe which is closed under higher order universe operators.

Let us first introduce some abbreviations. Let $\mathcal{O}_k$ be the type of universe operators of order k and $\mathcal{F}_k$ of families of universe operators of order $k : \mathrm{N}_n$ given by the following recursive definitions:

$$\begin{aligned} \mathcal{O}_0 &= \mathrm{set} \\ \mathcal{O}_{k+1} &= \mathcal{F}_k \to \mathcal{F}_k \end{aligned}$$

$$\mathcal{F}_k = (A : \mathrm{set}) \times (A \to \mathcal{O}_k)$$

Note that n is given in advance and that Palmgren defines a family $\mathbf{ML}^n$ of Martin-Löf type theories with universes closed under universe operations of level less than n. It is not possible to internalise the dependence on $n : \mathrm{N}$, since it would require that we could define a family of *types* by recursion on $n : \mathrm{N}$. In Palmgren [27] the definition of $\mathcal{O}_k$ and $\mathcal{F}_k$ are also given by an external recursion on k, but since we included the constant $\mathrm{case}_2^{\mathrm{type}}$ which admits elimination into type for elements of the type $\mathbf{2}$ in our logical framework, we can internalise this recursion on the finite type N_n.

Let $A : \mathrm{N}_{n+1} \to \mathrm{set}$ and $B : (k : \mathrm{N}_{n+1}) \to A_k \to \mathcal{O}_k$ be a family of universe operators. So A_k is an index set for universe operators of level k and B_k is the family of universe operators of level k indexed by A_k.

We shall now give an IIRD of a family

$$\begin{aligned}
&\mathrm{U} : \mathrm{N}_{n+1} \to \mathrm{set} \\
&\mathrm{T} : (k : \mathrm{N}_{n+1}) \to \mathrm{U}_k \to \mathcal{O}_k
\end{aligned}$$

which depends on the parameters A and B. We will make the dependence on parameters implicit in the sequel, so that $\mathrm{U} = \mathrm{U}(A, B)$ and $\mathrm{T} = \mathrm{T}(A, B)$. The idea is that U_k is a set of codes for universe operators of level k with decoding function T_k. Informally, U, T is inductive-recursively generated by the following rules:

- There is one constructor for U_0 corresponding to each of the standard set formers $\Sigma, \Pi, +$, etc.
- There is a code for each index set A_k in U_0.
- There is a code in U_k for each universe operator $B_k(a)$.
- There are two constructors which together encode the application of an operator of level $i+1$ to a family of operators of level i resulting in another family of operators of level i. To see what this means more precisely, note that families of operators of level i are coded by pairs $a : \mathrm{U}_0$ and $b : (\mathrm{T}_0(a) \to \mathrm{U}_i)$. So to each code $f : \mathrm{U}_{i+1}$ for an operator of level $i+1$ and each code (a, b) for a family of level i, we construct another code (a', b') for a family of level i encoding the result of applying the operator encoded by f to the family encoded by (a, b). Note that we need two constructors for this operation: one returns $a' : \mathrm{U}_0$ and the other returns $b'(x') : \mathrm{U}_i$ for $x' : \mathrm{T}_0(a')$.

Let $0 : \mathrm{N}_{n+1}$, and let $\mathrm{inj}, \mathrm{succ} : \mathrm{N}_n \to \mathrm{N}_{n+1}$ be the injections which map a number in N_n to the corresponding number in N_{n+1} and its successor in N_{n+1}, respectively. We have constructors expressing that $(\mathrm{U}_0, \mathrm{T}_0)$ is a universe closed under Π, Σ, $+$ etc. and the corresponding equality rules. Furthermore, we have the following constructors for U (note that $\mathrm{ap}^0_i(f, a, b)$ stands for $\mathrm{ap}^0(i, f, a, b)$, similarly for $\mathrm{ap}^1_i(f, a, b, x)$)

$$\begin{aligned}
&\widehat{\mathrm{A}} : \mathrm{N}_{n+1} \to \mathrm{U}_0 \\
&\widehat{\mathrm{B}} : (k : \mathrm{N}_{n+1}) \to A_k \to \mathrm{U}_k \\
&\mathrm{ap}^0 : (i : \mathrm{N}_n) \to \mathrm{U}_{\mathrm{succ}(i)} \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_{\mathrm{inj}(i)}) \to \mathrm{U}_0 \\
&\mathrm{ap}^1 : (i : \mathrm{N}_n) \to (f : \mathrm{U}_{\mathrm{succ}(i)}) \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_{\mathrm{inj}(i)}) \to \\
&\qquad \mathrm{T}_0(\mathrm{ap}^0_i(f, a, b)) \to \mathrm{U}_{\mathrm{inj}(i)}
\end{aligned}$$

The equations for T are

$$\begin{aligned}
\mathrm{T}_0(\widehat{\mathrm{A}}_k) &= A_k \\
\mathrm{T}_k(\widehat{\mathrm{B}}_k(a)) &= B_k(a) \\
\mathrm{T}_0(\mathrm{ap}^0_i(f, a, b)) &= \pi_0(\mathrm{T}_{\mathrm{succ}(i)}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_{\mathrm{inj}(i)} \circ b \rangle)) \\
\mathrm{T}_{\mathrm{inj}(i)}(\mathrm{ap}^1_i(f, a, b, x)) &= \pi_1(\mathrm{T}_{\mathrm{succ}(i)}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_{\mathrm{inj}(i)} \circ b \rangle))(x)
\end{aligned}$$

As it stands the above is seems not to be an inductive-recursive definition, since the type of ap^1 depends on $\mathrm{T}_0(\mathrm{ap}^0_i(f,a,b))$, which is not the result of applying T to one of the previous arguments of ap^1. However, this type is by the equality rules for T_0 equal to $\pi_0(\mathrm{T}_{\mathrm{succ}(i)}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_{\mathrm{inj}(i)} \circ b\rangle))$, which is of the correct form: we make use of $\mathrm{T}_0(a)$, $\mathrm{T}_{\mathrm{inj}(i)}(b(x))$ and $\mathrm{T}_{\mathrm{succ}(i)}(f)$, which is always T applied to a previous inductive argument. Note that there is no problem in applying $\mathrm{T}_{\mathrm{succ}(i)}(f)$ to $\langle \mathrm{T}_0(a), \mathrm{T}_{\mathrm{inj}(i)} \circ b\rangle$: Once we have applied $\mathrm{T}_{\mathrm{succ}(i)}$ to a previous inductive argument of ap^1, we obtain an element of $\mathcal{O}_{\mathrm{succ}(i)} = \mathcal{F}_i \to \mathcal{F}_i$, which can be applied to arbitrary elements of $\mathcal{F}_i$. If we spell out the type of ap^1 so that it is clear that it forms part of an IIRD, we obtain the following:

$$\begin{aligned}\mathrm{ap}^1 : (i : \mathrm{N}_n) \to (f : \mathrm{U}_{\mathrm{succ}(i)}) \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_{\mathrm{inj}(i)}) \to \\ \pi_0(\mathrm{T}_{\mathrm{succ}(i)}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_{\mathrm{inj}(i)} \circ b\rangle)) \to \mathrm{U}_{\mathrm{inj}(i)}\end{aligned}$$

For more information about higher-order universes the reader is referred to Palmgren [27]. The difference between our version and Palmgren's is that his version is a simultaneous inductive-recursive definition of n universes, whereas ours is an indexed inductive-recursive definition where N_n is the index set. Note also that ours is a *general* IIRD.

Alternatively, we can define a *restricted* IIRD which is closer to Palmgren's by instead defining an *external* sequence of constructors (indexed by $k = 0, \ldots, n$ and $i = 0, \ldots, n-1$):

$$\begin{aligned}&\widehat{\mathrm{A}}_k : \mathrm{U}_0 \\ &\widehat{\mathrm{B}}_k : A_k \to \mathrm{U}_k \\ &\mathrm{ap}^0_i : \mathrm{U}_{i+1} \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_i) \to \mathrm{U}_0 \\ &\mathrm{ap}^1_i : (f : \mathrm{U}_{i+1}) \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_i) \to \mathrm{T}_0(\mathrm{ap}^0_i(f,a,b)) \to \mathrm{U}_i\end{aligned}$$

Again, by using the equality of T_0 the type of ap^1_i is equal to the following, which makes clear that this is a constructor of an IIRD:

$$\begin{aligned}\mathrm{ap}^1_i : (f : \mathrm{U}_{i+1}) \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_i) \to \\ \pi_0(\mathrm{T}_{i+1}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_i \circ b\rangle)) \to \mathrm{U}_i\end{aligned}$$

Note that there are $4n + 2$ constructors rather than the 4 constructors in the first version. Note also the following subtlety: k in A_k is an external index, whereas k in U_k is the corresponding internal index in N_{n+1}. Similarly, i is also used both as internal and external index.

The equations for T are written in our notation in the same way as before, except that $\mathrm{inj}(i), \mathrm{succ}(i)$ are now replaced by i and $i{+}1$, respectively. However, although they *look* the same, they are to be understood differently. We now have $4n + 2$ equations indexed externally by k and i, whereas before we had 4 equations, and k and i were internal variables.

We include Palmgren's construction as an example of a proof-theoretically strong construction which is subsumed by our theory of IIRD. The version, in which elimination is restricted to sets, is conjectured to reach the strength of Kripke-Platek set theory with one recursive Mahlo ordinal. Since U is given by an IIRD it does not appear negatively in any type of its constructors. Note however, that the type of the parameter B refers negatively to set, so if U is considered as a constructor for set with the parameters as arguments, then the type of this constructor refers negatively to set, and we thus have a construction similar in character to the Mahlo universe.

Bove and Capretta's analysis of the termination of nested recursive definitions of functional programs. Bove and Capretta [6] use indexed inductive-recursive definitions in their analysis of the termination of functions defined by nested general recursion. Given such a function f the idea is to simultaneously define a predicate $D(x)$ expressing that $f(x)$ terminates, and a function $f'(x, p)$, which returns the same value as $f(x)$, but has a proof $p : D(x)$ that $f(x)$ terminates as second argument. So, f' is a total function defined on the subset of arguments for which f terminates. The role of f' is to be a version of f which is definable in intuitionistic type theory. For nested recursion the introduction rules for D will refer to f' and thus we have an indexed inductive-recursive definition.

Assume for instance the rewrite rules $f(0) \longrightarrow f(f(1))$, $f(1) \longrightarrow 2$, $f(2) \longrightarrow f(1)$ on the domain $\{0, 1, 2\}$. We now inductive-recursively define the termination predicate D for f. We get one constructor for each rewrite rule: $\mathrm{C}_0 : (p : D(1), q : D(f'(1, p))) \to D(0)$, $\mathrm{C}_1 : D(1)$, $\mathrm{C}_2 : D(1) \to D(2)$. Furthermore, the equality rules for f' are $f'(0, \mathrm{C}_0(p, q)) = f'(f'(1, p), q)$, $f'(1, \mathrm{C}_1) = 2$, $f'(2, \mathrm{C}_2(p)) = f'(1, p)$. This is a proper IIRD, since in the type of C_0 the second argument depends on $f'(1, p)$, where p is the first argument.

3.3 Why Restricted Indexed Inductive-Recursive Definitions?

As already mentioned in the introduction (p. 3) the syntax of case expressions in proof assistants based on dependent type theory is simpler when using *restricted* IIRD rather than *general* IIRD. Restricted IIRD were introduced by Thierry Coquand in the implementation of the proof assistant Half, and

were also used in the Agda system [7], the successor of Half. (Recently, general IIRD have been added as a separate concept to Agda.) If U and T are given by a restricted IIRD, then we can determine the arguments of a constructor for a particular set $\mathrm{U}(i)$ in the family without analysing i. More precisely, if C is a constructor and we write its type in uncurried form, then its type is of the form $((i : I) \times \mathrm{A}(i)) \to \mathrm{U}(i)$.

An example which satisfies this restriction is the accessible part of a relation. We will see in the next subsection how to write the example of trees and forests and Palmgren's higher order universes in restricted form.

It is easier to construct mathematical models of restricted IIRD, since they can be modelled as initial algebras in an I-indexed slice category. Furthermore, domain-theoretic models of restricted IIRD can be given more easily. One of the reasons why some believe that a fully satisfactory understanding of the identity type has not yet been achieved is that complications arise when introducing certain kinds of models, e.g. domain theoretic models, of unrestricted IIRD.

3.4 The definition of Data Types in the Proof Assistant Agda

Agda [7] allows a slightly more general form of restricted IIRD, in which one defines simultaneously finitely many sets inductively by defining different constructors for each of these sets. An example are the trees and forests mentioned above (8). This is not directly an IIRD in restricted form: If we replace Tree and Forest by $\mathrm{Tree}'(\star_0)$ and $\mathrm{Tree}'(\star_1)$ as mentioned above, the type of the constructors is as follows:

$$\begin{aligned} \mathrm{tree} &: \mathrm{Tree}'(\star_1) \to \mathrm{Tree}'(\star_0) \ , \\ \mathrm{nil} &: \mathrm{Tree}'(\star_1) \ , \\ \mathrm{cons} &: \mathrm{Tree}'(\star_0) \to \mathrm{Tree}'(\star_1) \to \mathrm{Tree}'(\star_1) \ . \end{aligned}$$

These types are not of the form $((i : \mathbf{2}) \times \mathrm{A}(i)) \to \mathrm{Tree}'(i)$ as required in restricted IIRD. However, we can easily translate this kind of general IIRD into restricted form. In this example, we replace $\mathrm{tree}, \mathrm{nil}, \mathrm{cons}$ by one constructor

$$\mathrm{C} : ((i : \mathbf{2}) \times \mathrm{A}(i)) \to \mathrm{Tree}'(i) \ ,$$

where $\mathrm{A}(i)$ is defined by case distinction on $i : \mathbf{2}$:

$$\mathrm{A}(\star_0) = \mathrm{Tree}'(\star_1) \ ,$$
$$\mathrm{A}(\star_1) = \mathbf{1} + (\mathrm{Tree}'(\star_0) \times \mathrm{Tree}'(\star_1)) \ .$$

$\mathrm{C}(\langle \star_0, a \rangle)$, $\mathrm{C}(\langle \star_1, \mathrm{inl}(\star) \rangle)$, $\mathrm{C}(\langle \star_1, \mathrm{inr}(\langle a, b \rangle) \rangle)$ now play the rôle of $\mathrm{tree}(a)$, nil and $\mathrm{cons}(a, b)$, respectively. What we have done is to form for each index $i : \mathbf{2}$ the disjoint union $\mathrm{A}(i)$ of the product of the arguments of the constructors with result type $\mathrm{Tree}'(i)$. This method can as well be applied to Palmgren's higher order universes, where the constructors at each index can be determined by case analysis on N_{n+1}.

We will show how to generalise the above to the following situation: Assume that we have $i : \mathrm{I} \Rightarrow \mathrm{J}_i : \mathrm{stype}$, $i : \mathrm{I}, j : \mathrm{J}_i \Rightarrow D[i, j] : \mathrm{type}$. Assume we define inductive-recursively indexed over $i : \mathrm{I}, j : \mathrm{J}_i$

$$\mathrm{U}_{ij} : \mathrm{set} \ , \qquad \mathrm{T}_{ij} : \mathrm{U}_{ij} \to D[i, j] \ .$$

Assume that the constructors of U_{ij} are given as

$$\mathrm{C}_{ik} : (j : \mathrm{J}_i) \to \mathrm{B}_{ijk} \to \mathrm{U}_{ij} \ ,$$

where $k : \mathrm{K}_i$ for some $\mathrm{K}_i : \mathrm{stype}$. Here $\mathrm{B}_{ijk} : \mathrm{stype}$ for $i : \mathrm{I}, j : \mathrm{J}_i, k : \mathrm{K}_i$. So the constructors are in restricted form relative to j, but not relative to i, but for each $i : \mathrm{I}$ the collection of constructors for U_{ij} is given by an index set K_i. Assume that we have the equations

$$\mathrm{T}_{ij}(\mathrm{C}_{ik}(j, b)) = \mathrm{E}_{ik}(j, b) : \mathrm{D}[i, j] \ ,$$

and that the definition of U_{ij} together with T_{ij} forms an instance of a general IIRD. We can simulate this general IIRD by a restricted IIRD by replacing the constructors C_{ik} by one constructor

$$\mathrm{C}' : (i : \mathrm{I}, j : \mathrm{J}_i, k : \mathrm{K}_i, \mathrm{B}_{ijk}) \to \mathrm{U}_{ij}$$

with equality rule

$$\mathrm{T}_{ij}(\mathrm{C}'(i, j, k, b)) = \mathrm{E}_{ik}(j, b) \ .$$

So $(k : \mathrm{K}_i) \times \mathrm{B}_{ijk}$ is the disjoint union of the arguments of all constructors with target type U_{ij}, which generalises the sets $\mathrm{A}(i)$ in the example Tree' above.

One can now interpret the constructors of the original general IIRD into this restricted form by defining $\mathrm{C}_{ik} := (j, b)\mathrm{C}'(i, j, k, b)$. It is an easy exercise to interpret the recursion operator of the general IIRD as well and to see that with this interpretation the equalities required by the rules for the original IIRD hold in the interpreted version.

In this sense restricted IIRD are closely related to the original inductive definition facility of Agda. However, we should add that the correspondence is not precise, since Agda has a more general termination check for functions.

4 Categories for IIRD

4.1 The Category of Indexed Families of Types

As for the non-indexed case, we shall derive a formalisation of IIRD by modelling them as algebras of certain endofunctors in slice categories. Let R be a set of rules for the language of type theory (where each rule is given by a finite set of judgements as premises and one judgement as conclusion). R includes the logical framework used in this article (but not necessarily extensional equality). Let $\mathrm{TT}(R)$ be the type theory generated by the rules in R. The category $\mathbf{Type}(R)$ is the category, the objects of which are A such that $\mathrm{TT}(R)$ proves $A : \mathrm{type}$, and which has as morphisms from A to B terms f such that $\mathrm{TT}(R)$ proves $f : A \to B$. We identify objects A, A' such that $\mathrm{TT}(R)$ proves $A = A' : \mathrm{type}$ and functions $f, f' : A \to B$ such that $\mathrm{TT}(R)$ proves $f = f' : A \to B$. It is easy to see that we obtain a category. Note that this only relies on properties of judgemental equality and holds even if R does not contain any equality set. In particular it doesn't rely on working in extensional type theory.

In order to model I-indexed inductive-recursive definitions, where I is an arbitrary stype, we will use the category $\mathbf{Fam}(R, I)$ of I-indexed families of types. An object of $\mathbf{Fam}(R, I)$ is an I-indexed family of types, that is, an abstracted expression A for which we can prove $i : I \Rightarrow A[i] : \mathrm{type}$ in $\mathrm{TT}(R)$. An arrow from A to B is an I-indexed function, that is, an abstracted expression f for which we can prove $i : I \Rightarrow f[i] : A[i] \to B[i]$ (in $\mathrm{TT}(R)$). Again, we identify A, A' such that we can prove $i : I \Rightarrow A[i] = A'[i] : \mathrm{type}$ and f, f' such that we can prove $i : I \Rightarrow f[i] = f'[i] : B[i] \to C[i]$. Again, it is easy to verify that we obtain a category.

In the following, we will usually omit the argument R in $\mathbf{Fam}(R, I)$ and $\mathbf{Type}(R)$.

If $\mathbf{C}$ is a category and D an object in it, then $\mathbf{C}/D$ is the slice category with objects pairs (A, f), where A is an object of $\mathbf{C}$ and f an arrow $A \to D$, and morphisms from (A, f) to (B, g) are $\mathbf{C}$-morphisms $h : A \to B$ such that $g \circ h = f$. Note that we write, when working on the meta-level, pairs with round brackets. This is different from the notation $\langle a, b\rangle$ for the pair of a and b in the logical framework.

There are two alternative categories in which we can represent pairs (U, T) such that $U : I \to \text{set}$ and $T : (i : I, U(i)) \to D[i]$ (assuming $i : I \Rightarrow D[i] : \text{type}$).

One is $\mathbf{Fam}(I)/D$, where we identify D with $(i)D[i]$, which is therefore an object of $\mathbf{Fam}(I)$. For U, T as given before (U, T) is directly an object of $\mathbf{Fam}(I)/D$.

The other is to model families as fibrations and to use that $\mathbf{Fam}(I)$ is equivalent to $\mathbf{Type}/I$ (provided the rules of extensional equality are part of R). Thus instead of $\mathbf{Fam}(I)/D$ we can equivalently use $\mathbf{Type}/D'$ with $D' := (i : I) \times D[i]$. An object of this category is a type U' together with a function $T' : U' \to D'$. Intuitively, from U', T' we obtain $U(i)$ as the inverse image of i under $\pi_0 \circ T'$. Furthermore, if $u : U(i)$ and $\pi_0(T'(u)) = i$, then we can define $T(i, u) := \pi_1(T'(u)) : D[i]$. Note that in $\mathbf{Type}/D'$ we access the elements of $U(i)$ in an indirect way.

