The Simply Typed System $\mathcal{N}$ and Extendable Recursion

S. Berardi, U. de'Liguoro

December 1, 2017

Abstract. We define the system "Nicla", or $\mathcal{N}$ for short, from a set `D` of data types (all of them types of trees) using product and arrow types. Maps are defined by primitive recursion on trees. Recursion includes a special clause we call the *polymorphic clause* to be used when the domain of the recursive map is extended. Extending a recursion to a larger domain produces a circularity problem, which we solve in our semantics. Our goal is defining a system as expressive as Girard's system $\mathcal{F}$, but using no explicit quantification on types. For sake of simplicity we use no variables in $\mathcal{N}$, but this is an arbitrary choice. Our results include termination of computations in $\mathcal{N}$ and that fact that all trees denoted by terms of $\mathcal{N}$ are well-founded.

Contents

Chapter 1

System $\mathcal{N}$ and Extendable Recursion

1.1. Extendable Recursion By extending a recursive definition we mean the following problem. Assume $\mathtt{j}_{x\in X}F(x)$ denotes a tree whose immediate subtrees are all $F(x)$ for $x \in X$. Assume D denotes some set of trees which are *well-founded* trees, and whose subtrees all have the form $\mathtt{j}_{x\in X}F(x)$ for some index set $X \in \{D_0, \ldots, D_{n-1}\} \cup \mathcal{X}$. Assume $D@E$ denotes the set of well-founded trees whose subtrees all have the form $\mathtt{j}_{x\in X}F(x)$ for some index set $X \in \{D_0, \ldots, D_{n-1}, E\} \cup \mathcal{X}$: in $D@E$ we have one more index set, namely E. We want some uniform way of extending simply typed primitive recursive map from $D \to D$ to $D@E \to D@E$, while preserving the fact that all trees of the model are well-founded.

More in detail, we are looking for an extensional model of simply typed lambda calculus (maps are set-theoretical maps) having:

1. a map $\mathtt{r}$ taking as argument the recursive clauses $R_i : (D_i \to D) \to D$ for all $i < n$ and $R_n : D \to D$, and a map $\mathtt{u} : (D \to D) \to (D \to D)$
2. for all E a map $\mathtt{lift}_E : (D \to D) \to (D@E \to D@E)$. We write F^* for $\mathtt{lift}(F)$.
3. for all at most countable $X \subseteq E$ some map $(.)\lceil X : D@E \to D$.

$\mathtt{r}$ defines maps : $D \to D$ by primitive recursion, with one defalt case for all index sets $X \neq D_0, \ldots, D_{n-1}$. $\mathtt{u}$ applies a map to all children of a tree. $\mathtt{lift}$ extends the recursive definition to a new index set E by using the default clause of primitive recursion. $(.)\lceil X$ restrict the index set E to X, and allows us to state that the behavior of an extended recursion on the index set E and the index set X are the same, namely an application of the inductive clause R_n. We express these requests by the following equations.

1. If $H = \mathtt{r}\vec{R} : D \to D$ and $X = D_0, \ldots, D_{n-1}$ then

$$H(\mathtt{j}_{x\in X}F(x)) = R_i(\lambda x \in X.H(F(x)))$$

and if $X \neq D_1, \ldots, D_{n-1}$ then

$$H(\mathtt{j}_{x\in X}F(x)) = R_n(\mathtt{j}_{x\in X}.H(F(x)))$$

2. $\mathtt{u}(F, \mathtt{j}_{y\in Y}G(y)) = \mathtt{j}_{y\in Y}F(G(y))$

3. If $t \in D@E$ and $Y \neq E$ then

$$(\mathtt{j}_{e\in E}F(e))\lceil X = \mathtt{j}_{e\in X}F(e)\lceil X \qquad (\mathtt{j}_{y\in Y}G(y))\lceil X = \mathtt{j}_{y\in Y}G(y)\lceil X$$

4. `lift` and $(.)\lceil X$ commute: if $F^* = \mathtt{lift}(F)$ then

$$F^*(t)\lceil X = F(t\lceil X)$$

Extendable Recursion A *model of Extendable Recursion* is any extensional model of simply typed lambda calculus with $\times, \to$ satisfying all equations above, and in which all elements of atomic type are well-founded trees.

In any model of extendable recursion, we may characterize the behavior of a recursively defined map $H = \mathtt{r}\vec{R} : D \to D$ when extended to $H^* : D@E \to D@E$. The map H^* uses the default clause on the constructor $\mathtt{j}_{x\in E}(.)$, as expected. That is, we have $H^*(\mathtt{j}_{x\in E}F(x)) = R_n^*(\mathtt{j}_{x\in E}.H(F(x)))$. (*Proof Sketch*, This follows from $H^*(\mathtt{j}_{x\in E}F(x))\lceil X = R_n^*(\mathtt{j}_{x\in E}.H(F(x)))\lceil X$ for any countable $X \subseteq E$. This in turn follows from the properties of $(.\lceil X)$ and the default clause for recursive definitions).

We say that two notation having a simple type are extensionally equivalent if they are tree notations and they denote the same tree, or are function notations and are pointwise equivalent, or are pairs and are componentwise equivalent. We introduce a simply typed lambda calculus $\mathcal{N}$ whose extensional quotient defines a model of extendable recursion. In $\mathcal{N}$ there is a primitive constant for $\mathtt{r}$, and there are terms $\mathtt{lift}, (.)\lceil X$ which satisfy the required equations. We prove that $\mathcal{N}$ has an extra feature, namely Normalization: there are reductions taking any expression t denoting a tree and returning its index set E and its children.

In the rest of this chapter we introduce types and terms of $\mathcal{N}$ and an informal interpretation for them.

1.2. Types of $\mathcal{N}$ We informally describe the data types in $\mathtt{D}$ as follows. Any $D \in \mathtt{D}$ is a set of trees having finitely many constructors $c_{0,D}, \ldots, c_{n-1,D}$, plus a family of others of the form $\mathtt{j}_X$ for some $X \in \mathcal{X}$. For the moment we describe D in the case $\mathcal{X} = \emptyset$. Each constructor $c \equiv c_{i,D}$ of D has argument a family f of elements of D indexed on some previously defined data type D_i. Thus, the argument f of c has type $D_i \to D$. c returns an element $cf : D$ of D. cf denotes the tree with root labeled c and having as immediate sub-tree of index $e : D_i$ the tree $f(e) : D$. We call D_i the *index set* of c and we write $D_i = \mathtt{I}(c)$. We call the cardinality of D_i the *arity* of c. We call the elements of D_i indexes.

We inductively define the set $\mathtt{D}$ of data types, then the set $\mathtt{Tp}$ of product and arrow types, and eventually the terms of $\mathcal{N}$.

Data types and Types

1. *Data Types.* If $\vec{D} = D_0, \ldots, D_{n-1} \in \mathtt{D}$ is a (possibly empty) finite list of data types, then $D = (\vec{D}) \in \mathtt{D}$ is a data type. $n = \mathtt{l}(D)$ is the number of constructors of D, and for each $i < n$, D_i is the index set of the constructor number i of D.

2. *Types.* $\mathtt{D} \subseteq \mathtt{Tp}$. If $A, B \in \mathtt{Tp}$ then $A \times B, A \to B \in \mathtt{D}$.