We can thus construct an equivalence of categories (assuming extensional equality):

$$\mathbb{E}^{\text{Fam}\to\text{Type}}_{I,D} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((i : I) \times D[i]) \qquad \text{and}$$
$$\mathbb{E}^{\text{Type}\to\text{Fam}}_{I,D} : \mathbf{Type}/((i : I) \times D[i]) \to \mathbf{Fam}(I)/D \ .$$

The object part of these functors is defined as follows: Assume an element (A, f) of $\mathbf{Fam}(I)/D$. Then $i : I \Rightarrow A[i] : \text{type}$ and $f : (i : I, A[i]) \to D[i]$. We define $\mathbb{E}^{\text{Fam}\to\text{Type}}_{I,D}(A, f) := (A', f')$ where $A' := (i : I) \times A[i] : \text{type}$ and $f' : A' \to ((i : I) \times D[i])$, $f'(\langle i, a\rangle) := \langle i, f(i, a)\rangle$. Assume an element (A, f) of $\mathbf{Type}/((i : I) \times D[i])$. Then $A : \text{type}$, $f : A \to ((i : I) \times D[i])$. We define $\mathbb{E}^{\text{Type}\to\text{Fam}}_{I,D}(A, f) := (A', f')$ where $A' := (i)((a : A) \times (\pi_0(f(a)) =_I i))$, so A' is an element of $\mathbf{Fam}(I)$, and $f' : (i : I, A'[i]) \to D[i]$, $f'(i, \langle a, p\rangle) := \pi_1(f(a))$. We leave it to the reader to work out the morphism parts of these functors and to show that they form an equivalence.

Note that $\mathbb{E}^{\text{Fam}\to\text{Type}}_{I,D}$ and $\mathbb{E}^{\text{Type}\to\text{Fam}}_{I,D}$ are meta-level functions, as are the natural transformations showing that they form an equivalence. For instance the object part of $\mathbb{E}^{\text{Fam}\to\text{Type}}_{I,D}$ cannot be defined as a function inside type theory – we cannot even define its type. All we can do is to associate with every (A, f)

as above a corresponding pair (A', f') as defined before.

General Assumption 4.1 *(a) In the following, we assume* I : stype*,* $i : I \Rightarrow D[i]$: type *(*D *an abstracted expression).*

(b) We will often omit arguments I, D *in functions and constructors in the following, if they are implicitly contained in other arguments, e.g. when one of the arguments is* $\gamma : \mathrm{OP}_{I,D,E}$ *(where* OP *will be introduced below).*

4.2 Coding Several Constructors into One

We can code several constructors of an IIRD into one: let J be a finite index set for all constructors and A_j be the type of the jth constructor (see Appendix A.3 for the definition of J). Then replace all constructors by one constructor of type $(j : J) \to A_j$ which is definable using case-distinction on J. In case of restricted IIRD we can obtain one constructor in restricted form with type $(i : I, j : J) \to A_{ij} \to C_i$, if the type of the jth constructor is $(i : I) \to A_{ij} \to C_i$.

In this way, it will suffice to consider only IIRD with one constructor intro in the sequel.

4.3 Restricted IIRD as Algebras in $\mathbf{Fam}(I)/D$

We already stated that for restricted IIRD, the first argument of the constructor determines the index of the constructed element. By uncurrying the remaining arguments to an element of one set $\mathbb{H}^{\mathrm{U}}(\mathrm{U}, \mathrm{T}, i)$ we get the following general form of a constructor

$$\mathrm{intro} : (i : I) \to \mathbb{H}^{\mathrm{U}}(\mathrm{U}, \mathrm{T}, i) \to \mathrm{U}(i),$$

for certain functions (to be given later)

$$\mathbb{H}^{\mathrm{U}} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i], I) \to \mathrm{stype}$$

with no free occurrences of U, T and i.

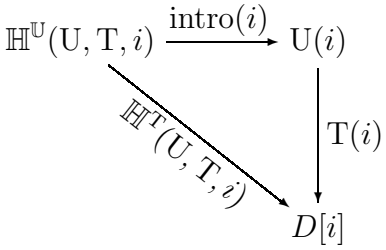
The equality rule for T has the form

$$\mathrm{T}(i, \mathrm{intro}(i, a)) = \mathbb{H}^{\mathrm{T}}(\mathrm{U}, \mathrm{T}, i, a) \ ,$$

for certain functions (also to be given later)

$$\mathbb{H}^{\mathrm{T}} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i], i : I, \mathbb{H}^{\mathrm{U}}(U, T, i)) \to D[i]$$

with no free occurrences of U, T, i or a. We draw the equality rule as a commuting diagram



in the category $\mathbf{Fam}(I)$, that is, the objects $D[i], U(i)$, etc, and the arrows are $i : I$-indexed families. We will later see that, if we assume extensionality, $\mathbb{H}^{\mathrm{U}}, \mathbb{H}^{\mathrm{T}}$ can be extended to work on all families of types (not just on families of sets) so that they together form the components of an endofunctor $\mathbb{H}$ on $\mathbf{Fam}(I)/D$, where $\mathbb{H}(U,T) := (\mathbb{H}^{\mathrm{U}}(U,T), \mathbb{H}^{\mathrm{T}}(U,T))$. [2] Hence (U, T) together with intro will be an $\mathbb{H}$-algebra in $\mathbf{Fam}(I)/D$.

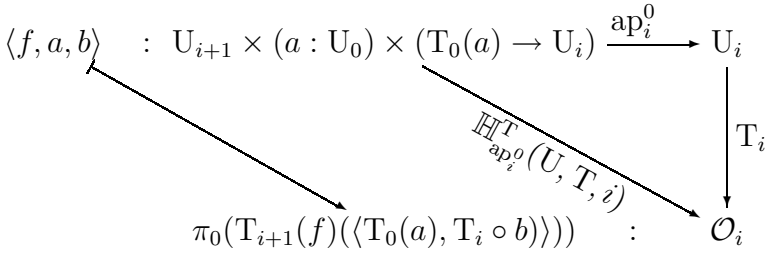
As an example, consider our second formulation of Palmgren's higher-order universe which is a restricted IIRD with $4n+2$ constructors. We only consider the constructor

$$\mathrm{ap}_i^0 : \mathrm{U}_{i+1} \to (a : \mathrm{U}_0) \to (\mathrm{T}_0(a) \to \mathrm{U}_i) \to \mathrm{U}_0$$

where $0 \leq i \leq n-1$. The equality rule is

$$\mathrm{T}_0(\mathrm{ap}_i^0(f,a,b)) = \pi_0(\mathrm{T}_{i+1}(f)(\langle \mathrm{T}_0(a), \mathrm{T}_i \circ b\rangle))$$

We consider the uncurried version of ap_i^0 and draw a diagram:



Hence, we have

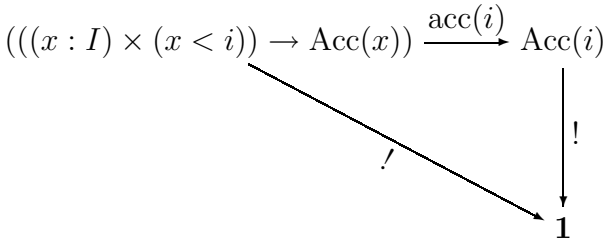
$$\mathbb{H}^{\mathrm{U}}_{\mathrm{ap}_i^0}(U,T,i) := U_{i+1} \times (a : U_0) \times (T_0(a) \to U_i)$$
$$\mathbb{H}^{\mathrm{T}}_{\mathrm{ap}_i^0}(U,T,i,\langle f,a,b\rangle) := \pi_0(T_{i+1}(f)(\langle T_0(a), T_i \circ b)\rangle))$$

[2] To be pedantic: one has to replace $\mathbb{H}(U,T)$, $\mathbb{H}^{\mathrm{U}}(U,T)$, $\mathbb{H}^{\mathrm{T}}(U,T)$ by uncurried variants $\mathbb{H}'((U,T))$, $\mathbb{H}'^{\mathrm{U}}((U,T))$

To define $\mathbb{H}^{\mathrm{U}}(U, T, i)$ for all constructors, we first do case analysis on i and then take the disjoint union of $\mathbb{H}^{\mathrm{U}}_C(U, T, i)$ for all constructors C with codomain U_i.

The diagram for IIRD generalises the situation for non-indexed induction-recursion [16], where the rules for U and T give rise to an algebra of an appropriate endofunctor $\mathbb{F}$ on the slice category $\mathbf{Type}/D$.

If $D[i] := \mathbf{1}$ and therefore $\mathbb{H}(U, T, i)$ does not depend on T, we have the important special case of a restricted indexed inductive definition (IID). Here follows as an example the diagram for the accessible part of a relation (we replace $(x : I) \to (x < i) \to \mathrm{Acc}(x)$ by its uncurried form $((x : I) \times (x < i)) \to \mathrm{Acc}(x)$ in order to obtain the form $(x : B) \to \mathrm{Acc}(x)$ of an inductive argument of an IID):



that is, $\mathbb{H}^{\mathrm{U}}(U, T, i) := ((x : I) \times (x < i)) \to U(x)$.

4.4 General IIRD and Functors from $\mathbf{Type}/((i : I) \times D[i])$ *to* $\mathbf{Fam}(I)/D$

In the general case, the constructor intro of an IIRD has no special first argument which determines the index. Instead, the index $i : I$, such that $\mathrm{intro}(a) : \mathrm{U}(i)$, is a function of the arguments. So the general form of the type of a constructor is

$$\mathrm{intro} : (a : \mathbb{G}^{\mathrm{U}}(\mathrm{U}, \mathrm{T})) \to \mathrm{U}(\mathbb{G}^{\mathrm{I}}(\mathrm{U}, \mathrm{T}, a))$$

for some

$$\mathbb{G}^{\mathrm{U}} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \to \mathrm{stype} \ ,$$
$$\mathbb{G}^{\mathrm{I}} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \to \mathbb{G}^{\mathrm{U}}(U, T, a)) \to I \ .$$

Furthermore, we have

$$\mathrm{T}(\mathbb{G}^{\mathrm{I}}(\mathrm{U}, \mathrm{T}, a), \mathrm{intro}(a)) = \mathbb{G}^{\mathrm{T}}(\mathrm{U}, \mathrm{T}, a)$$

for some

$$\begin{aligned}\mathbb{G}^{\mathrm{T}} &: (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \\ &\to \mathbb{G}^{\mathrm{U}}(U, T, a) \to D[\mathbb{G}^{\mathrm{I}}(U, T, a)] \ ,\end{aligned}$$

We can combine $\mathbb{G}^{\mathrm{I}}$ and $\mathbb{G}^{\mathrm{T}}$ to get one function

$$\begin{aligned}\mathbb{G}^{\mathrm{IT}} &:= (U, T, a)\langle \mathbb{G}^{\mathrm{I}}(U, T, a), \mathbb{G}^{\mathrm{T}}(U, T, a)\rangle \\ &: \ (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \\ &\quad \to \mathbb{G}^{\mathrm{U}}(U, T, a) \to ((i : I) \times D[i]) \ .\end{aligned}$$

Now we see that $\mathbb{G}^{\mathrm{U}}$ and $\mathbb{G}^{\mathrm{IT}}$ together form the two components of a function

$$\begin{aligned}\mathbb{G} &: (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \\ &\to ((U' : \mathrm{stype}) \times (U' \to ((i : I) \times D[i]))) \ .\end{aligned}$$

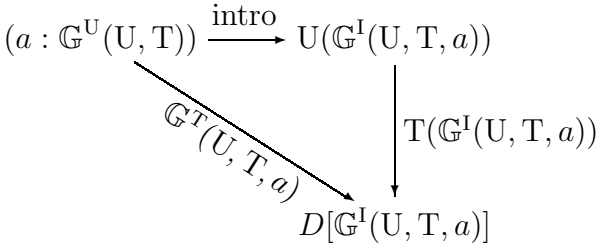
We will later see that this can be extended from sets to types, and that we can prove, if we assume extensionality, that it forms the object part of a functor

$$\mathbb{G} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((i : I) \times D[i]) \ .$$

The equality rule is

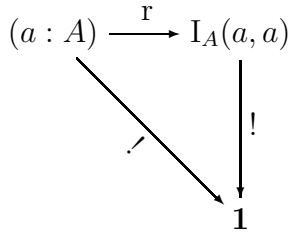
$$\mathrm{T}(\mathbb{G}^{\mathrm{I}}(\mathrm{U}, \mathrm{T}, a), \mathrm{intro}(a)) = \mathbb{G}^{\mathrm{T}}(\mathrm{U}, \mathrm{T}, a) \ ,$$

as expressed by the following "diagram":



In the example of the identity relation, we define $I := A \times A$, $D[i] := \mathbf{1}$, $\mathbb{G}^{\mathrm{U}}(U, T) := A$, $\mathbb{G}^{\mathrm{I}}(U, T, a) := \langle a, a\rangle$, and $\mathrm{U}(\langle a, a\rangle) := \mathrm{I}_A(a, a)$. We obtain the

following "diagram":

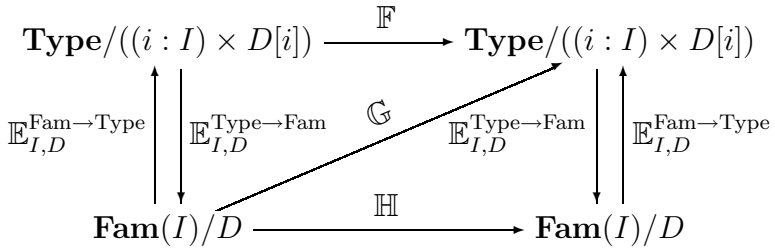


As a second illustration, we show how to obtain the rules for computability predicates for dependent types. (As in Section 3, we only give a definition containing one case, but the complete definition [8] can be obtained by expanding the definition corresponding to the additional constructors).

$$\begin{aligned}
\mathbb{G}^{\mathrm{U}}(\Psi,\psi) &:= (A:\mathrm{Exp})\times(p:\Psi(A))\times \\
&\qquad (B:\mathrm{Exp})\times((a:\mathrm{Exp})\to\psi(A,p,a)\to\Psi(B\ a))\ , \\
\mathbb{G}^{\mathrm{I}}(\Psi,\psi,\langle A,p,B,q\rangle) &:= \Pi(A,B)\ , \\
\mathbb{G}^{\mathrm{T}}(\Psi,\psi,\langle A,p,B,q\rangle) &:= (b)\forall a:\mathrm{Exp}.\forall x:\psi(A,p,a).\psi(B\ a,q(a,x),b\ a)\ .
\end{aligned}$$

4.5 Initial Algebras on $\mathbf{Type}/((i:I)\times D[i])$

If we assume extensional equality, we can extend the operation $\mathbb{G}$ for a general IIRD to a functor $\mathbb{G}:\mathbf{Fam}(I)/D\to\mathbf{Type}/((i:I)\times D[i])$. We can also define the endofunctor $\mathbb{F}:=\mathbb{G}\circ\mathbb{E}_{I,D}^{\mathrm{Type}\to\mathrm{Fam}}$ on $\mathbf{Type}/((i:I)\times D[i])$. If we define $\mathbb{H}:=\mathbb{E}_{I,D}^{\mathrm{Fam}\to\mathrm{Type}}\circ\mathbb{G}$, we also obtain an endofunctor $\mathbb{H}$ which has the same domain and codomain as the endofunctors corresponding to restricted IIRD, but $\mathbb{H}$ will usually not be strictly positive in the sense that it arises from a code for restricted IIRD. So we obtain the following diagram where the two triangles commute and $\mathbb{E}_{I,D}^{\mathrm{Fam}\to\mathrm{Type}}$ and $\mathbb{E}_{I,D}^{\mathrm{Fam}\to\mathrm{Type}}$ form an equivalence of categories:



Instead of considering families of sets introduced by a functor $\mathbb{G}$, we can consider initial algebras of the corresponding functor $\mathbb{F}$ according to the above diagram. This essentially amounts to a reduction of IIRD to non-indexed

inductive-recursive definitions, assuming extensional equality. We plan to prove this in detail in a future paper.

5 Formalising the Theory of IIRD

5.1 A Uniform Theory for Restricted and General IIRD

We now show how to formalise a uniform theory for restricted and general IIRD. To this end, we will introduce a theory which can be instantiated to these two cases. This may seem surprising, since restricted IIRD are naturally viewed as special cases of general IIRD – why not just formalise general IIRD, and then explain the restriction? The reason is that restricted IIRD give rise to an interesting simpler theory which can be defined without reference to general IIRD. Nevertheless, the theory of restricted IIRD shares much structure with the theory of general IIRD. It is therefore more economical to put both under one hat.

Recall that a general IIRD will be given by a functor

$$\mathbb{G} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((x : I) \times D[i])$$

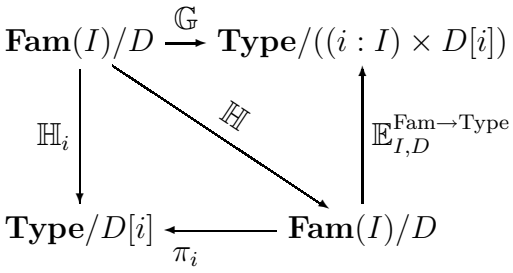
and a restricted IIRD by a functor

$$\mathbb{H} : \mathbf{Fam}(I)/D \to \mathbf{Fam}(I)/D \ .$$

Remark. By a functor, we here mean that we can define the object part of the functor and, assuming extensional equality, that we can also define the morphism part and prove the functor laws. However, only the object part of the functor will be used in the formalisation of the theories of IIRD, and therefore our theories can be used in an intensional setting.

Every element $(U, T) : \mathbf{Fam}(I)/D$ is uniquely determined by its projections $\pi_i(U, T)$. We also note that for every sequence of functors $\mathbb{H}_i : \mathbf{Fam}(I)/D \to \mathbf{Type}/D[i]$ there exists a unique functor $\mathbb{H} : \mathbf{Fam}(I)/D \to \mathbf{Fam}(I)/D$ such that $\pi_i \circ \mathbb{H} = \mathbb{H}_i$. $\mathbb{H}$ and $\mathbb{G}$ will be strictly positive functors in much the same way as $\mathbb{F}$ in [16]. We draw a diagram which summarises the relationship

between these functors:



However, since $\mathbb{H}$ is determined by $\pi_i \circ \mathbb{H}$ and both $\pi_i \circ \mathbb{H}$ and $\mathbb{G}$ are functors $\mathbf{Fam}(I)/D \longrightarrow \mathbf{Type}/E$, (where $E := (i : I) \times D[i]$ in the general case and $E := D[i]$ in the restricted case), it is more economical to introduce the more general notion of a strictly positive functor

$$\mathbb{K} : \mathbf{Fam}(I)/D \longrightarrow \mathbf{Type}/E$$

for an arbitrary type E. From this we can derive the functors $\mathbb{G}$ (by setting $E := (i : I) \times D[i]$) and $\mathbb{H}$ (by defining $\pi_i \circ \mathbb{H}$ using $E := D[i]$).

We proceed as in an earlier article [16] and define the type of indices $\mathrm{OP}_{I,D,E}$ of strictly positive functors

$$\frac{E : \mathrm{type}}{\mathrm{OP}_{I,D,E} : \mathrm{type}}$$

together with

$$\begin{aligned}
&\mathbb{K}^{\mathrm{U}}_{\gamma} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \to \mathrm{stype} \ , \\
&\mathbb{K}^{\mathrm{T}}_{\gamma} : (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i], a : \mathbb{K}^{\mathrm{U}}_{\gamma}(U, T)) \to E \ ,
\end{aligned}$$

for $\gamma : \mathrm{OP}_{I,D,E}$. (We suppress the arguments I, D, E, when the parameter γ is given.) We remark that one could extend the rules below and define for $i : I \Rightarrow U[i] : \mathrm{type}$, $T : (i : I, U[i]) \to D[i]$

$$\begin{aligned}
&\mathbb{K}^{\mathrm{U}}_{\gamma}(U, T) : \mathrm{type} \ , \\
&\mathbb{K}^{\mathrm{T}}_{\gamma}(U, T) : (a : \mathbb{K}^{\mathrm{U}}_{\gamma}(U, T)) \to E \ .
\end{aligned}$$

However, this kind of extension will not play any rôle in our theories of IIRD. We mention it only to be able to consider $\mathbb{K}_{\gamma}$ as a functor which is defined for all elements (U, T) of $\mathbf{Fam}(I)/D$. Assuming extensional equality, we define the morphism part of this functor (but as before this will not be part of our rules): if $i : I \Rightarrow U'[i] : \mathrm{type}$, $T' : (i : I, U'[i]) \to D[i]$, then

$$\begin{aligned}\mathbb{K}^{\mathrm{mor}}_{\gamma}(U,T,U',T') \;:\; & (f : (i : I, U[i]) \to U'[i]) \\ & \to ((i : I, a : U[i]) \to T'(i, f(i,a)) =_{D[i]} T(i,a)) \\ & \to \mathbb{K}^{\mathrm{U}}_{\gamma}(U,T) \to \mathbb{K}^{\mathrm{U}}_{\gamma}(U',T')\end{aligned}$$

We leave it to the reader to verify (by induction on γ) that $\mathbb{K}^{\mathrm{mor}}_{\gamma}(U,T,U',T',f,p)$ will be a morphism (i.e. to verify the corresponding commutative diagram) and that the functor laws hold.

We construct elements of $\mathrm{OP}_{I,D,E}$ in a similar way as in the non-indexed case:

- Base case: This corresponds to having an IIRD with no arguments of the constructor (i.e. the argument is of type $\mathbf{1}$). We only have to determine the result of E, which encodes the result of $\mathrm{T}(\mathrm{intro}(a))$ for restricted IIRD and both this result and the index i such that $\mathrm{intro}(a) : \mathrm{U}(i)$ for general IIRD (Note that we suppress the dependency of ι on I, D, E, similarly for the other constructors of $\mathrm{OP}_{I,D,E}$)

  $$\begin{aligned}\iota \;:\; & E \to \mathrm{OP}_{I,D,E} \;, \\ \mathbb{K}^{\mathrm{U}}_{\iota(e)}(U,T) = & \;\mathbf{1} \;, \\ \mathbb{K}^{\mathrm{T}}_{\iota(e)}(U,T,\star) = & \;e \;,\end{aligned}$$

  and, assuming extensionality

  $$\mathbb{K}^{\mathrm{mor}}_{\iota(e)}(U,T,U',T',f,p,\star) = \star \;.$$

- Non-dependent union of functors: This corresponds to the situation where the constructor has a first non-inductive argument of type A (that is, an argument which does not refer to U) and where the remaining arguments are coded by $\gamma(a)$ which depends on $a : A$.

  $$\begin{aligned}\sigma \;:\; & (A : \mathrm{stype}, \gamma : A \to \mathrm{OP}_{I,D,E}) \to \mathrm{OP}_{I,D,E} \;, \\ \mathbb{K}^{\mathrm{U}}_{\sigma(A,\gamma)}(U,T) = & \;(a : A) \times \mathbb{K}^{\mathrm{U}}_{\gamma(a)}(U,T) \;, \\ \mathbb{K}^{\mathrm{T}}_{\sigma(A,\gamma)}(U,T,\langle a,b\rangle) = & \;\mathbb{K}^{\mathrm{T}}_{\gamma(a)}(U,T,b) \;.\end{aligned}$$

  and, assuming extensionality

  $$\mathbb{K}^{\mathrm{mor}}_{\sigma(A,\gamma)}(U,T,U',T',f,p,\langle a,b\rangle) = \langle a, \mathbb{K}^{\mathrm{mor}}_{\gamma(a)}(U,T,U',T',f,p,b)\rangle \;.$$

- Dependent union of functors: This corresponds to the situation where the constructor has an inductive argument, referring to U. This argument has the form $g : (a : A) \to \mathrm{U}(i(a))$, where $i : A \to I$. Later arguments can depend on T applied to elements of U, that is, on (see Def. A.6 in Appendix

A for the definition of $\mathrm{T} \circ \langle\!\langle i, g \rangle\!\rangle$):

$$\mathrm{T} \circ \langle\!\langle i, g \rangle\!\rangle : (a : A) \to D[i(a)] \ .$$

Therefore the later arguments are given by a function $\gamma : ((a : A) \to D[i(a)]) \to \mathrm{OP}_{I,D,E}$. So the parameters of the constructor for the dependent union of elements of $\mathrm{OP}_{I,D,E}$ are the stype A, the index function i and the function γ. If A, i, γ are given, the inductive argument will be of type $(a : A) \to \mathrm{U}(i(a))$. This argument is followed by the arguments given by $\gamma(\mathrm{T} \circ \langle\!\langle i, g \rangle\!\rangle)$, and the result of E will be determined by the remaining arguments (which depend on $\mathrm{T} \circ \langle\!\langle i, g \rangle\!\rangle$). So we have the following constructor and equations:

$$\delta : (A : \mathrm{stype}, i : A \to I, \gamma : ((a : A) \to D[i(a)]) \to \mathrm{OP}_{I,D,E}) \to \mathrm{OP}_{I,D,E} \ ,$$

$$\mathbb{K}^{\mathrm{U}}_{\delta(A,i,\gamma)}(U, T) = (g : (a : A) \to U(i(a))) \times \mathbb{K}^{\mathrm{U}}_{\gamma(T \circ \langle\!\langle i,g \rangle\!\rangle)}(U, T) \ ,$$

$$\mathbb{K}^{\mathrm{T}}_{\delta(A,i,\gamma)}(U, T, \langle g, b \rangle) = \mathbb{K}^{\mathrm{T}}_{\gamma(T \circ \langle\!\langle i,g \rangle\!\rangle)}(U, T, b) \ ,$$

and, assuming extensionality

$$\mathbb{K}^{\mathrm{mor}}_{\delta(A,i,\gamma)}(U, T, U', T', f, p, \langle g, b \rangle) = \langle f \circ \langle\!\langle i, g \rangle\!\rangle, \mathbb{K}^{\mathrm{mor}}_{\gamma(T \circ \langle\!\langle i,g \rangle\!\rangle)}(U, T, b) \rangle \ .$$

5.2 Formation and Introduction Rules for Restricted IIRD

Restricted IIRD (indicated by a superscript r) are given by strictly positive endofunctors $\mathbb{H}$ on the category $\mathbf{Fam}(I)/D$, which can be given by their (strictly positive) projections $\pi_i \circ \mathbb{H} : \mathbf{Fam}(I)/D \to \mathbf{Type}/D[i]$. So, the set of codes for these functors is given as a family of codes for $\pi_i \circ \mathbb{H}$. The type of codes is given as

$$\mathrm{OP}^{\mathrm{r}}_{I,D} := (i : I) \to \mathrm{OP}_{I,D,D[i]} \ : \ \mathrm{type} \ .$$

Assume now $\gamma : \mathrm{OP}^{\mathrm{r}}_{I,D}$, $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, $i : I$. The object part of $\mathbb{H}$ is defined as

$$\begin{aligned} \mathbb{H}^{\mathrm{U}}_{\gamma}(U, T, i) &:= \mathbb{K}^{\mathrm{U}}_{\gamma(i)}(U, T) : \mathrm{stype} \\ \mathbb{H}^{\mathrm{T}}_{\gamma}(U, T, i, a) &:= \mathbb{K}^{\mathrm{T}}_{\gamma(i)}(U, T, a) : D[i] \end{aligned}$$

for $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, and $a : \mathbb{H}^{\mathrm{U}}_{\gamma}(U, T, i)$.

Using extensionality we obtain the morphism part

$$\mathbb{H}^{\mathrm{mor}}_{\gamma}(U,T,U',T',f,p,i,a) := \mathbb{K}^{\mathrm{mor}}_{\gamma(i)}(U,T,U',T',f,p,a) : \mathbb{H}^{\mathrm{U}}_{\gamma}(U',T',i)$$

We have the following formation rules for $\mathrm{U}^{\mathrm{r}}_{\gamma}$ and $\mathrm{T}^{\mathrm{r}}_{\gamma}$:

$$\mathrm{U}^{\mathrm{r}}_{\gamma}(i) : \mathrm{set} \ , \qquad \mathrm{T}^{\mathrm{r}}_{\gamma}(i) : \mathrm{U}^{\mathrm{r}}_{\gamma}(i) \to D[i] \ .$$

$\mathrm{U}^{\mathrm{r}}_{\gamma}(i)$ has constructor

$$\mathrm{intro}^{\mathrm{r}}_{\gamma}(i) : \mathbb{H}^{\mathrm{U}}_{\gamma}(\mathrm{U}^{\mathrm{r}}_{\gamma}, \mathrm{T}^{\mathrm{r}}_{\gamma}, i) \to \mathrm{U}^{\mathrm{r}}_{\gamma}(i) \ ,$$

and the equality rule for $\mathrm{T}^{\mathrm{r}}_{\gamma}(i)$ is:

$$\mathrm{T}^{\mathrm{r}}_{\gamma}(i, \mathrm{intro}^{\mathrm{r}}_{\gamma}(i,a)) = \mathbb{H}^{\mathrm{T}}_{\gamma}(\mathrm{U}^{\mathrm{r}}_{\gamma}, \mathrm{T}^{\mathrm{r}}_{\gamma}, i, a) \ .$$

5.3 Formation and Introduction Rules for General IIRD

For general IIRD (as indicated by superscript g) we consider strictly positive functors $\mathbb{G}_{\gamma} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((i : I) \times D[i])$ for each code γ in the type of codes for general IIRD, defined as

$$\mathrm{OP}^{\mathrm{g}}_{I,D} := \mathrm{OP}_{I,D,(i:I)\times D[i]} : \mathrm{type} \ .$$

The object part of the functor $\mathbb{G}_{\gamma}$ consists of the following three components:

$$\begin{aligned}
\mathbb{G}^{\mathrm{U}}_{\gamma}(U,T) &:= \mathbb{K}^{\mathrm{U}}_{\gamma}(U,T) : \mathrm{stype} \\
\mathbb{G}^{\mathrm{I}}_{\gamma}(U,T,a) &:= \pi_0(\mathbb{K}^{\mathrm{T}}_{\gamma}(U,T,a)) : I \\
\mathbb{G}^{\mathrm{T}}_{\gamma}(U,T,a) &:= \pi_1(\mathbb{K}^{\mathrm{T}}_{\gamma}(U,T,a)) : D[\mathbb{G}^{\mathrm{I}}_{\gamma}(U,T,a)]
\end{aligned}$$

for $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, and $a : \mathbb{G}^{\mathrm{U}}_{\gamma}(U,T)$.

Using extensionality we obtain the morphism part

$$\mathbb{G}^{\mathrm{mor}}_{\gamma}(U,T,U',T',f,p,a) := \mathbb{K}^{\mathrm{mor}}_{\gamma}(U,T,U',T',f,p,a) : \mathbb{G}^{\mathrm{U}}_{\gamma}(U',T') \ .$$

We have essentially the same formation rules for $\mathrm{U}^{\mathrm{g}}_{\gamma}$ and $\mathrm{T}^{\mathrm{g}}_{\gamma}$ as in the restricted case:

$$\mathrm{U}^{\mathrm{g}}_{\gamma} : I \to \mathrm{set} \ , \qquad \mathrm{T}^{\mathrm{g}}_{\gamma} : (i : I, \mathrm{U}^{\mathrm{g}}_{\gamma}(i)) \to D[i] \ .$$

There is one constructor for all $\mathrm{U}^{\mathrm{g}}_{\gamma}(i)$. The introduction rule is:

$$\mathrm{intro}^{\mathrm{g}}_{\gamma} : (a : \mathbb{G}^{\mathrm{U}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma})) \to \mathrm{U}^{\mathrm{g}}_{\gamma}(\mathbb{G}^{\mathrm{I}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, a)) \ .$$

where $\mathbb{G}^{\mathrm{I}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, a)$ determines the index from the argument of the constructor.

The equality rule for $\mathrm{T}^{\mathrm{g}}_{\gamma}$ is:

$$\mathrm{T}^{\mathrm{g}}_{\gamma}(\mathbb{G}^{\mathrm{I}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, a), \mathrm{intro}^{\mathrm{g}}_{\gamma}(a)) = \mathbb{G}^{\mathrm{T}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, a) \ .$$

5.4 Elimination Rules for IIRD

We now give the induction principle both for the restricted and the general case. We deviate in an important way from our previous formalisation [16] of non-indexed inductive-recursive definitions. In that paper we constructed the following type of induction hypotheses for the elimination and equality rules:

$$\frac{\begin{array}{ccc} D : \mathrm{type} & \gamma : \mathrm{OP}_D & U : \mathrm{set} \\ T : U \to D & a : \mathbb{F}^{\mathrm{U}}_{\gamma}(U, T) & x : U \Rightarrow E[x] : \mathrm{type} \end{array}}{\mathbb{F}^{\mathrm{IH}}_{\gamma}(U, T, E, a) : \mathrm{type}}$$

Moreover, for the recursive call in the equality rule we defined

$$\mathbb{F}^{\mathrm{map}}_{\gamma}(U, T, E, h, a) : \mathbb{F}^{\mathrm{IH}}_{\gamma}(U, T, E, a) \ ,$$

where $h : (u : U) \to E[u]$ and $a : \mathbb{F}^{\mathrm{U}}_{\gamma}(U, T)$.

$\mathbb{F}^{\mathrm{IH}}_{\gamma}(U, T, E, a)$ collects the types $(b : B) \to E[u'(b)]$ of induction hypotheses for each inductive argument of the form $u' : B \to U$ contained in a. $\mathbb{F}^{\mathrm{map}}_{\gamma}(U, T, E, h, a)$ composes these inductive arguments with h and creates an element of the corresponding induction hypothesis $h \circ u' : (b : B) \to E[u'(b)]$. The elimination rule and equality rule were then defined as follows:

$$\frac{g : (a : \mathbb{F}^{\mathrm{U}}_{\gamma}(\mathrm{U}_{\gamma}, \mathrm{T}_{\gamma}), \mathbb{F}^{\mathrm{IH}}_{\gamma}(\mathrm{U}_{\gamma}, \mathrm{T}_{\gamma}, E, a)) \to E[\mathrm{intro}_{\gamma}(a)]}{\mathrm{R}_{\gamma,E}(g) : (u : \mathrm{U}_{\gamma}) \to E[u]}$$

$$\mathrm{R}_{\gamma,E}(g, \mathrm{intro}_{\gamma}(a)) = g(a, \mathbb{F}^{\mathrm{map}}_{\gamma}(\mathrm{U}_{\gamma}, \mathrm{T}_{\gamma}, E, \mathrm{R}_{\gamma,E}(g), a)) \ .$$

As pointed out to us by Ralph Matthes [24], the problem with that approach, is that $\mathbb{F}^{\mathrm{IH}}_{\gamma}(U, T, E, a)$ is a type and cannot be defined by OP-elimination. This led to problems when interpreting theories into each other. (We plan to publish a note in which we elaborate on this and show how to redeem that problem). Since we would get similar problems in this article, we give an alternative

definition. We first define (by OP-elimination)

$$\begin{aligned}\mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a) &: \mathrm{stype}\ ,\\ \mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a) &: \mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a)\to U\ .\end{aligned}$$

$(\mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a),\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a))$ is a family of elements of U, namely those elements in U referred to in a by an inductive argument. $\mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a)$ is obtained as the disjoint union of all B such that an inductive argument $u' : B\to U$ occurs in a. If $b : \mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a)$ originates from $b' : B$, where B is as before, then $\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a)$ maps b to $u'(b') : U$. Now we can define a variant of $\mathbb{F}^{\mathrm{IH}}_{\gamma}$ and $\mathbb{F}^{\mathrm{map}}_{\gamma}$:

$$\begin{aligned}\mathbb{F}^{\mathrm{IH}'}_{\gamma}(U,T,E,a) &:= (v : \mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a))\to E[\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)]\ ,\\ \mathbb{F}^{\mathrm{map}'}_{\gamma}(U,T,E,g,a) &:= (v)g(\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)) : \mathbb{F}^{\mathrm{IH}'}_{\gamma}(U,T,E,a)\ .\end{aligned}$$

In appendix B the variants of the theories for inductive-recursive definitions will be introduced in detail.

In case of *indexed* inductive-recursive definitions, we also need a function which maps elements of $\mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a)$ to the index $i : I$ which the original inductive argument (of the form $u' : (b : B)\to U(i(b))$) was referring to. In general we proceed as follows:

First we define more generally $\mathbb{K}^{\mathrm{IArg}}_{\gamma}$, $\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}$ and $\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}$ for $\gamma : \mathrm{OP}_{I,D,E}$. Assume

$$\begin{aligned}&\gamma : \mathrm{OP}_{I,D,E}\ ,\quad U : I\to\mathrm{set}\ ,\quad T : (i : I, U(i))\to D[i]\ ,\\ &a : \mathbb{K}^{\mathrm{U}}_{\gamma}(U,T)\ .\end{aligned}$$

Then we have the following rules:

$$\begin{aligned}&\mathbb{K}^{\mathrm{IArg}}_{\gamma}(U,T,a) : \mathrm{stype}\ ,\\ &\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}(U,T,a) : \mathbb{K}^{\mathrm{IArg}}_{\gamma}(U,T,a)\to I\ ,\\ &\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a) : (v : \mathbb{K}^{\mathrm{IArg}}_{\gamma}(U,T,a))\to U(\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}(U,T,a,v))\ .\end{aligned}$$

$$\begin{aligned}&\mathbb{K}^{\mathrm{IArg}}_{\iota(e)}(U,T,F,\star) = \mathbf{0}\ ,\\ &\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\iota(e)}(U,T,F,\star,x) = \mathrm{case}_0(\_\,,x)\ ,\\ &\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\iota(e)}(U,T,F,\star,x) = \mathrm{case}_0(\_\,,x)\ .\end{aligned}$$

$$\mathbb{K}^{\mathrm{IArg}}_{\sigma(A,\gamma)}(U,T,\langle a,b\rangle) = \mathbb{K}^{\mathrm{IArg}}_{\gamma(a)}(U,T,b) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\sigma(A,\gamma)}(U,T,\langle a,b\rangle,c) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma(a)}(U,T,b,c) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\sigma(A,\gamma)}(U,T,\langle a,b\rangle,c) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma(a)}(U,T,b,c) \ .$$

$$\mathbb{K}^{\mathrm{IArg}}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle) = A + \mathbb{K}^{\mathrm{IArg}}_{\gamma(T\circ\langle\!\langle i,f\rangle\!\rangle)}(U,T,b) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle,\mathrm{inl}(a)) = i(a) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle,\mathrm{inl}(a)) = f(a) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle,\mathrm{inr}(a)) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma(T\circ\langle\!\langle i,f\rangle\!\rangle)}(U,T,b,a) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle,\mathrm{inr}(a)) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma(T\circ\langle\!\langle i,f\rangle\!\rangle)}(U,T,b,a) \ .$$

We now define for $U : I \to \mathrm{stype}$, $T : (i : I, U(i)) \to D[i]$,
$i : I, u : U \Rightarrow F[i,u] : \mathrm{type}$, $a : \mathbb{K}^{\mathrm{U}}_{\gamma}(U,T)$, $g : (i : I, u : U) \to F[i,u]$,

$$\begin{aligned}\mathbb{K}^{\mathrm{IH}}_{\gamma}(U,T,F,a) :=& (v : \mathbb{K}^{\mathrm{IArg}}_{\gamma}(U,T,a)) \to F[\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}(U,T,a,v), \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)] \\ &: \mathrm{type} \ , \\ \mathbb{K}^{\mathrm{map}}_{\gamma}(U,T,F,g,a) :=& (v)g(\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}(U,T,a,v), \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)) \\ &: \mathbb{K}^{\mathrm{IH}}_{\gamma}(U,T,F,a) \ .\end{aligned}$$

In the restricted case, the elimination and equality rules are

$$\mathrm{R}^{\mathrm{r}}_{\gamma,F}(h) : (i : I, u : \mathrm{U}^{\mathrm{r}}_{\gamma}(i)) \to F[i,u] \ ,$$
$$\mathrm{R}^{\mathrm{r}}_{\gamma,F}(h,i,\mathrm{intro}^{\mathrm{r}}_{\gamma}(i,a)) = h(i,a,\mathbb{K}^{\mathrm{map}}_{\gamma(i)}(\mathrm{U}^{\mathrm{r}}_{\gamma},\mathrm{T}^{\mathrm{r}}_{\gamma},F,\mathrm{R}^{\mathrm{r}}_{\gamma,F}(h),a)) \ ,$$

under the assumptions

$$\gamma : \mathrm{OP}^{\mathrm{r}}_{I,D} \ ,$$
$$i : I, u : \mathrm{U}^{\mathrm{r}}_{\gamma}(i) \Rightarrow F[i,u] : \mathrm{type} \ ,$$
$$h : (i : I, a : \mathbb{H}^{\mathrm{U}}_{\gamma}(\mathrm{U}^{\mathrm{r}}_{\gamma},\mathrm{T}^{\mathrm{r}}_{\gamma},i), \mathbb{K}^{\mathrm{IH}}_{\gamma(i)}(\mathrm{U}^{\mathrm{r}}_{\gamma},\mathrm{T}^{\mathrm{r}}_{\gamma},F,a)) \to F[i,\mathrm{intro}^{\mathrm{r}}_{\gamma}(i,a)] \ .$$

In the general case the elimination and equality rules are

$$\mathrm{R}^{\mathrm{g}}_{\gamma,F}(h) : (i : I, u : \mathrm{U}^{\mathrm{g}}_{\gamma}(i)) \to F[i,u] \ ,$$
$$\mathrm{R}^{\mathrm{g}}_{\gamma,F}(h,\mathbb{G}^{\mathrm{I}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma},\mathrm{T}^{\mathrm{g}}_{\gamma},a),\mathrm{intro}^{\mathrm{g}}_{\gamma}(a)) = h(a,\mathbb{K}^{\mathrm{map}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma},\mathrm{T}^{\mathrm{g}}_{\gamma},F,\mathrm{R}^{\mathrm{g}}_{\gamma,F}(h),a)) \ ,$$

under the assumptions

$$\gamma : \mathrm{OP}^{\mathrm{g}}_{I,D} \ ,$$
$$i : I, u : \mathrm{U}^{\mathrm{g}}_{\gamma}(i) \Rightarrow F[i,u] : \mathrm{type} \ ,$$
$$h : (a : \mathbb{G}^{\mathrm{U}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}), \mathbb{K}^{\mathrm{IH}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, F, a)) \to F[\mathbb{G}^{\mathrm{I}}_{\gamma}(\mathrm{U}^{\mathrm{g}}_{\gamma}, \mathrm{T}^{\mathrm{g}}_{\gamma}, a), \mathrm{intro}^{\mathrm{g}}_{\gamma}(a)] \ .$$