A data type D is strictly positive in a type A if D occurs in the left-hand side of no arrow type in A. We define an operation $@ : \mathtt{Tp}, \mathtt{D} \to \mathtt{Tp}$, extending a type A to a type $A@E$. $A@E$ adds the constructor of index set $E \in \mathtt{D}$ to all strictly positive data types in A.

Extending a type

1. If $A \equiv (D_0, \ldots, D_{n-1})$ then $A@E \equiv (D_0, \ldots, D_{n-1}, E)$

2. If $A \equiv B \times C$ then $A@E \equiv B@E \times C@E$

3. If $A \equiv (B \to C)$ then $A@E \equiv (B \to C@E)$.

1.3. Contexts of $\mathcal{N}$ The constants in $\mathcal{N}$ are algebraic combinators, pairing and projections, constructors of a data type, typed primitive recursion on a data type, plus two families of constants dealing with the extension of a recursive definition. These latter are are:

1. *future constructors* $\mathtt{f}$, denoting a constructor which is not yet part of a type, but will be inserted by some extension $(.)@E$

2. *uniform application*, a constant $\mathtt{u}$ which may modify the result of a future constructor.

Besides constants, $\mathcal{N}$ has a unary operator `F`, extending an element of A to an element of $A@E$, and binary application.

Alternatively, we may imagine that a constructor has a state, and it is either a current constructor of $D = (D_0, \ldots, D_{n-1})$ or a future constructor with index set E. The extension operator `F` changes the state of a constructor c from a future constructor to a current constructor. When we define a recursive map f on D we have a special recursive clause dealing with a term of D starting with a future constructor `f`. When we want to apply f to a term beginning with a constructor c not in D, we have first to replace c with a future constructor `f`, coming back to a time in which c is not defined yet. Then we apply f, and eventually we use the operator `F` to replace `f` back with c, coming back to the instant of time from which we started. The result is applying f to *any* new constructor: this is a wide extension to the possibilities offered by a recursive definition.

Terms of $\mathcal{N}$ are defined w.r.t. a list of pending extensions or *context*, representing steps in a computation or a temporal line. We have no explicit notion of variable in $\mathcal{N}$: a context in is any finite list of data types, that is, any $\Gamma = E_0, \ldots, E_{m-1} \in \mathtt{D}$. For any $A \in \mathtt{Tp}$, the context Γ has constructors $\mathtt{f}_0 : (E_0 \to A) \to A, \mathtt{f}_{m-1} \ldots, (E_{m-1} \to A) \to A$. For any $j < m$, we call each constant $\mathtt{f}_j$ a *future constructor*. The index set of $\mathtt{f}_j$ is $\mathtt{I}(\mathtt{f}_j) = E_j$, a type which we assume to be "not known yet".

1.4. Terms of $\mathcal{N}$ We informally describe how terms of $\mathcal{N}$ are defined w.r.t. a context Γ.

We have all types algebraic constructors in $\mathcal{N}$.

A term t of type $D = (D_0, \ldots, D_{n-1})$ in the context Γ may include any constructor of D: $c_0, \ldots, c_{n-1}$, of index set $D_0, \ldots, D_{n-1}$, and any constructor $\mathtt{f}_0 : (E_0 \to D) \to D, \mathtt{f}_{m-1} \ldots, (E_{m-1} \to D) \to D$ of Γ. The difference between constructors and future constructors is that, in a recursive definition, we do not allow to select a child fe of $\mathtt{f}f$, while we do allow to select a child fe of cf if c is a constructor.

As we said, there is constant for *uniform* application, applying the same map on all children of a future constructor.

We have constants for primitive recursions on trees, requiring one clause for each constructor, and one special clause we call the polymorphic clause, to be used for all future constructors. The polymorphic clause cannot include the index set E_j of a future constructor, because we assumed E_j to be "unknown". The polymorphic clause, however, may use uniform application to act in the same way on all children of a future constructor.

There is application and a unary operator `F`. `F` transforms a future con-

structor $\mathtt{f}f$ into a constructor cf, removes its index set E from the list of pending extensions, and modifies the numbering of future constructors and of the remaining $\mathtt{F}$ operators accordingly. After this operation a constant for recursion may be used to select a child fe of cf.

We recall the definition of algebraic combinators. An algebraic expression α is either a variable x^A for some $A \in \mathtt{Tp}$ or an application $\beta(\gamma)$ of algebraic expressions $\beta : B \to C$, $\gamma : C$. We usually drop the type superscript of a variable; variables in α do not appear in $\mathcal{N}$. A combinator C denotes the map defined by the equation $C(\vec{x}) = \alpha$, for some algebraic expression α whose variables are all in $\vec{x}$. Here are some examples. $S_{A,B,C} : (A, B \to C), (A \to B), A \to C$ is substitution, with defining equation $S_{A,B,C}(x, y, z) = x(z)(y(z))$. $K_{A,B} : A, B \to A$ is the eraser, with defining equation $K_{A,B}(x, y) = x$. $\circ_{A,B,C} : (B \to C), (A \to B) \to (A \to C)$ is composition, with defining equation $\circ_{A,B,C}(x)(y)(z) = x(y(z))$. Variables and defining equations are used to define the combinators and are no explicit part of $\mathcal{N}$.

We now formally define the terms of $\mathcal{N}$.

Terms of $\mathcal{N}$ For any types $D, E \in \mathtt{D}$, $A, B \in \mathtt{Tp}$, the system $\mathcal{N}$ has:

1. in any context Γ: constants for all combinators
2. in any context Γ: constants for pairing $< \_, \_ >$: $A, B \to A \times B$ and for projections $\pi_1 : A \times B \to A$, $\pi_2 : A \times B \to B$
3. in any context Γ: constants for constructors $c \equiv \mathtt{cons}_{i,D} : (D_i \to D) \to D$ for each data type $D = (D_0, \ldots, D_{n-1})$, each $i < n$, with $\mathtt{I}(c) = D_i$.
4. in any context Γ: constants for future constructors $c \equiv \mathtt{f}_{j,E,A} : (E \to A) \to A$, in the context Γ, E, Δ, with $\Gamma = E_0, \ldots, E_{j-1}$, and $\mathtt{I}(c) = E$.
5. in any context Γ: constants $\mathtt{u}_D : D, (D \to D) \to D$ for uniform application.
6. in any context Γ: constants $\mathtt{r} : \vec{R}, D \to A$ for recursion, with the clauses $r_i : R_i = (D_i \to A) \to A$ for $i = 0, \ldots, n-1$, and the polymorphic clause $r_n : R_n = (A \to A)$.
7. A unary operation for executing the j-th pending extension: if $t : A$ in the context Γ, E, Δ, then $\mathtt{F}_{j,E}.t : A@E$ in the context Γ, Δ, with $\Gamma = E_0, \ldots, E_{j-1}$.
8. A binary operation for application: if $t : A \to B$ and $u : A$ in the context Γ, then $t(u) : B$ in the context Γ.

We write $\Gamma \vdash t : A$ for "$t : A$ in the context Γ".

Thanks to the fact that we have pairing, we do not need in $\mathcal{N}$ the more involved typing $D, (D_i \to A) \to A$ for describing a recursive clause. This typing may be simulated by applying recursion to the type $D \times A$.