5.5 Elimination Rules for **OP**

Definition 5.1 *The elimination and equality rules for* OP *are (assuming* E : type *and* $\gamma : \mathrm{OP}_{I,D,E} \Rightarrow F[\gamma] : \mathrm{type}$*):*

$$\frac{\begin{array}{l} a : (e : E) \to F[\iota(e)] \\ b : (A : \mathrm{stype}, \gamma : A \to \mathrm{OP}_{I,D,E}, f : (x : A) \to F[\gamma(x)]) \to F[\sigma(A,\gamma)] \\ c : (A : \mathrm{stype}, i : A \to I, \gamma : ((a : A) \to D[i(a)]) \to \mathrm{OP}_{I,D,E}) \\ \quad \to (f : (x : (a : A) \to D[i(a)]) \to F[\gamma(x)]) \\ \quad \to F[\delta(A,i,\gamma)] \end{array}}{\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c) : (\gamma : \mathrm{OP}_{I,D,E}) \to F[\gamma]}$$

$$\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c,\iota(e)) = a(e) \ ,$$
$$\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c,\sigma(A,\gamma)) = b(A,\gamma,(x)\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c,\gamma(x))) \ ,$$
$$\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c,\delta(A,i,\gamma)) = c(A,i,\gamma,(x)\mathrm{R}^{\mathrm{OP}}_{I,D,E,F}(a,b,c,\gamma(x))) \ .$$

We call these rules $\mathbf{OP_{elim}}$*. They presuppose the formation/introduction rules for* OP*.*

5.6 The Resulting Theories

Definition 5.2 *(a)* $\mathbf{OP_{intro}}$ *consists of the logical framework and the formation and introduction rules for* OP*.*

*(b) The basic theory of indexed inductive-recursive definitions (***Bas-IIRD***) consists of* $\mathbf{OP_{intro}}$ *and the defining rules for* $\mathbb{K}^{\mathrm{U}}$*,* $\mathbb{K}^{\mathrm{T}}$*,* $\mathbb{K}^{\mathrm{IArg}}$*,* $\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}$*,* $\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}$*,* $\mathbb{K}^{\mathrm{IH}}$ *and* $\mathbb{K}^{\mathrm{map}}$*.*

(c) The theory $\mathbf{IIRD^r}$ *of restricted indexed inductive-recursive definitions consists of* **Bas-IIRD***, the defining rules for* OP^{r}*,* $\mathbb{H}^{\mathrm{U}}$*,* $\mathbb{H}^{\mathrm{T}}$*, and the formation/introduction/elimination/equality rules for* $\mathrm{U}^{\mathrm{r}}, \mathrm{T}^{\mathrm{r}}, \mathrm{intro}^{\mathrm{r}}$*, and* R^{r}*.*

(d) *The theory* $\mathbf{IIRD^g}$ *of general indexed inductive-recursive definitions consists of* **Bas-IIRD**, *the defining rules for* $\mathrm{OP^g}$, $\mathbb{G}^{\mathrm{U}}$, $\mathbb{G}^{\mathrm{I}}$, $\mathbb{G}^{\mathrm{T}}$, *and the formation/introduction/elimination/equality rules for* $\mathrm{U^g}, \mathrm{T^g}, \mathrm{intro^g}$, *and* $\mathrm{R^g}$.

(e) *By* **ext** *we mean the rules of extensionality.*

Note that we did not include the morphism parts of the functors into our rules. Assuming extensionality, they can be defined by recursion on γ, as long as we restrict ourselves as in the rules above to $U : I \to \mathrm{set}$.

6 The Examples Revisited

We first introduce the following abbreviations:

$$\gamma_0 +_{\mathrm{OP}} \gamma_1 := \sigma(\mathbf{2}, (x)\mathrm{case}_2(\_, x, \gamma_0, \gamma_1)) \ ,$$

and, if $n \geq 2$,

$$\gamma_0 +_{\mathrm{OP}} \cdots +_{\mathrm{OP}} \gamma_n := (\cdots((\gamma_0 +_{\mathrm{OP}} \gamma_1) +_{\mathrm{OP}} \gamma_2) +_{\mathrm{OP}} \cdots +_{\mathrm{OP}} \gamma_n) \ .$$

So, if γ_i are codes for constructors C_i, $\gamma_0 + \gamma_1$ is a code for a constructor C, which encodes C_0 and C_1. The first argument of C encodes an element i of $\{0, 1\}$. The later arguments of C are the arguments of the constructor C_i. Similarly $\gamma_0 +_{\mathrm{OP}} \cdots +_{\mathrm{OP}} \gamma_n$ is a code for a constructor C which encodes the union of the constructors C_i corresponding to γ_i. Let for $i : I$, $\iota^{\mathrm{g}}_\star(i) := \iota(\langle i, \star\rangle) : \mathrm{OP}^{\mathrm{g}}_{I,(-)\mathbf{1}}$, and let $\iota^{\mathrm{r}}_\star := \iota(\star) : \mathrm{OP}_{I,D,\mathbf{1}}$.

- The trees and forests have code $\gamma : \mathrm{OP}^{\mathrm{r}}_{\mathbf{2},(-)\mathbf{1}}$ $(= \mathbf{2} \to \mathrm{OP}_{\mathbf{2},(-)\mathbf{1},\mathbf{1}})$, where

  $\gamma(\star_0) = \delta(\mathbf{1}, (-)\star_1, (-)\iota^{\mathrm{r}}_\star) \ ,$

  $\gamma(\star_1) = \iota^{\mathrm{r}}_\star +_{\mathrm{OP}} \delta(\mathbf{1}, (-)\star_0, (-)\delta(\mathbf{1}, (-)\star_1, (-)\iota^{\mathrm{r}}_\star)) \ .$

  Then $\mathrm{Tree} = \mathrm{U}^{\mathrm{r}}_{\mathbf{2},(-)\mathbf{1},\gamma}(\star_0)$, $\mathrm{Forest} = \mathrm{U}^{\mathrm{r}}_{\mathbf{2},(-)\mathbf{1},\gamma}(\star_1)$ (we don't suppress the arguments $I = \mathbf{2}$, $D = (-)\mathbf{1}$).
- The even number predicate as a general IID as mentioned in the introduction has code

  $\iota^{\mathrm{g}}_\star(0) +_{\mathrm{OP}} \sigma(\mathrm{N}, (n)\delta(\mathbf{1}, (-)n, (-)\iota^{\mathrm{g}}_\star(\mathrm{S}(\mathrm{S}(n))))) : \mathrm{OP}^{\mathrm{g}}_{\mathrm{N},(-)\mathbf{1}}$

  $(= \mathrm{OP}_{\mathrm{N},(-)\mathbf{1},\mathrm{N}\times\mathbf{1}})$.
- The accessible part of a relation has code

  $(i)\delta((x : I) \times (x < i), (z)\pi_0(z), (-)\iota^{\mathrm{r}}_\star) : \mathrm{OP}^{\mathrm{r}}_{I,(-)\mathbf{1}}$

  $(= (i : I) \to \mathrm{OP}_{I,(-)\mathbf{1},\mathbf{1}})$.

  As a general IIRD, it has code

$\sigma(I, (i)\delta((x : I) \times (x < i), (z)\pi_0(z), (-)\iota^{\mathrm{g}}_{\star}(i)))$: $\mathrm{OP}^{\mathrm{g}}_{I,(-)\mathbf{1}}$
$(= \mathrm{OP}_{I,(-)\mathbf{1},I\times\mathbf{1}})$.

- The identity set has code
  $\sigma(A, (a)\iota^{\mathrm{g}}_{\star}(\langle a, a\rangle)) : \mathrm{OP}^{\mathrm{g}}_{A\times A,(-)\mathbf{1}}$ $(= \mathrm{OP}_{A\times A,(-)\mathbf{1},(A\times A)\times\mathbf{1}})$.
- The simply typed lambda calculus (p. 9) given as a general IIRD can be encoded in the following way. First, Lvar can be encoded as an element of $\mathrm{OP}^{\mathrm{g}}_{\mathrm{Lcontext}\times\mathrm{Ltype},(-)\mathbf{1}}$ by:

$$
\begin{aligned}
\gamma_{\mathrm{Lvar}} &:= \gamma^{\mathrm{zVar}}_{\mathrm{Lvar}} +_{\mathrm{OP}} \gamma^{\mathrm{sVar}}_{\mathrm{Lvar}} \\
\gamma^{\mathrm{zVar}}_{\mathrm{Lvar}} &:= \sigma(\mathrm{Lcontext}, (\Gamma)\sigma(\mathrm{Ltype}, (\alpha)\iota^{\mathrm{g}}_{\star}(\langle \mathrm{cons}(\Gamma, \alpha), \alpha\rangle))) \\
\gamma^{\mathrm{sVar}}_{\mathrm{Lvar}} &:= \sigma(\mathrm{Lcontext}, (\Gamma)\sigma(\mathrm{Ltype}, (\alpha)\sigma(\mathrm{Ltype}, (\beta) \\
&\qquad \delta(\mathbf{1}, (-)\langle\Gamma, \beta\rangle, (-)\iota^{\mathrm{g}}_{\star}(\langle \mathrm{cons}(\Gamma, \alpha), \beta\rangle))))))
\end{aligned}
$$

After introducing notations for Lvar and its constructors one can introduce a code for Lterm as an element of $\mathrm{OP}^{\mathrm{g}}_{\mathrm{Lcontext}\times\mathrm{Ltype},(-)\mathbf{1}}$ as follows:

$$
\begin{aligned}
\gamma_{\mathrm{Lterm}} &:= \gamma^{\mathrm{var}}_{\mathrm{Lterm}} +_{\mathrm{OP}} \gamma^{\mathrm{ap}}_{\mathrm{Lterm}} +_{\mathrm{OP}} \gamma^{\mathrm{lam}}_{\mathrm{Lterm}} \\
\gamma^{\mathrm{var}}_{\mathrm{Lterm}} &:= \sigma(\mathrm{Lcontext}, (\Gamma)\sigma(\mathrm{Ltype}, (\alpha)\sigma(\mathrm{Lvar}(\Gamma, \alpha), (-) \\
&\qquad \iota^{\mathrm{g}}_{\star}(\langle\Gamma, \alpha\rangle))))) \\
\gamma^{\mathrm{ap}}_{\mathrm{Lterm}} &:= \sigma(\mathrm{Lcontext}, (\Gamma)\sigma(\mathrm{Ltype}, (\alpha)\sigma(\mathrm{Ltype}, (\beta) \\
&\qquad \delta(\mathbf{1}, (-)\langle\Gamma, \mathrm{ar}(\alpha, \beta)\rangle, (-)\delta(\mathbf{1}, (-)\langle\Gamma, \alpha\rangle, (-) \\
&\qquad \iota^{\mathrm{g}}_{\star}(\langle\Gamma, \beta\rangle))))))) \\
\gamma^{\mathrm{lam}}_{\mathrm{Lterm}} &:= \sigma(\mathrm{Lcontext}, (\Gamma)\sigma(\mathrm{Ltype}, (\alpha)\sigma(\mathrm{Ltype}, (\beta) \\
&\qquad \delta(\mathbf{1}, (-)\langle\mathrm{cons}(\Gamma, \alpha), \beta\rangle, (-) \\
&\qquad \iota^{\mathrm{g}}_{\star}(\langle\Gamma, \mathrm{ar}(\alpha, \beta)\rangle))))))
\end{aligned}
$$

- For the Tait-style computability predicates for dependent types we have $I = \mathrm{Exp}$, $D[i] = \mathrm{Exp} \to \mathrm{set}$. The rules given in Subsection 3.2 are incomplete, additional constructors have to be added by using $+_{\mathrm{OP}}$ (the current definition actually defines the empty set). The code for the constructor given

in Subsection 3.2 is

$$\begin{aligned}
&\sigma(\mathrm{Exp},(A)\\
&\ \delta(\mathbf{1},(-)A,(f)\\
&\ \ \sigma(\mathrm{Exp},(B)\\
&\ \ \ \delta((a:\mathrm{Exp})\times f(\star,a),(y)(B\,\pi_0(y)),(g)\\
&\ \ \ \ \iota(\langle\Pi(A,B),\ (b)\forall a:\mathrm{Exp}.\forall x:f(\star,a).g(\langle a,x\rangle,b\,a)\rangle)))))\\
&:\mathrm{OP}^{\mathrm{g}}_{I,D}\ \ (=\mathrm{OP}_{I,D,(i:I)\times D[i]})\ .
\end{aligned}$$

- The first version of Palmgren's higher order universes given by a general IIRD has code

$$\gamma := \gamma^{\mathrm{g}}_{\mathrm{Univ}} + \gamma_{\widehat{\mathrm{A}}} + \gamma_{\widehat{\mathrm{B}}} + \gamma_{\mathrm{ap}^0} + \gamma_{\mathrm{ap}^1} : \mathrm{OP}^{\mathrm{g}}_{\mathrm{N}_{n+1},(k)\mathcal{O}^k}\ ,$$

where $\gamma^{\mathrm{g}}_{\mathrm{Univ}} : \mathrm{OP}^{\mathrm{g}}_{\mathrm{N}_{n+1},(k)\mathcal{O}^k}$ is a code expressing that $\mathrm{U}_{\gamma^{\mathrm{g}}_{\mathrm{Univ}},0}, \mathrm{T}_{\gamma^{\mathrm{g}}_{\mathrm{Univ}},0}$ is closed under the standard constructors for a universe, and

$$\begin{aligned}
\gamma_{\widehat{\mathrm{A}}} &:= \sigma(\mathrm{N}_{n+1},(k)\iota(\langle 0,A_k\rangle))\ ,\\
\gamma_{\widehat{\mathrm{B}}} &:= \sigma(\mathrm{N}_{n+1},(k)\sigma(A_k,(a)\iota(\langle k,B_k(a)\rangle)))\ ,\\
\gamma_{\mathrm{ap}^0} &:= \sigma(\mathrm{N}_{n+1},(i)\ \delta(\mathbf{1},(-)\mathrm{succ}(i),(T_f)\\
&\qquad\qquad \delta(\mathbf{1},(-)0,(T_a)\\
&\qquad\qquad\ \delta(T_a(\star),(-)\mathrm{inj}(i),(T_b)\\
&\qquad\qquad\ \ \iota(\langle 0,\pi_0(T_f(\star,\langle T_a(\star),T_b\rangle))\rangle)))))\ ,\\
\gamma_{\mathrm{ap}^1} &:= \sigma(\mathrm{N}_{n+1},(i)\ \delta(\mathbf{1},(-)\mathrm{succ}(i),(T_f)\\
&\qquad\qquad \delta(\mathbf{1},(-)0,(T_a)\\
&\qquad\qquad\ \delta(T_a(\star),(-)\mathrm{inj}(i),(T_b)\\
&\qquad\qquad\ \ \sigma(\pi_0(T_f(\star,\langle T_a(\star),T_b\rangle)),(x)\\
&\qquad\qquad\ \ \ \iota(\langle\mathrm{inj}(i),\pi_1(T_f(\star,\langle T_a(\star),T_b\rangle))(x)\rangle)))))\ .
\end{aligned}$$

- The second version of Palmgren's higher order universes given by a restricted IIRD has code

$$\gamma : \mathrm{OP}^{\mathrm{r}}_{\mathrm{N}_{n+1},(k)\mathcal{O}^k}$$

where

$$\begin{aligned}
\gamma(0) &:= \gamma^{\mathrm{r}}_{\mathrm{Univ}} +_{\mathrm{OP}} \gamma_{\widehat{\mathrm{A}}_0} +_{\mathrm{OP}} \cdots +_{\mathrm{OP}} \gamma_{\widehat{\mathrm{A}}_n} +_{\mathrm{OP}} \gamma_{\widehat{\mathrm{B}}_0} +_{\mathrm{OP}} \\
&\qquad \gamma_{\mathrm{ap}^0_0} +_{\mathrm{OP}} \gamma_{\mathrm{ap}^0_1} +_{\mathrm{OP}} \cdots +_{\mathrm{OP}} \gamma_{\mathrm{ap}^0_{n-1}} +_{\mathrm{OP}} \gamma_{\mathrm{ap}^1_0} \ , \\
\gamma(i) &:= \gamma_{\widehat{\mathrm{B}}_i} +_{\mathrm{OP}} \gamma_{\mathrm{ap}^1_i} \qquad (i = 1, \ldots, n-1) \ , \\
\gamma(n) &:= \gamma_{\widehat{\mathrm{B}}_n} \ ,
\end{aligned}$$

$\gamma^{\mathrm{r}}_{\mathrm{Univ}} : \mathrm{OP}^{\mathrm{r}}_{\mathrm{N}_{n+1},(k)\mathcal{O}^k}(0)$ expresses as $\gamma^{\mathrm{g}}_{\mathrm{Univ}}$ above that $\mathrm{U}_{\gamma^{\mathrm{r}}_{\mathrm{Univ}},0}, \mathrm{T}_{\gamma^{\mathrm{r}}_{\mathrm{Univ}},0}$ is closed under the standard constructors for a universe, and

$$\begin{aligned}
\gamma_{\widehat{\mathrm{A}}_k} &:= \iota(A_k) \ , \\
\gamma_{\widehat{\mathrm{B}}_k} &:= \sigma(A_k, (a)\iota(B_k(a))) \ , \\
\gamma_{\mathrm{ap}^0_i} &:= \delta(\mathbf{1}, (-)(i+1), (T_f) \\
&\qquad \delta(\mathbf{1}, (-)0, (T_a) \\
&\qquad\ \delta(T_a(\star), (-)i, (T_b) \\
&\qquad\quad \iota(\pi_0(T_f(\star, \langle T_a(\star), T_b\rangle)))))) \ , \\
\gamma_{\mathrm{ap}^1_i} &:= \delta(\mathbf{1}, (-)(i+1), (T_f) \\
&\qquad \delta(\mathbf{1}, (-)0, (T_a) \\
&\qquad\ \delta(T_a(\star), (-)i, (T_b) \\
&\qquad\quad \sigma(\pi_0(T_f(\star, \langle T_a(\star), T_b\rangle))), (x) \\
&\qquad\quad\ \iota(\pi_1(T_f(\star), \langle T_a(\star), T_b\rangle)(x)))))) \ .
\end{aligned}$$

7 Interpretations between Restricted and General IIRD

7.1 Preliminaries

We assume the rules **ext** of extensionality. For concrete examples, some of these translations can be carried out using only definitional equality. In such examples we usually only have finitely many levels of nesting of OP, whereas in our theory we may introduce infinitely nested (but still well-founded) elements of OP. For instance, assuming $A : \mathrm{N} \to \mathrm{set}$, let for $n : \mathrm{N}$

$$\rho(n) := \sigma(A(0), (-)\sigma(A(1), \cdots \sigma(A(n), (-)\iota^{\mathrm{g}}_{\star}(*)) \cdots)) : \mathrm{OP}^{\mathrm{g}}_{\mathbf{1},(-)\mathbf{1}} \ ,$$

and define $\gamma := \sigma(\mathrm{N}, (n)\rho(n)) : \mathrm{OP}^{\mathrm{g}}_{\mathbf{1},(-)\mathbf{1}}$. $\rho(n)$ has $n+1$ nesting of OP. Therefore γ has infinite nesting. $\mathrm{U}^{\mathrm{g}}_{\gamma}$ has constructor

$$\mathrm{intro}^{\mathrm{g}}_{\gamma}(\langle n, \langle a_0, \langle a_1, \ldots, \langle a_n, \mathbf{1}\rangle \cdots\rangle\rangle\rangle)$$

where $n : \mathrm{N}$, $a_i : A(i)$. So, $\mathrm{intro}^{\mathrm{g}}_{\gamma}$ has an unbounded number of arguments. When translating codes we need to prove certain equalities by recursion on γ which require extensional equality in the presence of unbounded nesting of γ. These equalities often become definitional equalities if the nesting is finite, which is the case for most concrete examples.

We will assume $\mathbf{OP_{elim}}$ in order to be able to carry out various definitions by induction on γ.

For simplicity, we identify set and stype in the sequel. Note that for every stype A there exists a set A' together with functions $f : A \to A'$ and $g : A' \to A$ such that $\forall a : A.g(f(a)) =_A a$ and $\forall a : A'.f(g(a)) =_{A'} a$. Take as A' the set, which is inductively defined with constructor $C : A \to A'$. It is easy to define A', f, g in any of the theories under consideration and prove the above equalities (using intensional or extensional equality).