In the rest of the paper we need an operation of context lifting $a^{i,E}$, moving a term a of $\mathcal{N}$ + constants from the context Γ, Δ to the context Γ, E, Δ. $a^{i,E}$ increases by 1 all subscripts j of expressions $\mathtt{f}_{j,E}, \mathtt{F}_{j,E}v$ with $j \geq i$. $(.)^{i,E}$ is the opposite of the operator $\mathtt{F}_{i,E}$, which decreases indexes by 1 and removes $\mathtt{f}_{j,E}$. Taking all this into account we are lead to the following clauses.

Context Lifting Assume $t : A$ in $\mathcal{N}$ + constants in the context Γ, Δ, with $\Gamma = E_0, \ldots, E_{k-1}$. We define a term $t^{k,E} : A$ in the context Γ, E, Δ by induction on t. We write t^k skipping the superscript E for brevity.

- $c^k \equiv c$ for any constant $c \not\equiv \mathtt{f}$ of $\mathcal{N}$ + constants.

($\mathtt{f}_<$) $(\mathtt{f}_{j,F})^k \equiv \mathtt{f}_{j,F}$ for all $j \leq k-1$

($\mathtt{f}^+_\geq$) $(\mathtt{f}_{j,F})^k \equiv \mathtt{f}_{j+1,F}$ for all $k \leq j$

($\mathtt{F}_<$) $\mathtt{F}_{j,F}(u)^k \equiv \mathtt{F}_{j,F}(u^{k+1})$ for all $j \leq k-1$

($\mathtt{F}^+_\geq$) $\mathtt{F}_{j,F}(u)^k \equiv \mathtt{F}_{j+1,F}(u^k)$ for all $k \leq j$.

- $t(u)^k \equiv t^k(u^k)$

If $\Gamma = E_0, \ldots, E_{p-1}$ and $\mathtt{nil} \vdash a : A$ we write a^Γ for $(\ldots(a^{0,E_0})\ldots)^{p-1,E_{p-1}}$.

If Γ has i elements and $\Gamma, \Delta \vdash a : A$ then $\Gamma, E, \Delta \vdash a^{i,E} : A$ (*Proof*: by induction on the proof of $\Gamma, \Delta \vdash a : A$).

In order to provide a semantics for $\mathcal{N}$ we add an *infinite* set $\mathcal{I}$ of constants we call "names", and we define an extension $\mathcal{N} + \mathcal{I}$ of $\mathcal{N}$. Any infinite set is fine for $\mathcal{I}$, and reader may assume that $\mathcal{I} = \mathtt{N}$, the set of natural numbers. Any name $i \in \mathcal{I}$ represents some "not yet known" element of some "not yet known" data type E. $\mathcal{N}+\mathcal{I}$ has more constants $\mathtt{j}_{X,E,A}$ we call "approximations of future constructors", with the rule: if $I \subseteq \mathcal{I}$ and $t_i : E$ has context $\mathtt{nil}$ for all $i \in I$ and $X = \{t_i | i \in I\}$, then $\mathtt{j}_{X,E,A} : (E \to A) \to A$ is a constant in the context Γ. The index set of $\mathtt{j}_{X,E,A}$ is defined as $\mathtt{I}(\mathtt{j}_{X,E,A}) = X$. We often drop the subscripts E, A. Since $\mathtt{j}_X$ is a new constant of $\mathcal{N}$, by definition $\mathtt{j}_X$ does not change if we change context: we have ${\mathtt{j}_X}^{i,E} \equiv \mathtt{j}_X$.

We call a term $\mathtt{j}_X f$ a *joint*: it represents the tree whose root has index set X (a subset of E) and children all $f(e)$ for $e \in X$. $\mathtt{j}_X f$ approximates the tree $\mathtt{f}_{j,E} f$ whose index set is E.

The operator $\mathtt{F}_{i,E}$ takes a term $\Gamma, E, \Delta \vdash t : A$, with $\Gamma = E_0, \ldots, E_{i-1}$ and returns a term $\Gamma, \Delta \vdash \mathtt{F}.t : A@E$, whose value is obtained by replacing $\mathtt{f}_{i,E,D}$ with $\mathtt{cons}_{n,D@E}$ if $A = (D_0, \ldots, D_{n-1}) \in \mathtt{D}$. $\mathtt{F}$ is extended point-wise and component-wise to all types of $\mathtt{Tp}$. We will precise the definition of $\mathtt{F}$ through a set of reductions.

An extension takes place when a future constructor $\mathtt{f}_{i,E}$ is replaced by a constructor using an operator $\mathtt{F}_{i,E}$. An extension increases the type of a term from A to $A@E$.

1.5. Reductions for $\mathcal{N}$ We define an interpretation for terms of $\mathcal{N} + \mathcal{I}$ by first introducing a notion of reduction. Later we introduce a notion of model and we use models to prove that all terms of $\mathcal{N} + \mathcal{I}$ normalize and all terms $t : D \in \mathtt{D}$ denote a well-founded tree.

A term t is an *incomplete redex* if t is a constant s, possibly followed by less arguments than those required to define the reduction for s.

Assume that C is a combinator of arity k and $\mathtt{r}$ is the recursion on some $D \in \mathtt{D}$ with n constructors. Then t is an *incomplete redex* if we have: $t \equiv C\vec{a}, \pi_i, \mathtt{u}, \mathtt{u}b, \mathtt{r}\vec{c}$, with $\mathtt{l}(\vec{a}) < k$ (C requires k arguments) and $\mathtt{l}(\vec{c}) < n+2$ (a recursion on D requires $n+1$ arguments).

A term t is a *construction* if $t \equiv \mathtt{cons} f, < a, b >$. A term t is a *future construction* if $t \equiv \mathtt{f} f$ and a *joint* when $t \equiv \mathtt{j}_X f$. A term t is an *incomplete construction* if $t \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X, < \_, \_ >, < a, \_ >$.

Normal form Let X be any set of the form $\{t_i | i \in I\}$, for some $I \subseteq \mathcal{I}$. Any $t : A$ of $\mathcal{N} + \mathcal{I}$ is a normal form if

1. either $A \in \mathtt{D}$ and $t \equiv \mathtt{cons} f, \mathtt{f} f$ (t is a construction or future construction), or $t \equiv \mathtt{j}_X f$, or
2. $A \equiv (B \times C)$ and $t \equiv < b, c >, \mathtt{f} f$ (t is a construction or future construction) or $t \equiv \mathtt{F}.u, \mathtt{j}_X f$ or
3. $A \equiv (B \to C)$ and $t \equiv \mathtt{f} f$ (t is a future construction), or t is an incomplete redex, or an incomplete construction, or $t \equiv \mathtt{F}.u, \mathtt{j}_X f$.

Summing up: a normal term $t : D \in \mathtt{D}$ is either a construction $\mathtt{cons} f$, or a future construction $\mathtt{f} f$, or a joint approximating a future construction in the present time. Thus, if $t \equiv \mathtt{F}u : D \in \mathtt{D}$ then t is never normal: we want that a forward operation on an atomic type is executed.

A normal term $t : A \times B$ is either a construction $< a, b >$, or a future construction $\mathtt{f}f$, or or a joint approximating a future construction in the present time, or is some $\mathtt{F}.u$. A forward operation $\mathtt{F}.u$ is suspended in a product type, and executed only in a data type.

A normal term $t : A \to B$ is never a construction: it is either a future construction or an incomplete redex or an incomplete construction or a joint approximating some future construction, or is some suspended computation $\mathtt{F}.u$.