We will work informally. From the proof it follows that terms of the first language can be interpreted in the second language such that all rules are valid (that is, the conclusion can be derived from the premises). Note that what we achieve is not just a reduction of categorical principles, but the proof theoretic result that the formal theories can be interpreted into each other. This is especially important when translating the elimination rules – we have to make sure that we can translate the elimination rules of one theory into the other. Most of these proofs rely on recursion on OP.

Notations: In this section, we will make use of the Notation A.4 (informal use of equality).

We summarise the results of this section in the following theorem

Theorem 7.1 *(a)* $\mathbf{IIRD^r} + \mathbf{OP_{elim}} + \mathbf{ext}$ *and* $\mathbf{IIRD^g} + \mathbf{OP_{elim}} + \mathbf{ext}$ *can be interpreted into each other.*
(b) The same holds with the restriction of these theories to $D[i] : \mathrm{stype}$.

7.2 Interpretations between Restricted and General IIRD

Informally it is clear that restricted IIRD are special cases of general IIRD. Moreover, we argued in Section 3.3 that general IIRD can be represented by restricted IIRD, provided we have an equality on the index set.

Below we show that the two theories, the one with restricted IIRD and the one with general IIRD can be formally interpreted in each other, provided they are extended with OP-elimination and extensional equality.

Modelling Restricted IIRD by General IIRD. The translation of restricted IIRD into general IIRD is quite simple: in restricted IIRD, we have a constructor intro : $(i : I) \to A \to \mathrm{U}_i$ for some set A. In general IIRD, we have a constructor intro$'$: $(a : A') \to \mathrm{U}'_{i'(a)}$ for some set A' and some function $i' : A' \to I$. So, in order to reduce restricted to general IIRD, we define $A' := (i : I) \times A$ and $i'(a) := \pi_0(a)$. This means turning the special first argument $i : I$ of a restricted IIRD into a similar non-inductive argument of a general IIRD. When we have no argument of the constructor, we have to provide an element of $(i : I) \times D[i]$ instead of an element of $e : D[i]$ as in the restricted case: this element will be $\langle i, e\rangle$.

Categorically, this transformation can be seen as follows: From a functor $\mathbb{H} : \mathbf{Fam}(I)/D \to \mathbf{Fam}(I)/D$ of a restricted IIRD we obtain a functor $\mathbb{G} := \mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D} \circ \mathbb{H} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((i : I) \times D[i])$ of a general IIRD. $\mathbb{G}$ satisfies the following equations (where $\mathbb{H}(U, T, i) = (\mathbb{H}^{\mathrm{U}}(U, T, i), \mathbb{H}^{\mathrm{T}}(U, T, i))$ and $\mathbb{G}(U, T) = (\mathbb{G}^{\mathrm{U}}(U, T), (a)\langle\mathbb{G}^{\mathrm{I}}(U, T, a), \mathbb{G}^{\mathrm{T}}(U, T, a)\rangle))$:

$$\begin{aligned}
\mathbb{G}^{\mathrm{U}}(U, T) &= (i : I) \times \mathbb{H}^{\mathrm{U}}(U, T, i) \ , \\
\mathbb{G}^{\mathrm{I}}(U, T, \langle i, a\rangle) &= i \ , \\
\mathbb{G}^{\mathrm{T}}(U, T, \langle i, a\rangle) &= \mathbb{H}^{\mathrm{T}}(U, T, i, a) \ .
\end{aligned}$$

We are going to show that this translates strictly positive functors for restricted IIRD into strictly positive functors for general IIRD. For every $\gamma : \mathrm{OP}^{\mathrm{r}}_{I,D}$ we define $\gamma^{\wedge} : \mathrm{OP}^{\mathrm{g}}_{I,D}$ such that $\mathbb{G}_{\gamma^{\wedge}} = \mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D} \circ \mathbb{H}_{\gamma}$. (This equality is to be understood componentwise, see Subsection 4.1 for how $\mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D}$ is to be understood type-theoretically.) We first define $\gamma^{\wedge,i} : \mathrm{OP}^{\mathrm{g}}_{I,D}$ for $i : I$, $\gamma : \mathrm{OP}_{I,D,D[i]}$:

$$\begin{aligned}
\iota(e)^{\wedge,i} &= \iota(\langle i, e\rangle) \ , \\
\sigma(A, \gamma)^{\wedge,i} &= \sigma(A, (a)(\gamma(a))^{\wedge,i}) \ , \\
\delta(A, j, \gamma)^{\wedge,i} &= \delta(A, j, (f)(\gamma(f))^{\wedge,i}) \ .
\end{aligned}$$

For $\gamma : \mathrm{OP}^{\mathrm{r}}_{I,D}$ we define

$$\gamma^{\wedge} := \sigma(I, (i)(\gamma(i))^{\wedge,i}) : \mathrm{OP}^{\mathrm{g}}_{I,D} \ .$$

Note that the above amounts to the replacement of the special first argument of a restricted IIRD by a similar non-inductive argument and in the base case (ι) the replacement of the argument e by $\langle i, e\rangle$.

Using extensional equality one easily derives for $i : I$, $\rho : \mathrm{OP}_{I,D,D[i]}$, $\gamma : \mathrm{OP}^{\mathrm{r}}_{I,D}$, $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, $a : \mathbb{K}^{\mathrm{U}}_{\gamma}(U, T)$ the following (the first three equations are shown by induction on ρ):

$$\begin{aligned}
\mathbb{G}^{\mathrm{U}}_{\rho^\wedge,i}(U,T) &= \mathbb{K}^{\mathrm{U}}_{\rho}(U,T) \ , \\
\mathbb{G}^{\mathrm{I}}_{\rho^\wedge,i}(U,T,a) &= i \ , \\
\mathbb{G}^{\mathrm{T}}_{\rho^\wedge,i}(U,T,a) &= \mathbb{K}^{\mathrm{T}}_{\rho}(U,T,a) \ , \\
\mathbb{G}^{\mathrm{U}}_{\gamma^\wedge}(U,T) &= (i : I) \times \mathbb{H}^{\mathrm{U}}_{\gamma}(U,T,i) \ , \\
\mathbb{G}^{\mathrm{I}}_{\gamma^\wedge}(U,T,\langle i,a\rangle) &= i \ , \\
\mathbb{G}^{\mathrm{T}}_{\gamma^\wedge}(U,T,\langle i,a\rangle) &= \mathbb{H}^{\mathrm{T}}_{\gamma}(U,T,i,a) \ .
\end{aligned}$$

The interpretation of restricted IIRD into general IIRD is defined by

$$\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}} := \mathrm{U}^{\mathrm{g}}_{\gamma^\wedge} \ , \qquad \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}} := \mathrm{T}^{\mathrm{g}}_{\gamma^\wedge} \ ,$$

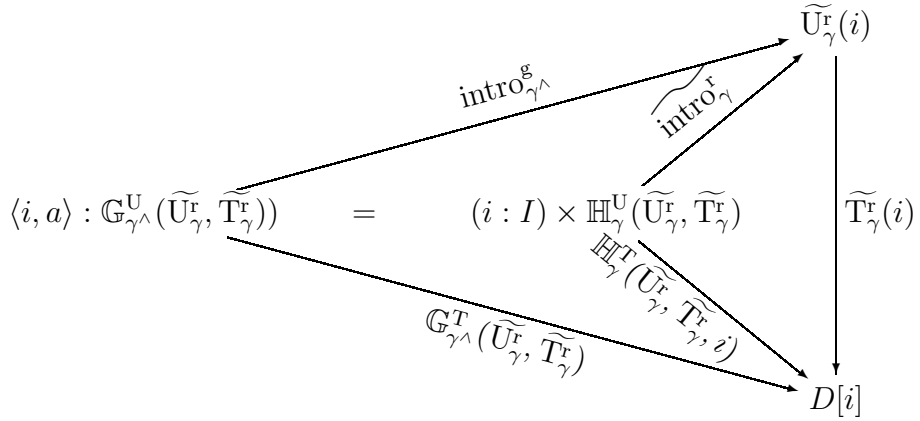
and for $i : I$, $a : \mathbb{H}^{\mathrm{U}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, i)$,

$$\widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}(i,a) := \mathrm{intro}^{\mathrm{g}}_{\gamma^\wedge}(\langle i,a\rangle) : \widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}(i) \ (= \mathrm{U}^{\mathrm{g}}_{\gamma^\wedge}(\mathbb{G}^{\mathrm{I}}_{\gamma^\wedge}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, \langle i,a\rangle))) \ .$$

We have

$$\begin{aligned}
\widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}(\widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}(i,a)) &= \mathrm{T}^{\mathrm{g}}_{\gamma^\wedge}(\mathrm{intro}^{\mathrm{g}}_{\gamma^\wedge}(\langle i,a\rangle)) \\
&= \mathbb{G}^{\mathrm{T}}_{\gamma^\wedge}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, \langle i,a\rangle) \\
&= \mathbb{H}^{\mathrm{T}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, i, a) \ .
\end{aligned}$$

The following "diagram" summarises the relationships:



where we let (for typographical reasons) $\widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}$ represent its uncurried version

$\langle i, a\rangle \mapsto \widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}(i, a)$.

We will now interpret the elimination rules. First, one easily derives by induction over ρ, assuming $i : I$, $\rho : \mathrm{OP}_{I,D,D[i]}$, $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, $a : \mathbb{K}^{\mathrm{U}}_{\rho}(U, T)$, and $v : \mathbb{K}^{\mathrm{IArg}}_{\rho^{\wedge},i}(U, T, a)$ the following equations:

$$\begin{aligned}
\mathbb{K}^{\mathrm{IArg}}_{\rho^{\wedge},i}(U, T, a) &= \mathbb{K}^{\mathrm{IArg}}_{\rho}(U, T, a) \ , \\
\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\rho^{\wedge},i}(U, T, a, v) &= \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\rho}(U, T, a, v) \ , \\
\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\rho^{\wedge},i}(U, T, a, v) &= \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\rho}(U, T, a, v) \ .
\end{aligned}$$

Therefore we obtain for $\gamma : \mathrm{OP}^{\mathrm{g}}_{I,D}$; $i : I$; $a : \mathbb{K}^{\mathrm{U}}_{\gamma^{\wedge}(i)}(U, T)$; $v : \mathbb{K}^{\mathrm{IArg}}_{\gamma^{\wedge}(i)}(U, T, a)$; $i : I, u : U \Rightarrow E[i, u] : \mathrm{type}$; $g : (i : I, u : U(i)) \to E[i, u]$:

$$\begin{aligned}
\mathbb{K}^{\mathrm{IArg}}_{\gamma^{\wedge}}(U, T, \langle i, a\rangle) &= \mathbb{K}^{\mathrm{IArg}}_{\gamma(i)}(U, T, a) \ , \\
\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma^{\wedge}}(U, T, \langle i, a\rangle, v) &= \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma(i)}(U, T, a, v) \ , \\
\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma^{\wedge}}(U, T, \langle i, a\rangle, v) &= \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma(i)}(U, T, a, v) \ , \\
\mathbb{K}^{\mathrm{IH}}_{\gamma^{\wedge}}(U, T, E, \langle i, a\rangle) &= \mathbb{K}^{\mathrm{IH}}_{\gamma(i)}(U, T, E, a) \ , \\
\mathbb{K}^{\mathrm{map}}_{\gamma^{\wedge}}(U, T, E, g, \langle i, a\rangle, v) &= \mathbb{K}^{\mathrm{map}}_{\gamma(i)}(U, T, E, g, a, v) \ .
\end{aligned}$$

Now assume the assumptions of the elimination rules, that is,

$$\begin{aligned}
&i : I, u : \widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}(i) \Rightarrow E[i, u] : \mathrm{type} \ , \\
&h : (i : I, a : \mathbb{H}^{\mathrm{U}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, i), \mathbb{K}^{\mathrm{IH}}_{\gamma(i)}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, E, a)) \to E[i, \widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}(i, a)] \ .
\end{aligned}$$

Define

$$\begin{aligned}
h' &:= (a, b)h(\pi_0(a), \pi_1(a), b) \\
&\quad : (a : \mathbb{G}^{\mathrm{U}}_{\gamma^{\wedge}}(\mathrm{U}^{\mathrm{g}}_{\gamma^{\wedge}}, \mathrm{T}^{\mathrm{g}}_{\gamma^{\wedge}}), \mathbb{K}^{\mathrm{IH}}_{\gamma^{\wedge}}(\mathrm{U}^{\mathrm{g}}_{\gamma^{\wedge}}, \mathrm{T}^{\mathrm{g}}_{\gamma^{\wedge}}, E, a)) \\
&\qquad \to E[\mathbb{G}^{\mathrm{I}}_{\gamma^{\wedge}}(\mathrm{U}^{\mathrm{g}}_{\gamma^{\wedge}}, \mathrm{T}^{\mathrm{g}}_{\gamma^{\wedge}}, a), \mathrm{intro}^{\mathrm{g}}_{\gamma^{\wedge}}(a)] \ .
\end{aligned}$$

Then we can interpret $\mathrm{R}^{\mathrm{r}}_{\gamma,E}(h)$ as

$$k := \mathrm{R}^{\mathrm{g}}_{\gamma^{\wedge},E}(h') : (i : I, u : \widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}(i)) \to E[i, u] \ ,$$

and can verify

$$k(i, \widetilde{\mathrm{intro}^{\mathrm{r}}_{\gamma}}(i, a)) = h(i, a, \mathbb{K}^{\mathrm{map}}_{\gamma(i)}(\widetilde{\mathrm{U}^{\mathrm{r}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{r}}_{\gamma}}, E, k, a)) \ .$$

So, we have shown

Lemma 7.2 $\mathbf{IIRD^r + OP_{elim} + ext}$ *can be interpreted in* $\mathbf{IIRD^g + OP_{elim} + ext}$. *The same holds if we restrict* $D[i]$ *to* stype *or to* $D[i] = \mathbf{1}$ *in both theories.*

Modelling General IIRD by Restricted IIRD. From a functor $\mathbb{G} : \mathbf{Fam}(I)/D \to \mathbf{Type}/((i : I) \times D[i])$ of a general IIRD we obtain a functor $\mathbb{H} := \mathbb{E}^{\mathrm{Type}\to\mathrm{Fam}}_{I,D} \circ \mathbb{G} : \mathbf{Fam}(I)/D \to \mathbf{Fam}(I)/D$. To define $\mathbb{E}^{\mathrm{Type}\to\mathrm{Fam}}_{I,D}$ we need however an extensional equality relation $=_I$ on I. Then $\mathbb{H}$ satisfies the following equations (where as before $\mathbb{H}(U,T,i) = (\mathbb{H}^{\mathrm{U}}(U,T,i), \mathbb{H}^{\mathrm{T}}(U,T,i))$ and $\mathbb{G}(U,T) = (\mathbb{G}^{\mathrm{U}}(U,T), (a)\langle\mathbb{G}^{\mathrm{I}}(U,T,a), \mathbb{G}^{\mathrm{T}}(U,T,a)\rangle))$:

$$\begin{aligned}
\mathbb{H}^{\mathrm{U}} &: (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \to I \to \mathrm{set} \ , \\
\mathbb{H}^{\mathrm{U}}(U,T,i) &= (a : \mathbb{G}^{\mathrm{U}}(U,T)) \times (\mathbb{G}^{\mathrm{I}}(U,T,a) =_I i) \ , \\
\mathbb{H}^{\mathrm{T}} &: (U : I \to \mathrm{set}, T : (i : I, U(i)) \to D[i]) \\
&\quad \to (i : I) \to \mathbb{H}^{\mathrm{U}}(U,T,i) \to D[i] \ , \\
\mathbb{H}^{\mathrm{T}}(U,T,i,\langle a,p\rangle) &= \mathbb{G}^{\mathrm{T}}(U,T,a) \ .
\end{aligned}$$

Note that the constructors have an additional argument, namely a proof of $\mathbb{G}^{\mathrm{I}}(U,T,a) =_I i$. This will add a computational overhead when working with proof assistants.

We show that for every strictly positive functor for a general IIRD the above essentially yields a strictly positive functor for a restricted IIRD, that is, we define for every $\gamma : \mathrm{OP}^{\mathrm{g}}_{I,D}$ a $\gamma^\vee : \mathrm{OP}^{\mathrm{r}}_{I,D}$ such that $\mathbb{H}_{\gamma^\vee}$ is isomorphic to $\mathbb{E}^{\mathrm{Type}\to\mathrm{Fam}}_{I,D} \circ \mathbb{G}_\gamma$. (We do not obtain equality. Assume for simplicity that γ does not impose a variable number of arguments and that therefore $\mathbb{G}^{\mathrm{U}}_\gamma(U,T)$ can be written as $A_1 \times A_2 \times \cdots \times A_n$. Then we will get $\mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(U,T,i) = A_1 \times (A_2 \times \cdots \times (A_n \times \mathbb{G}^{\mathrm{I}}(U,T,a) =_I i))$ instead of $(A_1 \times A_2 \times \cdots \times A_n) \times (\mathbb{G}^{\mathrm{I}}(U,T,a) =_I i)$.)

First, we define for $\gamma : \mathrm{OP}^{\mathrm{g}}_{I,D}$ $\gamma^{\vee,i} : \mathrm{OP}_{I,D,D[i]}$ as follows:

$$\begin{aligned}
\iota(\langle i', e\rangle)^{\vee,i} &= \sigma(i' =_I i, (-)\iota(e)) \ , \\
\sigma(A,\gamma)^{\vee,i} &= \sigma(A, (a)(\gamma(a))^{\vee,i}) \ , \\
\delta(A,j,\gamma)^{\vee,i} &= \delta(A,j,(f)(\gamma(f))^{\vee,i}) \ .
\end{aligned}$$

Now let for $\gamma : \mathrm{OP}^{\mathrm{g}}_{I,D}$

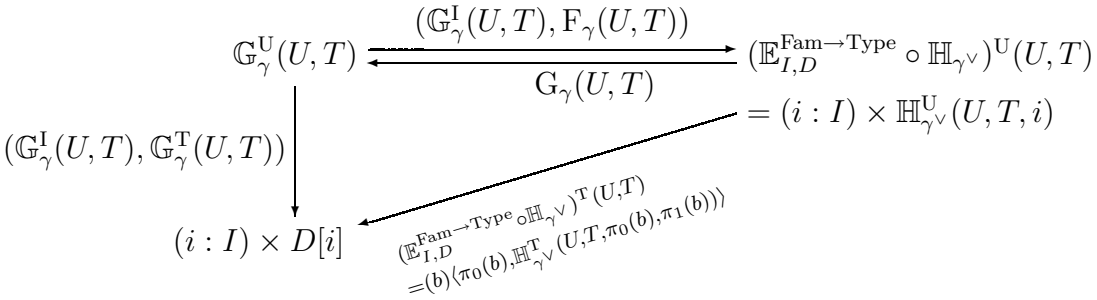
$$\gamma^\vee := (i)\gamma^{\vee,i} : \mathrm{OP}^{\mathrm{r}}_{I,D} \ .$$

$\mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D}$ and $\mathbb{E}^{\mathrm{Type}\to\mathrm{Fam}}_{I,D}$ form an equivalence, and $\mathbb{H}_{\gamma^\vee}$ is isomorphic to $\mathbb{E}^{\mathrm{Type}\to\mathrm{Fam}}_{I,D} \circ \mathbb{G}_\gamma$ Therefore, $\mathbb{G}_\gamma$ is isomorphic to $\mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D} \circ \mathbb{H}_{\gamma^\vee}$. In order to interpret $\mathbf{IIRD^g + OP_{elim} + ext}$ in $\mathbf{IIRD^r + OP_{elim} + ext}$, we need to

introduce such an isomorphism in $\mathbf{IIRD^r + OP_{elim} + ext}$. Therefore, we define in this theory

$$\begin{aligned}
&\mathrm{F}_\gamma(U,T) : (a : \mathbb{G}^{\mathrm{U}}_\gamma(U,T)) \to \mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(U,T,\mathbb{G}^{\mathrm{I}}_\gamma(U,T,a)) \ , \\
&\mathrm{G}_\gamma(U,T) : (i : I) \to \mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(U,T,i) \to \mathbb{G}^{\mathrm{U}}_\gamma(U,T) \ .
\end{aligned}$$

We then show that (we use here the notation (f,g) introduced in Def. A.7 in Appendix A) $(\mathbb{G}^{\mathrm{I}}_\gamma, \mathrm{F}_\gamma) : \mathbb{G}_\gamma \dot{\to} \mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D} \circ \mathbb{H}_{\gamma^\vee}$ and $\mathrm{G}_\gamma : \mathbb{E}^{\mathrm{Fam}\to\mathrm{Type}}_{I,D} \circ \mathbb{H}_{\gamma^\vee} \dot{\to} \mathbb{G}_\gamma$ are natural isomorphisms as expressed by the following diagram [3]:



F_γ and G_γ are defined as follows:

$$\begin{aligned}
\mathrm{F}_{\iota(e)}(U,T,\star) &= \langle \mathrm{ref}, \star \rangle \ , \\
\mathrm{F}_{\sigma(A,\gamma)}(U,T,\langle a,b\rangle) &= \langle a, \mathrm{F}_{\gamma(a)}(U,T,b)\rangle \ , \\
\mathrm{F}_{\delta(A,i,\gamma)}(U,T,\langle f,b\rangle) &= \langle f, \mathrm{F}_{\gamma(T\circ\langle\!\langle i,f\rangle\!\rangle)}(U,T,b)\rangle \ ,
\end{aligned}$$

$$\begin{aligned}
\mathrm{G}_{\iota(e)}(U,T,i,\langle p,\star\rangle) &= \star \ , \\
\mathrm{G}_{\sigma(A,\gamma)}(U,T,i,\langle a,b\rangle) &= \langle a, \mathrm{G}_{\gamma(a)}(U,T,i,b)\rangle \ , \\
\mathrm{G}_{\delta(A,i,\gamma)}(U,T,i,\langle f,b\rangle) &= \langle f, \mathrm{G}_{\gamma(T\circ\langle\!\langle i,f\rangle\!\rangle)}(U,T,i,b)\rangle \ .
\end{aligned}$$

The commutativity of the above diagram is expressed by the following equations, which can be shown by induction on γ (assuming $U : I \to \mathrm{set}$, $T : (i : I, U(i)) \to D[i]$, $a : \mathbb{G}^{\mathrm{U}}_\gamma(U,T)$, $i : I$, $b : \mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(U,T,i)$):

$$\begin{aligned}
\mathrm{G}_\gamma(U,T,\mathbb{G}^{\mathrm{I}}_\gamma(U,T,a),\mathrm{F}_\gamma(U,T,a)) &= a \ , \\
\mathbb{G}^{\mathrm{I}}_\gamma(U,T,\mathrm{G}_\gamma(U,T,i,b)) &= i \ ,
\end{aligned}$$

[3] More precisely, in the previous equations and in the diagram below we refer to an uncurried version of $\mathrm{G}_\gamma(U,T)$ of type $((i : I) \times \mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(U,T,i)) \to \mathbb{G}^{\mathrm{U}}_\gamma(U,T)$. Furthermore, if A is an object of the slice category $\mathbf{Type}/((i : I) \times D[i])$, then we denote its two components by $A^{\mathrm{U}} : \mathbf{Type}$ and $A^{\mathrm{T}} : A^{\mathrm{U}} \to ((i : I) \times D[i])$.

$$\mathbb{G}^{\mathrm{T}}_{\gamma}(U,T,\mathrm{G}_{\gamma}(U,T,i,b)) = \mathbb{H}^{\mathrm{T}}_{\gamma^{\vee}}(U,T,i,b) \ ,$$
$$\mathrm{F}_{\gamma}(U,T,\mathrm{G}_{\gamma}(U,T,i,b)) = b \ .$$

One can verify that we have obtained natural transformations, but that fact will not be needed in our interpretation.

From the above equations follows

$$\mathbb{H}^{\mathrm{T}}_{\gamma^{\vee}}(U,T,\mathbb{G}^{\mathrm{I}}_{\gamma}(U,T,a),\mathrm{F}_{\gamma}(U,T,a)) = \mathbb{G}^{\mathrm{T}}_{\gamma}(U,T,a) \ .$$

Now define

$$\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}} := \mathrm{U}^{\mathrm{r}}_{\gamma^{\vee}} \ , \qquad \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}} := \mathrm{T}^{\mathrm{r}}_{\gamma^{\vee}} \ ,$$

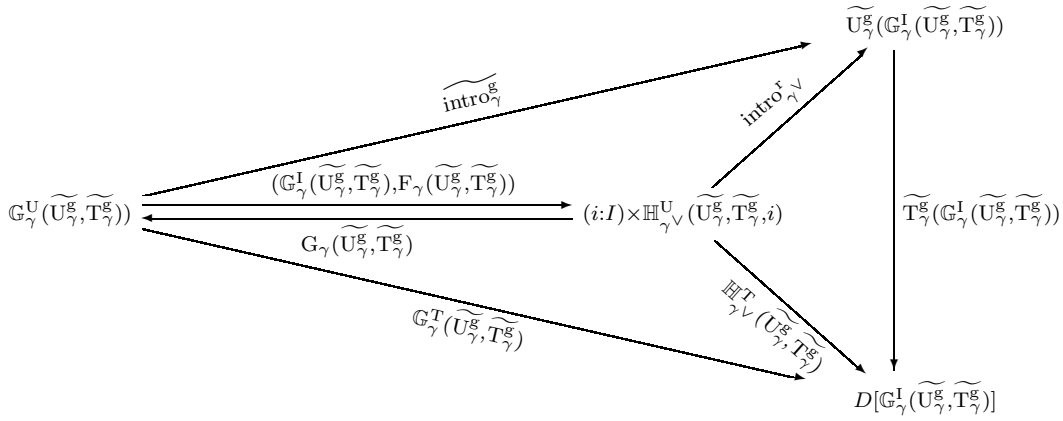
and for $a : \mathbb{G}^{\mathrm{U}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}})$,

$$\widetilde{\mathrm{intro}^{\mathrm{g}}_{\gamma}}(a) := \mathrm{intro}^{\mathrm{r}}_{\gamma^{\vee}}(\mathbb{G}^{\mathrm{I}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}, a), \mathrm{F}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}, a)) : \widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}(\mathbb{G}^{\mathrm{I}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}, a)) \ .$$

We easily obtain

$$\widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}(\mathbb{G}^{\mathrm{I}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}, a), \widetilde{\mathrm{intro}^{\mathrm{g}}_{\gamma}}(a)) = \mathbb{G}^{\mathrm{T}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}, a) \ .$$

The above is summarised by the following diagram [4]:



By induction on γ we can easily derive:

$$\mathbb{K}^{\mathrm{IArg}}_{\gamma^{\vee}(i)}(U,T,a) = \mathbb{K}^{\mathrm{IArg}}_{\gamma}(U,T,\mathrm{G}_{\gamma}(U,T,i,a)) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma^{\vee}(i)}(U,T,a,v) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}}_{\gamma}(U,T,\mathrm{G}_{\gamma}(U,T,i,a),v) \ ,$$
$$\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma^{\vee}(i)}(U,T,a,v) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,\mathrm{G}_{\gamma}(U,T,i,a),v) \ .$$

[4] Again we are using uncurried versions of $\mathrm{intro}^{\mathrm{r}}_{\gamma^{\vee}}, \mathbb{H}^{\mathrm{T}}_{\gamma^{\vee}}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}}), \mathrm{G}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}})$ and $\mathbb{G}^{\mathrm{T}}_{\gamma}(\widetilde{\mathrm{U}^{\mathrm{g}}_{\gamma}}, \widetilde{\mathrm{T}^{\mathrm{g}}_{\gamma}})$.

Therefore, assuming $i : I, u : U(i) \Rightarrow E[i,u]$: type, $g : (i : I, u : U(i)) \to E[i,u]$, and $a : \mathbb{K}^{\mathrm{U}}_{\gamma^\vee(i)}(U,T)$, we get

$$\begin{aligned}\mathbb{K}^{\mathrm{IH}}_{\gamma^\vee(i)}(U,T,E,a) &= \mathbb{K}^{\mathrm{IH}}_{\gamma}(U,T,E,\mathrm{G}_\gamma(U,T,i,a)) \ , \\ \mathbb{K}^{\mathrm{map}}_{\gamma^\vee(i)}(U,T,E,g,a) &= \mathbb{K}^{\mathrm{map}}_{\gamma}(U,T,E,g,\mathrm{G}_\gamma(U,T,i,a)) \ .\end{aligned}$$

For interpreting the elimination rules assume that

$$\begin{aligned}&i : I, a : \widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}(i) \Rightarrow E[i,a] : \text{type} \ , \\ &h : (a : \mathbb{G}^{\mathrm{U}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}), \mathbb{K}^{\mathrm{IH}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, E, a)) \to E[\mathbb{G}^{\mathrm{I}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, a), \widetilde{\mathrm{intro}^{\mathrm{g}}_\gamma(a)}] \ .\end{aligned}$$

Define

$$\begin{aligned}h' &:= (i,a,v)h(\mathrm{G}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, i, a), v) \\ &\quad : (i : I, a : \mathbb{H}^{\mathrm{U}}_{\gamma^\vee}(\mathrm{U}^{\mathrm{r}}_{\gamma^\vee}, \mathrm{T}^{\mathrm{r}}_{\gamma^\vee}, i), \mathbb{K}^{\mathrm{IH}}_{\gamma^\vee(i)}(\mathrm{U}^{\mathrm{r}}_{\gamma^\vee}, \mathrm{T}^{\mathrm{r}}_{\gamma^\vee}, E, a)) \to E[i, \mathrm{intro}^{\mathrm{r}}_{\gamma^\vee}(i,a)] \ .\end{aligned}$$

Note that the type of $h'(i,a,v)$ is

$$\begin{aligned}&E[\mathbb{G}^{\mathrm{I}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, \mathrm{G}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, i, a)), \\ &\quad \mathrm{intro}^{\mathrm{r}}_{\gamma^\vee}(\mathbb{G}^{\mathrm{I}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, \mathrm{G}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, i, a)), \mathrm{F}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, \mathrm{G}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, i, a)))] \\ &= E[i, \mathrm{intro}^{\mathrm{r}}_{\gamma^\vee}(i,a)] \ .\end{aligned}$$

Now we can interpret $\mathrm{R}^{\mathrm{g}}_{\gamma,E}(h)$ as

$$k := \mathrm{R}^{\mathrm{r}}_{\gamma^\vee,E}(h') : (i : I, u : \widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}(i)) \to E[i,u] \ ,$$

and obtain

$$k(\mathbb{G}^{\mathrm{I}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, a), \widetilde{\mathrm{intro}^{\mathrm{g}}_\gamma(a)}) = h(a, \mathbb{K}^{\mathrm{map}}_\gamma(\widetilde{\mathrm{U}^{\mathrm{g}}_\gamma}, \widetilde{\mathrm{T}^{\mathrm{g}}_\gamma}, E, k, a)) \ .$$

So, we have shown

Lemma 7.3 $\mathbf{IIRD^g + OP_{elim} + ext}$ *can be interpreted in* $\mathbf{IIRD^r + OP_{elim} + ext}$. *The same holds if we restrict* $D[i]$ *to* type *or to* $D[i] = \mathbf{1}$ *in both theories.*

8 A Model for IIRD$^{\mathbf{g}}$

We will prove the consistency of the theory **IIRD$^{\mathbf{g}}$** by constructing a set-theoretic model of it.

We modify the full set-theoretic model in Dybjer and Setzer [14]. The reader is referred to that paper for more information, since we lack space to repeat all definitions here.

The general setting is as given in [14] (like interpretation of the function type as the full function space, of set and stype as $\mathrm{set}^* := \mathrm{stype}^* := \mathrm{V_M}$ for some Mahlo cardinal M, of type as $\mathrm{type}^* := \mathrm{V}_\Lambda$, and the interpretation of the logical framework). However, in this article Λ will be the first inaccessible cardinal above M and we omit all statements about λ_n. This is necessary since $\mathrm{OP}_{I,D,E}$ now refers to families of types D. One could probably avoid this by constructing a more refined model, but our model uses too much strength anyway. We conjecture that the ωth admissible above a *recursive* Mahlo ordinal suffices in order to build a model.

In the model one uses the standard term and type constructors and elimination constants from the underlying type theory with their arities. The set of raw terms is then defined as the set of expressions formed using constructors, elimination constants, application, abstraction and variables. The interpretation A^*_ρ will be given for all raw terms A and all environments ρ referring to arbitrary raw terms, independent of whether $A : \mathrm{type}$ or $A : B$ is derivable or not. Therefore A^*_ρ might be undefined (which means it will be a special set used for undefinedness), written as $A^*_\rho \uparrow$, or defined, in which case we write $A^*_\rho \downarrow$. Consequently, equalities will be usually partial equalities, written as $\simeq$, where $A \simeq B$ means that $A \downarrow \Leftrightarrow B \downarrow$ and that if $A \downarrow$, $B \downarrow$ then A and B are the same set. By $A :\simeq B$ we mean that we define A in such a way that $A \simeq B$. It will be part of the soundness theorem, that whenever we can derive in the type theory in question $x_1 : A_1, \ldots, x_n : A_n \Rightarrow A : \mathrm{type}$ or $x_1 : A_1, \ldots, x_n : A_n \Rightarrow A : B$ and $a_i : A_i[x_1 := a_1, \ldots, x_{i-1} := a_{i-1}]$, then with $\rho := [x_1 := a_1, \ldots, x_n := a_n]$ we will have that $A^*_\rho \downarrow$.

The interpretation of SP, arg and map has to be replaced by interpretations of the new constructions. So, we interpret

- $(\mathrm{OP}_{I,D,E})^*_\rho :\simeq \mathrm{OP}^*_{I^*_\rho, \lambda x \in I^*_\rho . D[x]^*_\rho, E^*_\rho}$,
- $\sigma(A,\gamma)^*_\rho :\simeq \sigma^*(A^*_\rho, \gamma^*_\rho)$,
- similarly for the other new constructors,

where for $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$, $E \in \mathrm{type}^*$ we define $\mathrm{OP}^*_{I,D,E}$ as the least

fixed point of the operator

$$\Psi(X) = E + \Sigma_{A\in \mathrm{set}^*}(A \to X) + \Sigma_{A\in \mathrm{set}^*}\Sigma_{i\in(A\to I)}.((\Pi_{a\in A}.D(i(a))) \to X) \ .$$

By the inaccessibility of Λ, we have $I, D, E \in \mathrm{V}_\alpha$ for some $\alpha < \Lambda$, and therefore there exists a regular cardinal $\kappa < \Lambda$ such that for all $A \in \mathrm{set}^*$ and $i \in (A \to I)$ we have that the cardinality of E, A, $\Pi_{a\in A}D(i(a))$ is $< \kappa$. The fixed point is obtained by iterating Ψ up to κ. Since Λ is an inaccessible cardinal, this solution is an element of $\mathrm{V}_\Lambda = \mathrm{type}^*$.

The interpretations of ι, σ, δ, and ι^*, σ^* and δ^* are similar to analogous definitions of nil, nonind, ind in [14], and we define $(\mathrm{OP}^{\mathrm{g}}_{I,D})^*_\rho :\simeq \mathrm{OP}^{\mathrm{g},*}_{I^*_\rho,(i)D[i]^*_\rho}$ with $\mathrm{OP}^{\mathrm{g},*}_{I,D} := \mathrm{OP}^*_{I,D,E}$, where $E := \Sigma i : I.D[i]$.

We define $(\mathbb{K}^{\mathrm{U}}_\gamma(U,T))^*_\rho :\simeq \mathbb{K}^{\mathrm{U}*}_{\gamma^*_\rho}(U^*_\rho, T^*_\rho)$, similarly for the other operations on $\mathbb{K}$, where we define for $\gamma \in \mathrm{OP}^{\mathrm{g},*}_{I,D}$, $I \in \mathrm{type}^*$, $D \in I \to \mathrm{type}^*$, $E \in \mathrm{type}^*$ by recursion on γ the set $\mathbb{K}^{\mathrm{U}*}_{I,D,E,\gamma}(U,T) \in \mathrm{stype}^*$ as follows (if γ, I, D, E are not of this form, we have $\mathbb{K}^{\mathrm{U}*}_{I,D,E,\gamma}(U,T)\uparrow$; a similar proviso applies to all future definitions in this section):

$$\begin{aligned}
\mathbb{K}^{\mathrm{U}*}_{I,D,E,\iota^*(e)}(U,T) &= \mathbf{1}^* \ , \\
\mathbb{K}^{\mathrm{U}*}_{I,D,E,\sigma^*(A,\gamma)}(U,T) &= \Sigma_{a\in A}\mathbb{K}^{\mathrm{U}*}_{I,D,E,\gamma(a)}(U,T) \ , \\
\mathbb{K}^{\mathrm{U}*}_{I,D,E,\delta^*(A,i,\gamma)}(U,T) &= \Sigma_{f\in\Pi_{a\in A}U(i(a))} \ \mathbb{K}^{\mathrm{U}*}_{I,D,E,\gamma(\lambda x\in A.T(i(x))(f(x)))}(U,T) \ .
\end{aligned}$$

In a similar way, we can define $\mathbb{K}^{\mathrm{T}*}_{I,D,E,\gamma}$, $\mathbb{K}^{\mathrm{IArg}*}_{I,D,E,\gamma}$, $\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}$, $\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}$, and then define $\mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}$, $\mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}$, $\mathbb{G}^{\mathrm{T}*}_{I,D,\gamma}$, $\mathbb{K}^{\mathrm{IH}*}_{I,D,E,\gamma}$, $\mathbb{K}^{\mathrm{map}*}_{I,D,E,\gamma}$ in an obvious way. Now we interpret

$$\begin{aligned}
(\mathrm{U}_{I,D,\gamma}(i))^*_\rho &:\simeq \mathrm{U}^{\mathrm{M}}_{I^*_\rho,(i)D[i]^*_\rho,\gamma^*_\rho}(i^*_\rho) \ , \\
(\mathrm{T}_{I,D,\gamma}(i,x))^*_\rho &:\simeq \mathrm{T}^{\mathrm{M}}_{I^*_\rho,(i)D[i]^*_\rho,\gamma^*_\rho}(i^*_\rho, x^*_\rho) \ ,
\end{aligned}$$

where we define for $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$, $\gamma \in \mathrm{OP}^{\mathrm{g},*}_{I,D}$, $i \in I$ by recursion on γ simultaneously $\mathrm{U}^\alpha_{I,D,\gamma}(i) \in \mathrm{stype}^*$ and $\mathrm{T}^\alpha_{I,D,\gamma}(i) : \mathrm{U}^\alpha_{I,D,\gamma}(i) \to \mathrm{type}^*$, (written more briefly as $\mathrm{U}^\alpha(i)$, $\mathrm{T}^\alpha(i,x)$) as follows:

$$\begin{aligned}
\mathrm{U}^\alpha(i) &= \{x \mid x \in \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}, \mathrm{T}^{<\alpha}) \wedge \\
&\qquad\qquad \mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}, \mathrm{T}^{<\alpha}, x) = i\} \ , \\
\mathrm{T}^\alpha(i,x) &= \mathbb{G}^{\mathrm{T}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}, \mathrm{T}^{<\alpha}, x) \ .
\end{aligned}$$

with $U^{<\alpha} := \lambda i \in I.\bigcup_{\beta<\alpha} U^\beta(i)$, $T^{<\alpha} := \lambda i \in I.\bigcup_{\beta<\alpha} T^\beta(i)$.

We interpret $\mathrm{intro}^{\mathrm{g}}$ using $\mathrm{intro}^{\mathrm{g},*}_{\gamma}(a) := a$ and R^{g} as $\mathrm{R}^{\mathrm{g},*}$, where for $\alpha \in \mathrm{Ord}$ and $x \in \mathrm{U}^{<\alpha}_{\gamma}(i)$ we define $\mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g,i,x)$ by recursion on α in such a way that the definition is independent on α. Let $\mathrm{R}^{\mathrm{g},<\alpha}_{\gamma,F}(g,i,x)$ be the restriction of $\mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g,i,x)$ to $x \in \mathrm{U}^{<\alpha}_{\gamma}(i)$. Assume $x \in \mathrm{U}^{\alpha}_{\gamma}(i)$. If $x \in \mathrm{U}^{<\alpha}_{\gamma}(i)$, then

$$\mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g,i,x) :\simeq \mathrm{R}^{\mathrm{g},<\alpha}_{\gamma,F}(g,i,x) \ .$$

Otherwise we have $x \in \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma})$ and $i = \mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x)$. Let

$$\begin{aligned}
J &:= \mathbb{K}^{\mathrm{IArg}*}_{I,D,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x) \ , \\
\text{for } a \in J \quad j(a) &:= \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x, a) \ , \\
\text{for } a \in J \quad u(a) &:= \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x, a) \ , \\
c &:= \lambda a : J.\mathrm{R}^{\mathrm{g},<\alpha}_{\gamma,F}(g, j(a), u(a)) \ , \\
\mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g,i,x) &:\simeq g(x,c) \ .
\end{aligned}$$

We can now state a similar soundness theorem as in [14]:

Theorem 8.1 (Soundness theorem)

(a) *If* $\vdash \Gamma$ context, *then* $\Gamma^* \downarrow$.
(b) *If* $\vdash \Gamma \Rightarrow A : E$, *where* $E \equiv \mathrm{type}$ *or* E *is a term, then* $\Gamma^* \downarrow$, $\forall \rho \in \Gamma^*.A^*_\rho \downarrow \wedge A^*_\rho \in E^*_\rho$, *and if* $E \not\equiv \mathrm{type}$, $\forall \rho \in \Gamma^*.E^*_\rho \downarrow \wedge E^*_\rho \in \mathrm{type}^*$.
(c) *If* $\vdash \Gamma \Rightarrow A = B : E$, *where* $E \equiv \mathrm{type}$ *or* E *is a term, then* $\Gamma^* \downarrow$, $\forall \rho \in \Gamma^*(A^*_\rho \downarrow \wedge A^*_\rho \in E^*_\rho \wedge B^*_\rho = A^*_\rho)$, *and if* $E \not\equiv \mathrm{type}$, $\forall \rho \in \Gamma^*.E^*_\rho \downarrow \wedge E^*_\rho \in \mathrm{type}^*$.
(d) $\not\vdash a : \mathbf{0}$.