For any non-normal term t we define *exactly one one-step reduction* $t{\rightsquigarrow}u$. In the case of terms $\pi_i(a)$, $\mathtt{u}(a, f)$, $\mathtt{r}(\vec{r}, a$, the reduction applies only if the argument a of $\pi_i, \mathtt{u}, \mathtt{r}$ is a normal form, otherwise the reduction takes place in a. A term of the form $\mathtt{F}.d$ for some $d : D \in \mathtt{D}$ reduces if d is in normal form, otherwise the reduction takes place in d. When a context does not change (when a forward operation $\mathtt{F}$ is not involved), reductions for future constructions and for joints have to be the same, in order to use joints to represent future constructions.

Assume P is a combinator postponing an application, defined by $Pbg = g(b)$, hence $(Pb{\circ}f)(a) = P(b)(f(a)) = f(a)(b)$

Reductions for $\mathcal{N}$ (except those for $\mathtt{F}$) Let $X = \{t_i | i \in I\}$ denote any set of terms of type some $E \in \mathtt{D}$, context $\mathtt{nil}$ and index set some $I \subseteq \mathcal{I}$.

1. Reductions for a combinator $C(\vec{x}) = \alpha$ and pairing. Assume $\vec{t}$ is a vector of terms having the same length and the same types as $\vec{x}$.
   (a) $C(\vec{t}){\rightsquigarrow}\alpha[\vec{t}/\vec{x}]$.
   (b) $\pi_i(< a_1, a_2 >){\rightsquigarrow}a_i$ for $i = 1, 2$
   (c) If $a{\rightsquigarrow}b$ then $\pi_i(a){\rightsquigarrow}\pi_i(b)$.
   (d) If $f{\rightsquigarrow}g$ then $fa{\rightsquigarrow}ga$
2. Pointwise/componentwise reductions for approximated and future constructors. Let $c \equiv \mathtt{j}_X, \mathtt{f}_{i,E}$.
   (a) $c(f)(e){\rightsquigarrow}c(Pe{\circ}f)$
   (b) $\pi_i(c(f)){\rightsquigarrow}c(\pi_i{\circ}f)$ for $i = 1, 2$
3. Uniform application for *any* constructor. Let $c \equiv \mathtt{j}_X, \mathtt{f}_{i,E,D}, \mathtt{cons}_{i,D}$. Assume $g : D \to B$ and $c(f) : D$.
   (a) $\mathtt{u}(c(f))(g){\rightsquigarrow}c(g{\circ}f) : B$
   (b) If $d{\rightsquigarrow}e : D$ then $\mathtt{u}(d)(g){\rightsquigarrow}\mathtt{u}(e)(g)$

4. Reductions for $\mathtt{r}$. Assume $\mathtt{l}(D) = n$, $\vec{r} = r_0, \ldots, r_n$.

   (a) If $d \equiv c(f)$ and $c \equiv \mathtt{cons}$ is a constructor of index set $E = D_i$ for some $i < n$ then $\mathtt{r}\vec{r}d \rightsquigarrow r_i(\mathtt{r}(\vec{r}) \circ f)$ (c is *consumed*).

   (b) If $d \equiv c(f)$ and $c \equiv \mathtt{f}, \mathtt{j}_X$ is a future or approximated constructor of index set E then $\mathtt{r}\vec{r}d \rightsquigarrow r_n(c(\mathtt{r}(\vec{r}) \circ f))$ (c is *not consumed*).

   (c) If $d \rightsquigarrow e$ then $\mathtt{r}\vec{r}d \rightsquigarrow \mathtt{r}\vec{r}e$

Assume $\Gamma, E, \Delta \vdash a : A$, with E in the i-th position. Then $\Gamma, \Delta \vdash \mathtt{F}_{i,E}.a : A@E$. $\mathtt{F}_{i,E}$ executes the pending extension E, producing a term of type $A@E$ in the context Γ, Δ. We define reductions for $\mathtt{F}_{i,E}.a$.

One-step Reductions for $\mathtt{F}$ Let us abbreviate $\mathtt{F}_{i,E}$ with $\mathtt{F}$. Assume $D = (D_0, \ldots, D_{n-1})$.

1. $\mathtt{F}.(\mathtt{f}_{i,E,D} f) \rightsquigarrow c_{n,D@E}(\mathtt{F}.f)$
2. $\mathtt{F}.(\mathtt{f}_{j,E,D} f) \rightsquigarrow \mathtt{f}_{j-1,E,D@E}(\mathtt{F}.f)$ for $j \geq i+1$
3. $\mathtt{F}.(cf) \rightsquigarrow c(\mathtt{F}.f)$ for any other constructor, approximation or future constructor.
4. If $d : D \in \mathtt{D}$ and $d \rightsquigarrow e : D$ then $\mathtt{F}d \rightsquigarrow \mathtt{F}e$.
5. $(\mathtt{F}.f)(a) \rightsquigarrow \mathtt{F}.f(a^{i,E})$
6. $\pi_i(\mathtt{F}.a) \rightsquigarrow \mathtt{F}.\pi_i(a)$ for $i = 1, 2$.

We may prove Subject Reduction: the one-step reduction $t \rightsquigarrow u$ is type- and context-preserving. The proof is by induction on the proof of $\Gamma \vdash t : A$.

We write $t \rightsquigarrow^n u$ if there is a reduction path from t to u of length n, and $t \rightsquigarrow^* u$ or "t reduces to u" if $t \rightsquigarrow^n u$ for some $n \in \mathtt{N}$. A term is in normal form if there is no reduction from it, a term normalizes if $t \rightsquigarrow^* u$ for some normal u.

The terms of $\mathcal{N} + \mathcal{I}$ may be represented by (possibly infinite) trees we call *definition trees*, not to be confused with the tree denoted by the term in the semantics. The leaves of definition tree are among the constants of $\mathcal{N}$: C (combinators), $< \_, \_ >$, π_1, π_2, $\mathtt{cons}$, $\mathtt{f}$ (constructors and future constructors), $\mathtt{r}$, $\mathtt{u}$. There are unary branching for $\mathtt{F}$ and binary branching for applications. Each constant $\mathtt{j}_X$ with $X = \{t_i | i \in I\}$ has one child t_i for each $i \in I$. Remark that we do not consider a constant $\mathtt{j}_X$ as a leaf of the definition tree, as one would expect. In order to completely describe the

(possibly infinite) set X we have to consider the elements of X as children of $\mathtt{j}_X$.

The cardinality of a definition tree is at most the (infinite) cardinality of $\mathcal{I}$. If $\mathcal{I} = \mathtt{N}$, any definition tree (hence any term) may be represented by some set of integers, any at most countable set of terms may be represented by some set of integers, and any property of terms by a predicate on sets of integers.

Chapter 2

Models of $\mathcal{N}$

We introduce a semantics of system $\mathcal{N}$: models of $\mathcal{N}$ are extensions $\mathcal{N}+\mathcal{I}$ of $\mathcal{N}$.