The only difficulty is to show that $\mathrm{U}^*(i) \in \mathrm{set}^*$, that $\mathrm{U}^*(i)$ is closed under $\mathrm{intro}^{\mathrm{g},*}$, and that R^* is total and fulfils the equality rules. We introduce the following abbreviations for $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$:

(a) $\mathrm{Fam}(I,D) := \{(U,T) \mid U \in I \to \mathrm{set}^* \wedge T \in \Pi_{i\in I}(U(i) \to D(i))\}$.
(b) If $(U,T),(U',T') \in \mathrm{Fam}(I,D)$, then $(U,T) \leq_{I,D} (U',T') \Leftrightarrow \forall i \in I.(U(i) \subseteq U'(i) \wedge T(i') \restriction U(i) = T(i))$.

The analogue of Lemma 1 in [14] is now as follows:

Lemma 1 *Assume* $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$, $\gamma \in \mathrm{OP}^{\mathrm{g}*}_{I,D}$, $(U,T),(U',T') \in \mathrm{Fam}(I,D)$, $(U,T) \leq_{I,D} (U',T')$, $a \in \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U,T)$, $E := \Sigma i : I.D[i]$. *Then*
(a) $\mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U,T) \subseteq \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U',T')$.
(b) $\mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(U',T') \restriction \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U,T) = \mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(U,T)$.
(c) $\mathbb{G}^{\mathrm{T}*}_{I,D,\gamma}(U',T') \restriction \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U,T) = \mathbb{G}^{\mathrm{T}*}_{I,D,\gamma}(U,T)$.

(d) $\mathbb{K}^{\mathrm{IArg}*}_{I,D,E,\gamma}(U',T') \restriction \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U,T) = \mathbb{K}^{\mathrm{IArg}*}_{I,D,E,\gamma}(U,T)$.
(e) $\mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}(U',T',a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}(U,T,a)$.
(f) $\mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}(U',T',a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}(U,T,a)$.

As in [14], we define by induction on $\gamma \in \mathrm{OP}^{\mathrm{g}*}_{I,D}$ sets $\mathrm{Aux}_{I,D,\gamma}(U,T)$ by

$$\begin{aligned}
\mathrm{Aux}_{I,D,\iota^*(e)}(U,T) &= \mathbf{1} \ , \\
\mathrm{Aux}_{I,D,\sigma^*(A,\gamma)}(U,T) &= \Pi_{x\in A}\mathrm{Aux}_{I,D,\gamma(x)}(U,T) \ , \\
\mathrm{Aux}_{I,D,\delta^*(A,j,\gamma)}(U,T) &= A + \Pi_{f\in\Pi_{x\in A}U(j(x))}\mathrm{Aux}_{I,D,\gamma(T\circ f)}(U,T) \ .
\end{aligned}$$

The analogue of Lemma 2 in [14] is as follows:

Lemma 2 *Assume $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$, $E := \Sigma_{i\in I}D(i)$, $\gamma \in \mathrm{OP}^{\mathrm{g}*}_{I,D}$. Let κ be an inaccessible cardinal and let for $\alpha < \kappa$, $(U^\alpha, T^\alpha) \in \mathrm{Fam}(I,D)$ such that for $\alpha < \beta < \kappa$ we have $(U^\alpha, T^\alpha) \leq_{I,D} (U^\beta, T^\beta)$. Assume for some $\alpha_0 < \kappa$ and for all $\alpha_0 \leq \alpha < \kappa$*

$$\mathrm{Aux}_{I,D,\gamma}(U^\alpha, T^\alpha) \in \mathrm{V}_\kappa \ .$$

Then

$$\mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U^{<\kappa}, T^{<\kappa}) = \bigcup_{\alpha<\kappa} \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(U^\alpha, T^\alpha) \ .$$

Note that Lemma 2 does not hold if we assume $I \in \mathrm{type}^*$, so indexed induction-recursion over a proper type as index set is not covered by this model.

The proof is similar to the proof in [14] (one just lets $\phi = \delta^*(A,j,\gamma)$ and replaces $f : A \to U^{<\beta}$ by $f \in \Pi_{a\in A}U^{<\beta}(j(a))$).

The analogue of [14], Lemma 3 can be stated as follows:

Lemma 3 *Assume $I \in \mathrm{set}^*$, $D \in I \to \mathrm{type}^*$, $E := \Sigma_{i\in I}D(i)$, $\gamma \in \mathrm{OP}^{\mathrm{g}*}_{I,D}$. Abbreviate $\mathrm{U}^\alpha(i) := \mathrm{U}^\alpha_{I,D,\gamma}(i)$, $\mathrm{T}^\alpha(i,x) := \mathrm{T}^\alpha_{I,D,\gamma}(i,x)$, and note that $\mathrm{U}^{\mathrm{U}*}_{I,D,\gamma}(i) = \mathrm{U}^{\mathrm{M}}(i)$, $\mathrm{T}^{\mathrm{U}*}_{I,D,\gamma}(i,x) = \mathrm{T}^{\mathrm{M}}(i,x)$. Then the following holds:*

(a) *For $\alpha < \mathrm{M}$ $(\mathrm{U}^\alpha, \mathrm{T}^\alpha) \in \mathrm{Fam}(I,D)$.*
(b) *If $\alpha < \beta$, then $(\mathrm{U}^\alpha, \mathrm{T}^\alpha) \leq_{I,D} (\mathrm{U}^\beta, \mathrm{T}^\beta)$.*
(c) *There exists $\kappa < \mathrm{M}$ such that $\mathrm{U}^\alpha = \mathrm{U}^{\mathrm{M}}$ (and therefore $\mathrm{T}^\alpha = \mathrm{T}^{\mathrm{M}}$) for all $\alpha > \kappa$.*
(d) $\mathrm{U}^{\mathrm{M}} \in \mathrm{V}_{\mathrm{M}}$
(e) *For all $x \in \mathbb{G}^{\mathrm{U}*}(I,D,\gamma,\mathrm{U}^{\mathrm{M}},\mathrm{T}^{\mathrm{M}})$, $x \in \mathrm{U}^{\mathrm{M}}(\mathbb{G}^{\mathrm{I}*}(I,D,\gamma,\mathrm{U}^{\mathrm{M}},\mathrm{T}^{\mathrm{M}},x))$.*

The proof is similar to the proof of Lemma 3 in [14]. The chain of equivalences

now reads:

$$\begin{aligned} x \in \mathrm{U}^{\kappa}(i) &\Leftrightarrow x \in \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(\mathrm{U}^{<\kappa}, \mathrm{T}^{<\kappa}) \wedge \mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(\mathrm{U}^{<\kappa}, \mathrm{T}^{<\kappa}, x) = i \\ &\Leftrightarrow \exists \alpha < \kappa.(x \in \mathbb{G}^{\mathrm{U}*}_{I,D,\gamma}(\mathrm{U}^{\alpha}, \mathrm{T}^{\alpha}) \wedge \mathbb{G}^{\mathrm{I}*}_{I,D,\gamma}(\mathrm{U}^{\alpha}, \mathrm{T}^{\alpha}, x) = i) \\ &\Leftrightarrow \exists \alpha < \kappa.x \in \mathrm{U}^{\alpha+1}(i) \\ &\Leftrightarrow x \in \mathrm{U}^{<\kappa}(i) \ . \end{aligned}$$

The remaining proof of Lemma 3 follows as in [14].

By Lemma 3 (e) it follows that U^{M} is closed under $\mathrm{intro}^{\mathrm{g},*}$. The totality of R^* follows from its definition above (which was by induction on α) and verifying (using Lemma 1) that all definitions given there result in defined values and the constructions are elements of the interpretations of their types, even when referring intermittently to $\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}$ instead of $\mathrm{U}^{\mathrm{M}}_{\gamma}, \mathrm{T}^{\mathrm{M}}_{\gamma}$: We use the variables $x, a, u(a), j(a)$ as above. We have $x \in \mathrm{U}^{\mathrm{M}}_{\gamma}(i)$, $J = \mathbb{K}^{\mathrm{IArg}*}_{I,D,\gamma}(\mathrm{U}^{\mathrm{M}}_{\gamma}, \mathrm{T}^{\mathrm{M}}_{\gamma}, x)$, for $a \in J$, $j(a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}(\mathrm{U}^{\mathrm{M}}_{\gamma}, \mathrm{T}^{\mathrm{M}}_{\gamma}, x, a) \in I$, but also $j(a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{I}*}_{I,D,E,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x, a)$, $u(a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}(\mathrm{U}^{\mathrm{M}}_{\gamma}, \mathrm{T}^{\mathrm{M}}_{\gamma}, x, a)$, but as well $u(a) = \mathbb{K}^{\mathrm{IArg}\to\mathrm{U}*}_{I,D,E,\gamma}(\mathrm{U}^{<\alpha}_{\gamma}, \mathrm{T}^{<\alpha}_{\gamma}, x, a) \in \mathrm{U}^{<\alpha}_{\gamma}(j(a))$, therefore (using that by definition $\mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g) \restriction \mathrm{U}^{<\alpha}_{\gamma} = \mathrm{R}^{\mathrm{g},<\alpha}_{\gamma,F}(g)$) $c = \mathbb{K}^{\mathrm{map}*}_{I,D,E,\gamma}(\mathrm{U}^{\mathrm{M}}_{\gamma}, \mathrm{T}^{\mathrm{M}}_{\gamma}, F, \mathrm{R}^{\mathrm{g},*}_{\gamma,F}(g), x)$, which is an element of the interpretation of its associated type. Therefore $g(a, c) \in F[i, \mathrm{intro}^{\mathrm{g},*}_{\gamma}(a)]$.

The correctness of the interpretation of the equality rules for $\mathrm{T}^{\mathrm{M}}_{\gamma}$ and for $\mathrm{R}^{\mathrm{g}}_{\gamma,F}$ follows easily.

9 Acknowledgements

We would like to thank the anonymous referee for valuable comments on previous versions of this paper. We thank as well Ralph Matthes for discovering [24] an error in a previous article [16] about IRD. As a consequence, we have modified the corresponding constructions for IIRD, too.

An earlier and much shorter version of this paper appeared in LNCS 2183 [15].

A Appendix: The Logical Framework

A.1 Complete Rules of the Logical Framework

In this article we usually do not write out the whole contexts in rules. So, for $n \geq 1$ a rule

$$\frac{\Delta_1 \Rightarrow \theta_1 \qquad \cdots \qquad \Delta_n \Rightarrow \theta_n}{\Delta \Rightarrow \theta}$$

stands for

$$\frac{\Gamma, \Delta_1 \Rightarrow \theta_1 \qquad \cdots \qquad \Gamma, \Delta_n \Rightarrow \theta_n}{\Gamma, \Delta \Rightarrow \theta}$$

and a rule without premises $\Delta \Rightarrow \theta$ stands for $\dfrac{\Gamma \text{ context}}{\Gamma, \Delta \Rightarrow \theta}$

The only exception are the context and assumption rules.

Context- and Assumption-rules

$$\emptyset \text{ context} \qquad\qquad \frac{\Gamma \text{ context} \qquad \Gamma \Rightarrow A : \text{type}}{\Gamma, x : A \text{ context}}$$

$$\frac{\Gamma \text{ context} \qquad \Gamma \Rightarrow A : \text{type}}{\Gamma, x : A \Rightarrow x : A} \qquad\qquad \frac{\Gamma \Rightarrow x : A \qquad \Gamma \Rightarrow B : \text{type}}{\Gamma, y : B \Rightarrow x : A} \quad (\text{if } x \neq y,\ y \notin \text{FV}(A))$$

Rules Relating type, stype, set

$$\text{set} : \text{type} \qquad \text{stype} : \text{type} \qquad \frac{A : \text{set}}{A : \text{stype}}$$

$$\frac{A : \text{stype}}{A : \text{type}} \qquad \frac{A = B : \text{set}}{A = B : \text{stype}} \qquad \frac{A = B : \text{stype}}{A = B : \text{type}}$$

Equality Rules

$$\frac{a : A}{a = a : A} \qquad \frac{A : \text{type}}{A = A : \text{type}}$$

$$\frac{a = b : A}{b = a : A} \qquad \frac{A = B : \text{type}}{B = A : \text{type}}$$

$$\frac{a = b : A \qquad b = c : A}{a = c : A} \qquad \frac{A = B : \text{type} \qquad B = C : \text{type}}{A = C : \text{type}}$$

$$\frac{a : A \qquad A = B : \text{type}}{a : B} \qquad \frac{a = b : A \qquad A = B : \text{type}}{a = b : B}$$

Rules for $\rightarrow$

$$\frac{A : \text{stype} \qquad x : A \Rightarrow B : \text{stype}}{(x : A) \rightarrow B : \text{stype}} \qquad \frac{x : A \Rightarrow B : \text{type}}{(x : A) \rightarrow B : \text{type}}$$

$$\frac{A = A' : \text{stype} \qquad x : A \Rightarrow B = B' : \text{stype}}{(x : A) \rightarrow B = (x : A') \rightarrow B' : \text{stype}}$$

$$\frac{A = A' : \text{type} \qquad x : A \Rightarrow B = B' : \text{type}}{(x : A) \rightarrow B = (x : A') \rightarrow B' : \text{type}}$$

$$\frac{x : A \Rightarrow t : B}{(x : A)t : (x : A) \to B}$$

$$\frac{x : A \Rightarrow t = t' : B}{(x : A)t = (x : A)t' : (x : A) \to B}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad t : (x : A) \to B \qquad s : A}{t(s) : B[x := s]}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad t = t' : (x : A) \to B \qquad s = s' : A}{t(s) = t'(s') : B[x := s]}$$

$$\frac{x : A \Rightarrow r : B \qquad s : A}{((x : A)r)(s) = r[x := s] : B[x := s]}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad s : (x : A) \to B}{s = (x : A)s(x) : (x : A) \to B}$$

Rules for $\times$

$$\frac{A : \text{stype} \qquad x : A \Rightarrow B : \text{stype}}{(x : A) \times B : \text{stype}} \qquad \frac{x : A \Rightarrow B : \text{type}}{(x : A) \times B : \text{type}}$$

$$\frac{A = A' : \text{stype} \qquad x : A \Rightarrow B = B' : \text{stype}}{(x : A) \times B = (x : A') \times B' : \text{stype}}$$

$$\frac{A = A' : \text{type} \qquad x : A \Rightarrow B = B' : \text{type}}{(x : A) \times B = (x : A') \times B' : \text{type}}$$

$$\frac{r : A \qquad s : B[x := r] \qquad x : A \Rightarrow B : \text{type}}{\langle r, s \rangle : (x : A) \times B}$$

$$\frac{r = r' : A \qquad s = s' : B[x := r] \qquad x : A \Rightarrow B : \text{type}}{\langle r, s \rangle = \langle r', s' \rangle : (x : A) \times B}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad r : (x : A) \times B}{\pi_0(r) : A}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad r = r' : (x : A) \times B}{\pi_0(r) = \pi_0(r') : A}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad r : (x : A) \times B}{\pi_1(r) : B[x := \pi_0(r)]}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad r = r' : (x : A) \times B}{\pi_1(r) = \pi_1(r') : B[x := \pi_0(r)]}$$

$$\frac{r : A \qquad s : B[x := r] \qquad x : A \Rightarrow B : \text{type}}{\pi_0(\langle r, s\rangle) = r : A}$$

$$\frac{r : A \qquad s : B[x := r] \qquad x : A \Rightarrow B : \text{type}}{\pi_1(\langle r, s\rangle) = s : B[x := r]}$$

$$\frac{x : A \Rightarrow B : \text{type} \qquad r : (x : A) \times B}{r = \langle \pi_0(r), \pi_1(r)\rangle : (x : A) \times B}$$

Rules for 0, 1, 2

$$\mathbf{0} : \text{stype} \qquad \frac{a : \mathbf{0} \qquad x : \mathbf{0} \Rightarrow A : \text{type}}{\text{case}_0((x)A, a) : A[x := a]}$$

$$\mathbf{1} : \text{stype} \qquad \star : \mathbf{1} \qquad \frac{a : \mathbf{1}}{a = \star : \mathbf{1}}$$

$$\mathbf{2} : \text{stype} \qquad \star_0 : \mathbf{2} \qquad \star_1 : \mathbf{2}$$

$$\frac{x : \mathbf{2} \Rightarrow A : \text{type} \qquad a : \mathbf{2} \qquad b : A[x := \star_0] \qquad c : A[x := \star_1]}{\text{case}_2((x)A, a, b, c) : A[x := a]}$$

$$\frac{x : \mathbf{2} \Rightarrow A : \text{type} \qquad b : A[x := \star_0] \qquad c : A[x := \star_1]}{\text{case}_2((x)A, \star_0, b, c) = b : A[x := \star_0]}$$

$$\frac{x : \mathbf{2} \Rightarrow A : \text{type} \qquad b : A[x := \star_0] \qquad c : A[x := \star_1]}{\text{case}_2((x)A, \star_1, b, c) = c : A[x := \star_1]}$$

$$\frac{a : \mathbf{2} \qquad A : \text{type} \qquad B : \text{type}}{\text{case}_2^{\text{type}}(a, A, B) : \text{type}}$$

$$\text{case}_2^{\text{type}}(\star_0, A, B) = A : \text{type} \qquad \text{case}_2^{\text{type}}(\star_1, A, B) = B : \text{type}$$

Definition A.1 *(a) We write $(x)a$ instead of $(x : A)a$ (abstraction).*
(b) We write $a(b_1, \ldots, b_n)$ instead of $a(b_1) \cdots (b_n)$ for iterated application.
(c) We write repeated abstraction as $(x_1 : A_1, \ldots, x_n : A_n)a$ or $(x_1, \ldots, x_n)a$ instead of $(x_1 : A_1) \cdots (x_n : A_n)a$.
(d) We write $(x_1 : A_1, \ldots, x_n : A_n) \to C$ for $(x_1 : A_1) \to \cdots \to (x_n : A_n) \to C$.
(e) We write $A \to B$ for $(x : A) \to B$ for some fresh x.
(f) We write $(x : A, B) \to C$ for $(x : A, y : B) \to C$ for some fresh y and similarly for longer terms.
(g) We write $A \times B$ for $(x : A) \times B$ for some fresh x;
(h) We write $(x : A) \times (y : B) \times C$ for $(x : A) \times ((y : B) \times C)$ and similarly for longer products.
(i) We write $(x : A) \times B \times C$ for $(x : A) \times (y : B) \times C$ for some fresh y. $A \times (y : B) \times C$, $A \times B \times C$ and similar notions for longer products are defined in the same way.
(j) We write $\pi_0^3(x)$ for $\pi_0(x)$, $\pi_1^3(x)$ for $\pi_0(\pi_1(x))$ and $\pi_2^3(x)$ for $\pi_0(\pi_1(\pi_1(x)))$. These are the 3 projections of a product $(x : A) \times (y : B) \times C$. Similarly we define the projections π_i^n for $n > 3$, $i < n$.
(k) $(-)$ stands for an abstraction (x) for a variable x, which is not used later.
(l) We usually omit in $\text{case}_0((x)A, \cdots)$ and $\text{case}_2((x)A, \cdots)$ the first argument $(x)A$, and write $\text{case}_0(\_, \cdots)$, $\text{case}_2(\_, \cdots)$ instead.

The disjoint union of two types is not a primitive notion in our logical framework, but it can be defined as follows (note that it doesn't refer to an equality type; in the following definition, we usually suppress arguments A, B):

Definition A.2 (a) We define an equality $='_{\mathbf{2}}$ on $\mathbf{2}$ as follows $x ='_{\mathbf{2}} y := \text{case}_2((-)\text{stype}, x, \text{case}_2(\_, y, \mathbf{1}, \mathbf{0}), \text{case}_2(\_, y, \mathbf{0}, \mathbf{1}))$: stype. Note that $(\star_0 ='_{\mathbf{2}} \star_0) = (\star_1 ='_{\mathbf{2}} \star_1) = \mathbf{1}$, $(\star_0 ='_{\mathbf{2}} \star_1) = (\star_1 ='_{\mathbf{2}} \star_0) = \mathbf{0}$.
(b) Define $\text{ref}' : (x : \mathbf{2}) \to (x ='_{\mathbf{2}} x)$, $\text{ref}'\ x = \text{case}_2(\_, x, \star, \star)$. We write ref'_x for $\text{ref}'\ x$.
(c) We define for $A, B : \text{type}$, $h : (x : \mathbf{2}) \to (x ='_{\mathbf{2}} \star_0) \to \text{case}_2^{\text{type}}(x, A, B) \to A$, $h(x) = \text{case}_2(\_, x, (p, y)y, (p, y)\text{case}_0(\_, p))$, and have $h(\star_0, p, x) = x$.

Similarly we define $k : (x : \mathbf{2}) \to (x ='_{\mathbf{2}} \star_1) \to \mathrm{case}_2^{\mathrm{type}}(x, A, B) \to B$ s.t. $k(\star_1, p, x) = x$.