2.1. Definition of Model Assume that $I \subseteq \mathcal{I}$ and that $\mathcal{A} : I \to \mathcal{N}+\mathcal{I}$. $\mathcal{A}$ is any list of terms of $\mathcal{N}+\mathcal{I}$, whose cardinality is at most $\mathcal{I}$. If $\mathcal{I} = \mathtt{N}$, then $\mathcal{A}$ may be represented by some set of integers. We write $|\mathcal{A}| = \{t|\exists i \in I.t \equiv \mathcal{A}(i)\}$ and $t \in \mathcal{A}$ for $t \in |\mathcal{A}|$. We identify $\mathcal{A}, \mathcal{B}$ if $|\mathcal{A}| = |\mathcal{B}|$, we write $\mathcal{A} \subseteq \mathcal{B}$ for $|\mathcal{A}| \subseteq |\mathcal{B}|$. For any context (list of pending extensions) Γ, any $A \in \mathcal{T}$ we define $(\Gamma \vdash A)_{\mathcal{A}} = \{t \in |\mathcal{A}||\Gamma \vdash t : A\}$ and $A_{\mathcal{A}} = (\mathtt{nil} \vdash A)_{\mathcal{A}}$. If $\mathcal{A} \subseteq \mathcal{B}$ then $(\Gamma \vdash A)_{\mathcal{A}} \subseteq (\Gamma \vdash A)_{\mathcal{B}}$ and $A_{\mathcal{A}} \subseteq A_{\mathcal{B}}$.

A $\mathcal{I}$-model is any list of terms $\mathcal{A}$, such that $|\mathcal{A}|$ is some extension of $\mathcal{N}$ with new constants all of the form $\mathtt{j}_{X,E,A}$, and $\mathcal{A}$ includes with each $\mathtt{j}_{X,E,A}$ all $\mathtt{j}_{X,F,B}$ with $F \in \mathtt{D}$, $B \in \mathtt{Tp}$, and all $t \in X$.

If $\mathcal{I} = \mathtt{N}$ and $\mathcal{A}$ is a $\mathcal{I}$-model then $|\mathcal{A}|$ is countable.

If $\mathcal{A}$ is a model, then by definition $|\mathcal{A}|$ is closed by context lifting: if $a \in (\Gamma, \Delta \vdash A)_{\mathcal{A}}$ and $\Gamma = E_0, \ldots, E_{i-1}$ then $a^{i,E} \in (\Gamma, E, \Delta \vdash A)_{\mathcal{A}}$.

The smallest the model has universe the set of terms of $\mathcal{N}$ itself, having no approximation constants. By a notation abuse we denote it with $\mathcal{N}$ again.

For any list $\mathcal{X} : I \to \mathcal{N}+\mathcal{I}$ we may define a list $(\mathcal{X}) : I \to \mathcal{N}+\mathcal{I}$, defining the smallest model including $|\mathcal{X}|$. We define a countable set of models as the smallest set such that if $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathtt{Mod}$ and $X_1 = (E_1)_{\mathcal{A}_1}, \ldots, X_n = (E_n)_{\mathcal{A}_n}$ then $(\mathtt{j}_{X_1}, \ldots, \mathtt{j}_{X_n}) \in \mathtt{Mod}$. We define the set of all terms in some model in $\mathtt{Mod}$ as $|\mathtt{Mod}| = \bigcup\{|\mathcal{A}||\mathcal{A} \in \mathtt{Mod}\}$.

For any constructor c in $\mathcal{A}$ we define $\mathtt{I}_{\mathcal{A}}(c)$, the trace of $\sim$ on (the set denoted by) $\mathtt{I}(c)$. When $c = \mathtt{cons}, \mathtt{f}$ and $\mathtt{I}(c) = E$ we set $\mathtt{I}_{\mathcal{A}}(c) = (E_{\mathcal{A}} \times E_{\mathcal{A}}) \cap \sim$. When $c \equiv \mathtt{j}_X$ and $\mathtt{I}(c) = X \subseteq E_{\mathcal{A}}$ we set $\mathtt{I}_{\mathcal{A}}(c) = (X \times X) \cap \sim$. By definition unfolding, if $\mathcal{A} \subseteq \mathcal{B}$ then $E_{\mathcal{A}} \subseteq E_{\mathcal{B}}$ for all $E \in \mathtt{D}$, hence

$\mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{\mathcal{B}}(c)$.

2.2. A Partial Equivalence Relation for $\mathcal{N}$ We have two ways of proving that system $\mathcal{N}$ normalizes.

A first way is to adapt Tait's notion of totality to any model $\mathcal{A} \in \mathtt{Mod}$ of system $\mathcal{N}$. We denote with $\mathtt{T}_{\mathcal{A}}$ the set of total terms of $\mathcal{A}$. We define $t \in \mathtt{T}_{\mathcal{A}}$ by principal induction on the type A of t. If $A = D \in \mathtt{D}$, then $t \in \mathtt{T}_{\mathcal{A}}$ if $t \rightsquigarrow^* c(f)$ for some $c = \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$, some $f : E \to D$, and for all $u \in \mathtt{T}_{\mathcal{A}}$ if $u \in \mathtt{I}(c)$ then $f(u) \in \mathtt{T}_{\mathcal{A}}$. $t : A \times B$ is total in $\mathcal{A}$ if $\pi_1(t), \pi_2(t)$ are. $f : A \to B$ is total if $f(a)$ is total in $\mathcal{A}$ for all a which are total *in all* $\mathcal{C} \ni a$. We may prove that all terms of $\mathcal{N} + \mathcal{I}$ are total. By definition every total term t normalizes and if $t : D \in \mathtt{D}$ then t denotes a well-founded tree.

We have a second way of proving that $\mathcal{N}$ normalizes. We define an extensional model for $\mathcal{N}$ by adapting Longo-Moggi Partial Equivalence Relations model. We define a partial equivalence relation $\sim_{\mathcal{A}}$ in every model $\mathcal{A}$, and prove that all these relations are reflexive (all are equivalence relations). Also in this case, we deduce that all terms normalize and if they have type some $D \in \mathtt{D}$ they denote well-founded trees.

The two proofs are quite similar, step by step. We choose the proof using Partial Equivalence Relations because it has an advantage: it defines a model for extendable recursion.

We inductively define a set of Partial Equivalence Relations $\sim_{\mathcal{A}} \subseteq \mathcal{A} \times \mathcal{A}$ for any $\mathcal{A} \in \mathtt{Mod}$. We write $t \sim u$ for $t, u \in |\mathtt{Mod}|$ and $t \sim_{\mathcal{A}} u$ for all $\mathcal{A} \in \mathtt{Mod}$ such that $t, u \in \mathcal{A}$.

Wee inductively define $\sim_{\mathcal{A}}$, and $\sim$ from it.

The relation $\sim_{\mathcal{A}}$ Assume $\mathcal{A} \in \mathtt{Mod}$ is some model and $t_1, t_2 \in (\Gamma \vdash A)_{\mathcal{A}}$ and $X = \{e_i | i \in I\}$.

1. Let $A \equiv D \in \mathtt{D}$. Then $t_1 \sim_{\mathcal{A}} t_2$ if for some $c = \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$, some f_1, f_2 we have $t_j \rightsquigarrow^* c(f_j)$ for $j = 1, 2$, and for all $a_1 \mathtt{I}_{\mathcal{A}}(c) a_2$ we have: $f_1(a_1^{\Gamma}) \sim_{\mathcal{A}} f_2(a_2^{\Gamma})$.

2. Let $A \equiv (B \to C)$. Then $t_1 \sim_{\mathcal{A}} t_2$ if for all $a_1, a_2 \in (\Gamma \vdash B)_{\mathcal{A}}$, if $a_1 \sim a_2$ then $f_1(a_1) \sim_{\mathcal{A}} f_2(a_2)$.

3. Let $A \equiv (B \times C)$. Then $t_1 \sim_{\mathcal{A}} t_2$ if $\pi_i(t_1) \sim_{\mathcal{A}} \pi_i(t_2)$ for $i = 1, 2$.

In point 1 above we wrote $f_1(a_1^{\Gamma})$ instead of $f_1(a_1)$. The reason is that f_1 is in the context Γ while a_1 is in the context $\mathtt{nil}$, so we have to move a_1 in Γ.

Chapter 3

The Pruning Operator of $\mathcal{N}$

Our goal is proving that all relations $\sim_{\mathcal{A}}$ are reflexive. As a consequence, we will derive that all computations in $\mathcal{N}+\mathcal{I}$ terminate and all terms of $\mathcal{N}+\mathcal{I}$ denoting a term denote a well-founded tree. For this proof, for each term t in a model $\mathcal{A}$ we have to form some term t^{Γ} denoting an isomorphic tree if t is a tree, but without future constructors. We call this operation *pruning*, then we prove that it is an inverse of lifting. We define pruning by removing one future constructor at the time.

3.1. Definition of Pruning Assume $X : I \to E_{\mathcal{A}}$ (X is any list of terms of type E in the context `nil` and in $\mathcal{A}$). We define a syntactical operation $(.)^{-m,E,X}$ we call *pruning*. We usually skip the superscripts E, X and we write $(.)^{-m}$. Assume $\Gamma, E, \Delta \vdash t : A$ and $u \equiv t^{-m}$. Then $\Gamma, \Delta \vdash u : A$: u has one future constructor less than t. We use pruning in order to observe the behavior of t in $\mathcal{A}$ through the behavior of a term u, having one future constructor less, and belonging to some future model $\mathcal{B} \supseteq \mathcal{A}$ expanding $\mathcal{A}$. By repeatedly applying pruning we may remove all future constructors from the context Γ of t, forming a term $t^{-\Gamma}$ in the empty context.

Pruning Assume $\mathcal{A}$ is some model, $E \in \mathtt{D}$ and X any listing of some terms of $E_{\mathcal{A}}$. Let $\Gamma = E_0, \ldots, E_{m-1}$ and $t : A$ be a term in the context Γ, E, Δ. We inductively define $t^{-m,E,X} : A$ in the context Γ, Δ. We write t^{-m} skipping the superscripts E, X for brevity.

- $c^{-m} \equiv c$ for any constant $c \not\equiv \mathtt{f}$ of $\mathcal{N}$ + constants.

($\mathtt{f}_<$) $\mathtt{f}^{-m}_{j,E,A} \equiv \mathtt{f}_{j,E,A}$ for all $j \leq m-1$

($\mathtt{f}_=$) $\mathtt{f}^{-m}_{j,E,A} \equiv \mathtt{j}_{X,E,A}$ if $j = m$

($\mathtt{f}^-_<$) $\mathtt{f}^{-m}_{j,E,A} \equiv \mathtt{f}_{j-1,E,A}$ for all $m+1 \leq j$

($\mathtt{F}_\leq$) $(\mathtt{F}_{j,E}.u)^{-m} \equiv \mathtt{F}_{j,E}.u^{-(m+1)}$ for all $j \leq m$.

($\mathtt{F}^-_>$) $(\mathtt{F}_{j,E}.u)^{-m} \equiv \mathtt{F}_{j-1,E}.u^{-m}$ for all $m+1 \leq j$

- $t(u)^{-m} \equiv t^{-m}(u^{-m})$

If $\Gamma \vdash a : A$, we define $a^{-\Gamma,\mathcal{A}} \equiv (\ldots(a^{-(m-1),E_{m-1},\mathcal{A}})\ldots)^{-0,E_0,\mathcal{A}}$ in the context $\mathtt{nil}$.

3.2. Pruning and Lifting Pruning is the opposite operation of Lifting. By definition unfolding, we will check that for any $\Gamma, \Delta \vdash t : A$, if $k = m$ we have $(t^{k,E})^{-m,E,\mathcal{A}} \equiv t$. In the case k, m are different, moving m before k decreases k in the case $m < k$, otherwise it increases m.

Lemma 3.2.1 (Pruning and Lifting 1) *For all $k, m \in \mathtt{N}$, if $m+1 \leq k$ then $(a^k)^{-m} \equiv (a^{-m})^{k-1}$*

Proof By induction on a.

1. Assume a is a constant $c \not\equiv \mathtt{f}$. Then $(c^k)^{-m} \equiv c \equiv (c^{-m})^{k-1}$.
2. Assume a is a constant $c \equiv \mathtt{f}_{j,E}$. We distinguish four cases: $(j \leq m-1)$, $(j = m)$, $(m+1 \leq j \leq k-1)$ and $(k \leq j)$.
   (a) Assume $(j \leq m-1)$: from $m+1 \leq k$ we get $j \leq k-2$. Then:
   $(\mathtt{f}^k_{j,E})^{-m} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}^{-m}_{j,E} \equiv$ (by $j \leq m-1$)
   $\mathtt{f}_{j,E}$
   and:
   $(\mathtt{f}^{-m}_{j,E})^{k-1} \equiv$ (by $j \leq m-1$)
   $\mathtt{f}^{k-1}_{j,E} \equiv$ (by $j \leq k-2$)
   $\mathtt{f}_{j,E}$
   (b) Assume $j = m$: from $m+1 \leq k$ we get $j \leq k-1$. Then:
   $(\mathtt{f}^k_{j,E})^{-m} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}^{-m}_{j,E} \equiv$ (by $j = m$)
   $\mathtt{j}_X$
   and:
   $(\mathtt{f}^{-m}_{j,E})^{k-1} \equiv$ (by $j = m$)
   $\mathtt{j}_X{}^{k-1} \equiv$
   $\mathtt{j}_X$

(c) Assume $m+1 \leq j \leq k-1$: hence $m \leq j-1 \leq k-2$. Then:
$(\mathtt{f}^k_{j,E})^{-m} \equiv$ (by $j \leq k-1$)
$\mathtt{f}^{-m}_{j,E} \equiv$ (by $m+1 \leq j$)
$\mathtt{f}_{j-1,E}$
and:
$(\mathtt{f}^{-m}_{j,E})^{k-1}$ (by $m \leq j-1$)
$\mathtt{f}^{k-1}_{j-1,E} \equiv$ (by $j-1 \leq k-2$)
$\mathtt{f}_{j-1,E}$

(d) Assume $k \leq j$: then $k-1 \leq j-1$ and $m+1 \leq k \leq j \leq j+1$.
We have:
$(\mathtt{f}^k_{j,E})^{-m} \equiv$ (by $k \leq j$)
$\mathtt{f}^{-m}_{j+1,E} \equiv$ (by $m+1 \leq j+1$)
$\mathtt{f}_{j,E}$
and:
$(\mathtt{f}^{-m}_{j,E})^{k-1} \equiv$ (by $m+1 \leq j$)
$\mathtt{f}^{k-1}_{j-1,E} \equiv$ (by $k-1 \leq j-1$)
$\mathtt{f}_{j,E}$

3. Assume $a \equiv \mathtt{F}_{j,E}.b$. We distinguish three cases: $(j \leq m)$, $(m+1 \leq j \leq k-1)$ and $(k \leq j)$.