(d) We define $A + B := (x : \mathbf{2}) \times \mathrm{case}_2^{\mathrm{type}}(x, A, B)$, and for $a : A$, $\mathrm{inl}(a) := \langle \star_0, a \rangle : A + B$, and for $b : B$, $\mathrm{inr}(b) := \langle \star_1, b \rangle : A + B$.

(e) We define for $A, B : \mathrm{type}$, $x : (A + B) \Rightarrow C[x] : \mathrm{type}$

$$h'(C) : (x : \mathbf{2}, p : x ='_{\mathbf{2}} \star_0, y : \mathrm{case}_2^{\mathrm{type}}(x, A, B), C[\mathrm{inl}(h(x, p, y))])$$
$$\to C[\langle x, y \rangle] \ ,$$
$$h'(C, x) = \mathrm{case}_2(\_\,, x, (p, y, c)c, (p, y, c)\mathrm{case}_0(\_\,, p)).$$

Similarly we define

$$k'(C) : (x : \mathbf{2}, p : x ='_{\mathbf{2}} \star_1, y : \mathrm{case}_2^{\mathrm{type}}(x, A, B), C[\mathrm{inr}(k(x, p, y))])$$
$$\to C[\langle x, y \rangle] \ ,$$

and we note that $h'(C, \star_0, p, y, c) = c$, $k'(C, \star_1, p, y, c) = c$.

(f) We define for $x : A + B \Rightarrow C[x] : \mathrm{type}$,

$$\mathrm{elim}_+(C) : (step_A : (a : A) \to C[\mathrm{inl}(a)],$$
$$step_B : (b : B) \to C[\mathrm{inr}(b)],$$
$$c : A + B) \to C[c]$$

by $\mathrm{elim}_+(C, step_A, step_B, c) = f(\pi_0(c), c, \mathrm{ref}_{\pi_0(c)})$, where

$$f : (y : \mathbf{2}) \to (x : A + B) \to \pi_0(x) =_{\mathbf{2}} y \to C[x] \ ,$$
$$f(y) := \mathrm{case}_2(\_\,, y,$$
$$(x, p)h'(C, \pi_0(x), p, \pi_1(x), step_A(h(\pi_0(x), p, \pi_1(x))),$$
$$(x, p)k'(C, \pi_0(x), p, \pi_1(x), step_B(k(\pi_0(x), p, \pi_1(x)))) \ .$$

Note that
$\mathrm{elim}_+(C, step_A, step_B, \mathrm{inl}(a)) = f(\star_0, \mathrm{inl}(a), \mathrm{ref}_{\star_0}) =$
$h'(C, \star_0, \mathrm{ref}_{\star_0}, a, step_A(h(\star_0, \mathrm{ref}_{\star_0}, a)) = step_A(a)$
and similarly $\mathrm{elim}_+(C, step_A, step_B, \mathrm{inr}(b)) = step_B(b)$.

(g) Because of the previous definition, we can define $t[x] : C[x]$ depending on $x : A + B$ for some type $C[x]$ by defining $t[\mathrm{inl}(a)] := s_{\mathrm{inl}}[a]$ and $t[\mathrm{inr}(b)] = s_{\mathrm{inr}}[b]$ for some s_{inl}, s_{inr} s.t. $a : A \Rightarrow s_{\mathrm{inl}}[a] : C[\mathrm{inl}(a)]$ and $b : B \Rightarrow s_{\mathrm{inr}}[b] : C[\mathrm{inr}(b)]$. This amounts to defining $t[x] := \mathrm{elim}_+(x, (a)s_{\mathrm{inl}}[a], (b)s_{\mathrm{inr}}[b])$.

In this paper, we have the following general assumption about equality versions of rules and omitting types in equality judgements:

General Assumption A.3 *(a) In this article, except for the previous part of the Appendix, all rules are understood to be supplemented by additional equality rules. For the rules of the logical framework above, the equality rules were already included above. They give examples how rules are to*

be supplemented: E.g. the rule

$$\frac{(x : A) \Rightarrow B : \text{type}}{(x : A) \to B : \text{type}}$$

was supplemented by

$$\frac{A = A' : \text{type} \qquad (x : A) \Rightarrow B = B' : \text{type}}{(x : A) \to B = (x : A') \to B' : \text{type}}$$

and the rule

$$\frac{(x : A) \Rightarrow b : B}{(x : A)b : (x : A) \to B}$$

was supplemented by

$$\frac{(x : A) \Rightarrow b = b' : B}{(x : A)b = (x : A)b' : (x : A) \to B}$$

(b) We will usually omit the type in an equality judgement and assumptions about the types of the variables in it, as they can easily be filled in by the reader.

When proving the equivalence of theories, we will often argue informally, and use the following convention:

Notation A.4 By "we prove $a = b$" by induction on some parameter we mean (assuming $a, b : C$ for some type C) that we introduce $p : a =_C b$ by induction on this parameter. Note that in the presence of extensionality, the existence of such a p is equivalent to $a = b : C$.

We introduce the concept of isomorphisms between types:

Definition A.5 (a) Let $A, B : \text{type}$. Then

$$\begin{aligned} A \cong B := {} & (f : A \to B) \times \\ & (g : B \to A) \times \\ & ((x : A) \to (g(f(x)) =_A x)) \\ & ((x : B) \to (f(g(x)) =_B x) \end{aligned}$$

(b) Assume $f : A \cong B$. Then we define (see Lemma A.1 (j) for the definition of π_i^n):

$$
\begin{aligned}
f^{\rightarrow} &:= \pi_0^4(f) : A \to B \ , \\
f^{\leftarrow} &:= \pi_1^4(f) : B \to A \ , \\
f^{\overrightarrow{\leftarrow}} &:= \pi_2^4(f) : (a : A) \to f^{\leftarrow}(f^{\rightarrow}(a)) =_A a \ , \\
f^{\overleftarrow{\rightarrow}} &:= \pi_3^4(f) : (b : B) \to f^{\rightarrow}(f^{\leftarrow}(b)) =_B b \ .
\end{aligned}
$$

When referring to the later arguments on previous inductive arguments, we often make use of the expression $\mathrm{T} \circ \langle\!\langle \mathrm{i}, f \rangle\!\rangle$.

Definition A.6 *Assume* $A, B : \mathrm{type}$, $b : B \Rightarrow C[b] : \mathrm{type}$, $b : B, c : C[b] \Rightarrow F[b, c] : \mathrm{type}$, $g : A \to B$, $h : (a : A) \to C[g(a)]$, $f : (b : B, c : C[b]) \to F[b, c]$. *Then we define*

$$
f \circ \langle\!\langle g, h \rangle\!\rangle := (a) f(g(a), h(a)) : (a : A) \to F[g(a), h(a)] \ .
$$

We introduce as well a notation for forming the product of two functions (we cannot use the usual notation in category theory $\langle f, g \rangle$, since it is used for pairs):

Definition A.7 *Assume* $A, B : \mathrm{type}$, $b : B \Rightarrow C[b] : \mathrm{type}$, $f : A \to B$, $g : (a : A) \to C[f(a)]$. *Then we define*

$$
\begin{aligned}
(f, g) \quad &: (a : A) \to ((b : B) \times C[b]) \ , \\
(f, g)(a) &:= \langle f(a), g(a) \rangle \ .
\end{aligned}
$$

A.2 Rules of Extensionality

We have explained above how to define the equality set as an indexed inductive definition, but this equality will not be extensional. For some purposes we need to assume that we have an extensional equality type and add the following rules (**ext**) of extensionality. They are only added to our theory if mentioned explicitly.

$$
\frac{A : \mathrm{type} \qquad a : A \qquad b : A}{a =_A b : \mathrm{type}}
$$

$$
\frac{A : \mathrm{stype} \qquad a : A \qquad b : A}{a =_A b : \mathrm{stype}}
$$

$$\frac{A : \text{type} \qquad a : A}{\text{ref} : a =_A a}$$

$$\frac{A : \text{type} \qquad a : A \qquad b : A \qquad r : a =_A b}{a = b : A}$$

$$\frac{A : \text{type} \qquad a : A \qquad b : A \qquad r : a =_A b}{r = \text{ref} : a =_A b}$$

A.3 Finite Sets with Elimination into Type

From the rules of **2** we can derive for any natural number n the set $\mathbf{n}$ of n-elements with elimination into type as follows:

- $\mathbf{n}$ is defined as $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$ for $n <= 2$.
- For $n \geq 3$, $n = m + 1$, $\mathbf{n} := (a : \mathbf{2}) \times \text{case}_2^{\text{type}}(a, \mathbf{m}, \mathbf{1})$.
- For $n \geq 3$, $n = m + 1$, $a : \mathbf{n}$, $A_1, \ldots, A_n : \text{type}$, we define

$$\text{case}_n^{\text{type}}(a, A_1, \ldots, A_n) := \text{case}_2^{\text{type}}(\pi_0(a), \text{case}_m^{\text{type}}(\pi_1(a), A_1, \ldots, A_m), A_n) \ .$$

- Ordinary case distinction case_n can be defined similarly.

B Appendix: Modified Rules for Inductive-Recursive Definitions

We define in the following the theory $\mathbf{IR}_{\mathbf{elim}}^{\mathbf{ext}\ \prime}$, in which the use of $\mathbb{F}_\gamma^{\text{IH}}$, $\mathbb{F}_\gamma^{\text{map}}$ for defining the elimination and equality rules for U_γ is replaced by $\mathbb{F}_\gamma^{\text{IArg}}$ and

$\mathbb{F}_\gamma^{\mathrm{IArg}\to\mathrm{U}}$. See Subsection 5.4 on the motivation for this change.

$$\frac{\gamma : \mathrm{OP}_D \qquad U : \mathrm{set} \qquad T : U \to D \qquad a : \mathbb{F}_\gamma^{\mathrm{U}}(U,T)}{\mathbb{F}_\gamma^{\mathrm{IArg}}(U,T,a) : \mathrm{stype}}$$

$$\frac{\gamma : \mathrm{OP}_D \qquad U : \mathrm{set} \qquad T : U \to D \qquad a : \mathbb{F}_\gamma^{\mathrm{U}}(U,T) \qquad v : \mathbb{F}_\gamma^{\mathrm{IArg}}(U,T,a)}{\mathbb{F}_\gamma^{\mathrm{IArg}\to\mathrm{U}}(U,T,a,v) : U}$$

$$\begin{aligned}
\mathbb{F}_{\iota(e)}^{\mathrm{IArg}}(U,T,*) &= \mathbf{0} \ , \\
\mathbb{F}_{\iota(e)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,*,x) &= \mathrm{case}_0(_\,,x) \ , \\
\mathbb{F}_{\sigma(A,\gamma)}^{\mathrm{IArg}}(U,T,\langle a,b\rangle) &= \mathbb{F}_{\gamma(a)}^{\mathrm{IArg}}(U,T,b) \ , \\
\mathbb{F}_{\sigma(A,\gamma)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,\langle a,b\rangle,c) &= \mathbb{F}_{\gamma(a)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,b,c) \ , \\
\mathbb{F}_{\delta(A,\gamma)}^{\mathrm{IArg}}(U,T,\langle f,b\rangle) &= A + \mathbb{F}_{\gamma(T\circ f)}^{\mathrm{IArg}}(U,T,b) \ , \\
\mathbb{F}_{\delta(A,\gamma)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,\langle f,b\rangle,\mathrm{inl}(a)) &= f(a) \ , \\
\mathbb{F}_{\delta(A,\gamma)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,\langle f,b\rangle,\mathrm{inr}(a)) &= \mathbb{F}_{\gamma(T\circ f)}^{\mathrm{IArg}\to\mathrm{U}}(U,T,b,a) \ ,
\end{aligned}$$

$$\frac{\begin{array}{c}\gamma : \mathrm{OP}_D \\ x : \mathrm{U}_\gamma \Rightarrow E[x] : \mathrm{type} \\ g : (a : \mathbb{F}_\gamma^{\mathrm{U}}(\mathrm{U}_\gamma,\mathrm{T}_\gamma), ih : (v : \mathbb{F}_\gamma^{\mathrm{IArg}}(\mathrm{U}_\gamma,\mathrm{T}_\gamma,a)) \to E[\mathbb{F}_\gamma^{\mathrm{IArg}\to\mathrm{U}}(\mathrm{U}_\gamma,\mathrm{T}_\gamma,a,v)]) \to E[\mathrm{intro}_\gamma(a)]\end{array}}{\begin{array}{c}\mathrm{R}_{\gamma,E}(g) : (u : \mathrm{U}_\gamma) \to E[u] \\ \mathrm{R}_{\gamma,E}(g,\mathrm{intro}_\gamma(a)) = g(a,(v)\mathrm{R}_{\gamma,E}(g,\mathbb{F}_\gamma^{\mathrm{IArg}\to\mathrm{U}}(U,T,a,v)))\end{array}}$$

Definition B.1 *(a) The theory* $\mathbf{IR}_{\mathbf{elim}}^{\mathbf{ext}\,\prime}$ *is obtained from the theory* $\mathbf{IR}_{\mathbf{elim}}^{\mathbf{ext}}$*, as defined in [16], by replacing the logical framework by the one used in this article (that is, with* $\mathrm{case}_2^{\mathrm{type}}$*), omitting the constants* $\mathbb{F}^{\mathrm{IH}}$*,* $\mathbb{F}^{\mathrm{map}}$ *and the rules for introducing them, adding the rules for* $\mathbb{F}^{\mathrm{IArg}}$ *and* $\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}$ *as introduced above and replacing the elimination and equality rules for* U_γ *by the rules above.*

(b) $\mathbf{OP}_{\mathbf{intro}}^{\mathbf{s}}$ *and* $\mathbf{OP}_{\mathbf{elim}}^{\mathbf{s}}$ *are the introduction and elimination rules for* OP_D *for inductive-recursive definitions as introduced in [16]. Note that* $\mathbf{OP}_{\mathbf{intro}}^{\mathbf{s}}$ *is contained in the rules for* $\mathbf{IR}_{\mathbf{elim}}^{\mathbf{ext}\,\prime}$*, but not* $\mathbf{OP}_{\mathbf{elim}}^{\mathbf{s}}$*.*

Lemma B.2 *Let* $(\mathbb{F}^{\mathrm{IH}})$ *be the rules for* $\mathbb{F}^{\mathrm{IH}}$*. Define* $\mathbb{F}_\gamma^{\mathrm{map}}$ *by using* $\mathbf{OP}_{\mathbf{elim}}^{\mathbf{s}}$*.*

Define in $\mathbf{IR}^{\mathbf{ext}\ \prime}_{\mathbf{elim}} + (\mathbb{F}^{\mathrm{IH}}) + \mathbf{OP}^{\mathbf{s}}_{\mathbf{elim}}$

$$\mathbb{F}^{\mathrm{IH}\prime}_{\gamma}(U,T,E,a) := (v : \mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a)) \to E[\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)] \ ,$$
$$\mathbb{F}^{\mathrm{map}\prime}_{\gamma}(U,T,E,g,a) := (v : \mathbb{F}^{\mathrm{IArg}}_{\gamma}(U,T,a))g(\mathbb{F}^{\mathrm{IArg}\to\mathrm{U}}_{\gamma}(U,T,a,v)) \ .$$

Then in $\mathbf{IR}^{\mathbf{ext}\ \prime}_{\mathbf{elim}} + (\mathbb{F}^{\mathrm{IH}}) + \mathbf{OP}^{\mathbf{s}}_{\mathbf{elim}}$ *one can introduce the following isomorphism and show the following equation:*

$$\mathbb{F}^{\mathrm{IH},\cong}_{\gamma}(U,T,E,a) : \mathbb{F}^{\mathrm{IH}\prime}_{\gamma}(U,T,E,a) \cong \mathbb{F}^{\mathrm{IH}}_{\gamma}(U,T,E,a)$$
$$\mathbb{F}^{\mathrm{IH},\cong}_{\gamma}(U,T,E,a)^{\to}(\mathbb{F}^{\mathrm{map}\prime}_{\gamma}(U,T,E,g,a)) = \mathbb{F}^{\mathrm{map}}_{\gamma}(U,T,E,g,a)$$

In this sense it follows that the new rules are equivalent to the old rules. Note that however $\mathbb{F}^{\mathrm{IH}}_{\gamma}(U,T,E,a)$ cannot be defined using the new rules.

References

[1] P. Aczel. *Frege Structures and the Notions of Proposition, Truth, and Set*, pages 31–59. North-Holland, 1980.

[2] T. Altenkirch, V. Gaspes, B. Nordström, and B. von Sydow. A user's guide to ALF. http://www.cs.chalmers.se/ComputingScience/Research/Logic/alf/guide.html, 1996.

[3] T. Altenkirch and C. McBride. Generic programming within dependently typed programming. In *Generic Programming*, 2003. Proceedings of the IFIP TC2 Working Conference on Generic Programming, Schloss Dagstuhl, July 2002.

[4] T. Altenkirch and B. Reus. Monadic presentations of lambda terms using generalized inductive types. In *Computer Science Logic*, 1999.

[5] M. Benke, P. Dybjer, and P. Jansson. Universes for generic programs and proofs in dependent type theory. *Nordic Journal of Computing*, 10:265–289, 2003.

[6] A. Bove and V. Capretta. Nested general recursion and partiality in type theory. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher Order Logics: 43th International Conference, TPHOLs 2001*, volume 2152 of *Springer-Verlag, LNCS*, pages 121–135, September 2001.

[7] C. Coquand. Agda. 2000. http://www.cs.chalmers.se/∼catarina/agda/.

[8] C. Coquand. A realizability interpretation of Martin-Löf's type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998.

[9] T. Coquand. Pattern matching with dependent types. In B. Nordström and Kent Petersson and Gordon Plotkin, editor, *Proceedings of The 1992 Workshop on Types for Proofs and Programs*, June 1992.

[10] T. Coquand and C. Paulin. Inductively defined types, preliminary version. In *COLOG '88, International Conference on Computer Logic*, volume 417 of *LNCS*. Springer-Verlag, 1990.

[11] P. Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.

[12] P. Dybjer. Inductive families. *Formal Aspects of Computing*, 6:440–465, 1994.

[13] P. Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *Journal of Symbolic Logic*, 65(2):525–549, June 2000.

[14] P. Dybjer and A. Setzer. A finite axiomatization of inductive-recursive definitions. In J.-Y. Girard, editor, *Typed Lambda Calculi and Applications*, volume 1581 of *Lecture Notes in Computer Science*, pages 129–146. Springer, April 1999.

[15] P. Dybjer and A. Setzer. Indexed induction-recursion. In *Proof Theory in Computer Science International Seminar, PTCS 2001 Dagstuhl Castle, Germany*, number 2183 in LNCS, pages 93–113, October 2001.

[16] P. Dybjer and A. Setzer. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic*, 124:1–47, 2003.

[17] R. Hinze and J. Jeuring. Generic Haskell: Practice and theory. In R. Backhouse and J. Gibbons, editors, *Generic Programming*, number 2793 in LNCS, pages 1 – 56. Springer-Verlag, 2003.

[18] P. Jansson and J. Jeuring. PolyP — a polytypic programming language extension. In *Proc. POPL'97*, pages 470–482. ACM Press, 1997.

[19] L. Magnusson and B. Nordström. The ALF proof editor and its proof engine. In *Types for proofs and programs*, volume 806 of *Lecture Notes in Computer Science*, pages 213–317. Springer, 1994.

[20] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science, VI, 1979*, pages 153–175. North-Holland, 1982.

[21] P. Martin-Löf. On the meaning of the logical constants and the justification of the logical laws, 1983. Notes from a series of lectures given in Siena.

[22] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[23] P. Martin-Löf. An intuitionistic theory of types. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998. Reprinted version of an unpublished report from 1972.

[24] R. Matthes. Review of Dybjer, Peter and Setzer, Anton: Induction-recursion and initial algebras. *Mathematical Reviews*, MR2013392, 2005. To appear.

[25] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: an Introduction*. Oxford University Press, 1990.

[26] E. Palmgren. *On Fixed Point Operators, Inductive Definitions and Universes in Martin-Löf's Type Theory*. PhD thesis, Uppsala University, 1991.

[27] E. Palmgren. On universes in type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998.

[28] C. Paulin-Mohring. Inductive definitions in the system Coq - rules and properties. In *Proceedings Typed λ-Calculus and Applications*, pages 328–245. Springer-Verlag, LNCS, March 1993.

[29] H. Pfeifer and H. Rueß. Polytypic proof construction. In Y. Bertot, editor, *Proc. TPHOLs'99*, volume 1690 of *LNCS*, pages 55–72. Springer-Verlag, 1999.

[30] Qiao Haiyan. Formalising formulas-as-types-as-objects. In T. Coquand, P. Dybjer, B. Nordström, and J. Smith, editors, *Types for Proofs and Programs*, volume 1956 of *Lecture Notes in Computer Science*, pages 174–194. Springer, 2000.

[31] A. Setzer. Extending Martin-Löf type theory by one Mahlo-universe. *Arch. Math. Log.*, 39:155 – 181, 2000.

[32] A. Setzer. Proof theory of Martin-Löf type theory – an overview. *Mathematiques et Sciences Humaines*, 42 année, n^{o}165:59 – 99, 2004.