(a) Assume $(j \leq m)$: from $m+1 \leq k$ we get $j \leq k-1$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-m} \equiv$ (by $j \leq k-1$)
$(\mathtt{F}_{j,E}.b^{k+1})^{-m} \equiv$ (by $j \leq m$)
$\mathtt{F}_{j,E}.(b^{k+1})^{-(m+1)}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^{k-1} \equiv$ (by $j \leq m$)
$(\mathtt{F}_{j,E}.b^{-(m+1)})^{k-1} \equiv$ (by $j \leq k-1$)
$\mathtt{F}_{j,E}.(b^{-(m+1)})^k$
By induction hypothesis on $b, k+1, m+1$ and $(m+1)+1 \leq k+1$ we have $(b^{k+1})^{-(m+1)} \equiv (b^{-(m+1)})^k$. It follows the thesis.

(b) Assume $m+1 \leq j \leq k-1$: hence $m \leq j-1 \leq k-2$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-m} \equiv$ (by $j \leq k-1$)
$(\mathtt{F}_{j,E}.b^{k+1})^{-m} \equiv$ (by $m+1 \leq j$)
$\mathtt{F}_{j-1,E}.(b^{k+1})^{-m}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^{k-1} \equiv$ (by $m+1 \leq j$)
$(\mathtt{F}_{j-1,E}b^{-m})^{k-1} \equiv$ (by $j-1 \leq k-2$)

$\mathtt{F}_{j-1,E}.b^{-m})^k$
By induction hypothesis on $b, k+1, m$ and $m+1 \leq k \leq k+1$ we have $(b^{k+1})^{-m} \equiv (b^{-m})^k$. It follows the thesis.

(c) Assume $k \leq j$: then $k-1 \leq j-1$ and $m+1 \leq k \leq j \leq j+1$.
We have:
$((\mathtt{F}_{j,E}.b)^k)^{-m} \equiv$ (by $k \leq j$)
$(\mathtt{F}_{j+1,E}.b^k)^{-m} \equiv$ (by $m+1 \leq j+1$)
$\mathtt{F}_{j,E}.b^k)^{-m}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^{k-1} \equiv$ (by $m+1 \leq j$)
$(\mathtt{F}_{j-1,E}.b^{-m})^{k-1} \equiv$ (by $k-1 \leq j-1$)
$\mathtt{F}_{j,E}.(b^{-m})^{k-1}$
By induction hypothesis on b, k, m and $m+1 \leq k$ we have $(b^k)^{-m} \equiv (b^{-m})^{k-1}$. It follows the thesis.

4. Assume $a \equiv b(c)$. Then $(b(c)^k)^{-m} \equiv (b^k)^{-m}((c^k)^{-m}) \equiv$ (by inductive hypothesis on b, c) $(b^{-m})^{k-1}(c^{-m})^{k-1}) \equiv b(c)^{-m})^{k-1}$.

Lemma 3.2.2 (Pruning and Lifting 2) *For all $k, m \in \mathtt{N}$, if $k \leq m$ then $(a^k)^{-(m+1)} \equiv (a^{-m})^k$.*

Proof By induction on a.

1. Assume a is a constant $c \not\equiv \mathtt{f}$. Then $(c^k)^{-(m+1)} \equiv c \equiv (c^{-m})^k$.

2. Assume a is a constant $c \equiv \mathtt{f}_{j,E}$. We distinguish four cases: $(j \leq k-1)$, $(k \leq j \leq m-1)$, $(j = m)$ and $(m+1 \leq j)$.

   (a) Assume $(j \leq k-1)$: from $k \leq m$ we get $j \leq m-1, m$. Then:
   $(\mathtt{f}^k_{j,E})^{-(m+1)} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}^{-(m+1)}_{j,E} \equiv$ (by $j \leq m$)
   $\mathtt{f}_{j,E}$
   and:
   $(\mathtt{f}^{-m}_{j,E})^k \equiv$ (by $j \leq m-1$)
   $\mathtt{f}^k_{j,E} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}_{j,E}$

   (b) Assume $k \leq j \leq m-1$. Then:
   $(\mathtt{f}^k_{j,E})^{-(m+1)} \equiv$ (by $k \leq j$)
   $\mathtt{f}^{-m}_{j+1,E} \equiv$ (by $j \leq m-1$)
   $\mathtt{f}_{j+1,E}$
   and:

$(\mathtt{f}_{j,E}^{-m})^k \equiv$ (by $j \leq m-1$)
$\mathtt{f}_{j,E}^{k} \equiv$ (by $k \leq j$)
$\mathtt{f}_{j+1,E}$

(c) Assume $j = m$: from $k \leq m$ we get $k \leq j$ and $j+1 = m+1$. Then:
$(\mathtt{f}_{j,E}^{k})^{-(m+1)} \equiv$ (by $k \leq j$)
$\mathtt{f}_{j+1,E}^{-(m+1)} \equiv$ (by $j+1 = m+1$)
$\mathtt{j}_X$
and:
$(\mathtt{f}_{j,E}^{-m})^k \equiv$ (by $j = m$)
$\mathtt{j}_X{}^k \equiv$
$\mathtt{j}_X$

(d) Assume $m+1 \leq j$: from $k \leq m$ we get $k \leq j-1 \leq j$. Then:
$(\mathtt{f}_{j,E}^{k})^{-(m+1)} \equiv$ (by $k \leq j$)
$\mathtt{f}_{j+1,E}^{-(m+1)} \equiv$ (by $m+1 \leq j+1$)
$\mathtt{f}_{j,E}$
and:
$(\mathtt{f}_{j,E}^{-(m+1)})^k \equiv$ (by $m+1 \leq j$)
$\mathtt{f}_{j-1,E}^{k} \equiv$ (by $k \leq j-1$)
$\mathtt{f}_{j,E}$

3. Assume $a \equiv \mathtt{F}_{j,E}.b$. We distinguish three cases: $(j \leq k-1)$, $(k \leq j \leq m)$ and $(m+1 \leq j)$.

(a) Assume $(j \leq k-1)$: from $k \leq m$ we get $j \leq m-1$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-(m+1)} \equiv$ (by $j \leq k-1$)
$(\mathtt{F}_{j,E}.b^{k+1})^{-(m+1)} \equiv$ (by $j \leq m+1$)
$\mathtt{F}_{j,E}.(b^{k+1})^{-(m+2)}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^k \equiv$ (by $j \leq m$)
$(\mathtt{F}_{j,E}.b^{-(m+1)})^k \equiv$ (by $j \leq k-1$)
$\mathtt{F}_{j,E}.(b^{-(m+1)})^{k+1}$
By induction hypothesis on $b, k+1, m+1$ and $k+1 \leq m+1$ we have $(b^{k+1})^{-(m+2)} \equiv (b^{-(m+1)})^{k+1}$. It follows the thesis.

(b) Assume $k \leq j \leq m$: we get $k \leq m+1$ and $j+1 \leq m+1$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-(m+1)} \equiv$ (by $k \leq j$)
$(\mathtt{F}_{j+1,E}.b^k)^{-(m+1)} \equiv$ (by $j+1 \leq m+1$)

$\mathtt{F}_{j+1,E}.(b^k)^{-(m+2)}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^k$ (by $j \leq m$)
$(\mathtt{F}_{j,E}.b^{-(m+1)})^k \equiv$ (by $k \leq j$)
$\mathtt{F}_{j+1,E}.(b^{-(m+1)})^k$
By induction hypothesis on $b, k, m+1$ and $k \leq m+1$ we have $(b^k)^{-(m+2)} \equiv (b^{-(m+1)})^k$. It follows the thesis.

(c) Assume $m+1 \leq j$: from $k \leq m$ we get $k \leq j-1$ and $m+2 \leq j+1$.
We have:
$((\mathtt{F}_{j,E}.b)^k)^{-(m+1)} \equiv$ (by $k \leq j$)
$(\mathtt{F}_{j+1,E}.b^k)^{-(m+1)} \equiv$ (by $m+2 \leq j+1$)
$\mathtt{F}_{j,E}.b^k)^{-(m+1)}$
and:
$((\mathtt{F}_{j,E}.b)^{-m})^k \equiv$ (by $m+1 \leq j$)
$(\mathtt{F}_{j-1,E}.b^{-m})^k \equiv$ (by $k \leq j-1$)
$\mathtt{F}_{j,E}.(b^{-m})^k$
By induction hypothesis on b, k, m and $k \leq m$ we have $(b^k)^{-(m+1)} \equiv (b^{-m})^k$. It follows the thesis.

4. Assume $a \equiv b(c)$. Then $(b(c)^k)^{-(m+1)} \equiv (b^k)^{-(m+1)}((c^k)^{-(m+1)}) \equiv$ (by inductive hypothesis on b, c and k, m) $(b^{-m})^{k-1}((c^{-m})^{k-1}) \equiv (b(c)^{-m})^{k-1}$.

Lemma 3.2.3 (Pruning and Lifting 3) *For all $k \in \mathtt{N}$ we have $(a^k)^k \equiv a$.*

Proof By induction on a.

1. Assume a is a constant $c \not\equiv \mathtt{f}$. Then $(c^k)^{-k} \equiv c$.

2. Assume a is a constant $c \equiv \mathtt{f}_{j,E}$. We distinguish the cases: $(j \leq k-1)$ and $(k \leq j)$.

   (a) Assume $(j \leq k-1)$. Then:
   $(\mathtt{f}^k_{j,E})^{-k} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}^{-k}_{j,E} \equiv$ (by $j \leq k-1$)
   $\mathtt{f}_{j,E}$

   (b) Assume $k \leq j$. Then:
   $(\mathtt{f}^k_{j,E})^{-k} \equiv$ (by $k \leq j$)
   $\mathtt{f}^{-k}_{j+1,E} \equiv$ (by $k+1 \leq j+1$)
   $\mathtt{f}_{j,E}$

3. Assume $a \equiv \mathtt{F}_{j,E}.b$. We distinguish the cases: $(j \leq k-1)$ and $(k \leq j)$.

(a) Assume $(j \leq k-1)$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-k} \equiv$ (by $j \leq k-1$)
$(\mathtt{F}_{j,E}.b^{k+1})^{-k} \equiv$ (by $j \leq k-1$)
$\mathtt{F}_{j,E}.(b^{k+1})^{-(k+1)}$
and:
$((\mathtt{F}_{j,E}.b)^{-k})^k \equiv$ (by $j \leq k$)
$(\mathtt{F}_{j,E}.b^{-(k+1)})^k \equiv$ (by $j \leq k-1$)
$\mathtt{F}_{j,E}.(b^{-(k+1)})^{k+1}$
By induction hypothesis on $b, k+1$ we have $(b^{k+1})^{-(k+1)} \equiv b$. It follows the thesis.

(b) Assume $k \leq j$. Then:
$((\mathtt{F}_{j,E}.b)^k)^{-k} \equiv$ (by $k \leq j$)
$(\mathtt{F}_{j+1,E}.b^k)^{-k} \equiv$ (by $k+1 \leq j+1$)
$\mathtt{F}_{j,E}.(b^k)^{-k}$
By induction hypothesis on b, k we have $(b^k)^{-k} \equiv b$. It follows the thesis.

4. Assume $a \equiv b(c)$. Then $(b(c)^k)^{-k} \equiv (b^k)^{-k}((c^k)^{-k}) \equiv$ (by inductive hypothesis on b, c and k) $b(c)$.

Corollary 3.2.4 *Let $\Gamma = E_0, \ldots, E_{m-1}$ and a be any term.*

1. $(a^\Gamma)^{-\Gamma} \equiv a$.
2. $(a^{\Gamma,E,\Delta})^{-m,E,\mathcal{A}} \equiv a$

Proof 1. We have $(a^\Gamma)^{-\Gamma} \equiv (\ldots((\ldots(a^0)\ldots)^{m-1})^{-(m-1)}\ldots)^{-0}$. If we apply Lemma 3.2.3 for m times we obtain $(a^\Gamma)^{-\Gamma} \equiv a$.

2. We prove $(a^{\Gamma,E,\Delta})^{-m,E,\mathcal{A}} \equiv a$ by induction on Δ. Assume $\Delta = \emptyset$. Then $(a^\Gamma, E)^{-m,E,\mathcal{A}} \equiv ((a^\Gamma)^{m,E})^{-m,E,\mathcal{A}} \equiv$ (by Lemma 3.2.3) a^Γ. Assume $\Delta = \Theta, D$ has length $n+1$. Then $(a^{\Gamma,E,\Theta,D})^{-m,E,\mathcal{A}} \equiv ((a^{\Gamma,E,\Theta})^{m+n+1,D})^{-m,E,\mathcal{A}} \equiv$ (by Lemma 3.2.1) $((a^{\Gamma,E,\Theta})^{-m,E,\mathcal{A}})^{m+n,D} \equiv$ (by induction hypothesis) $(a^{\Gamma,\Theta})^{m+n,D} \equiv a^{\Gamma,\Theta,D} \equiv a^{\Gamma,\Delta}$.

3.3. Pruning and Reductions

Pruning is compatible with reduction and allows to interpret terms with future constructors into terms without them.

Lemma 3.3.1 (The Pruning Lemma) *Assume $\mathcal{A}$ is a model of $\mathcal{N} + \mathcal{I}$. Denote with $X : I \to E_\mathcal{A}$ any enumeration of some terms of $E_\mathcal{A}$. Denote with $\mathcal{B}^X$ one enumeration of the closure of $\mathcal{A}, \mathcal{B}$ w.r.t. $(.)^X$. Assume $E, D \in \mathtt{D}$ are data types and $e : E$.*

If $t \rightsquigarrow u$ then $t^X \rightsquigarrow u^X$ and if t is normal then t^X is normal.

Proof of Pruning Lemma Assume $t \rightsquigarrow u$. We prove $t^X \rightsquigarrow u^X$ by induction on the definition of $\rightsquigarrow$.

1. Assume $t \equiv C\vec{t} \rightsquigarrow \alpha[\vec{t}/\vec{x}] \equiv u$ for some combinator C. Then $t^X \equiv C\vec{t^X} \rightsquigarrow \alpha[\vec{t^X}/\vec{x}] \equiv$ (by commutation of $(.)^X$ and application) u^X.

2. Assume $t \equiv fa \rightsquigarrow ga \equiv u$ and $f \rightsquigarrow g$. By induction hypothesis we have $f^X \rightsquigarrow g^X$. We conclude that $t^X \equiv f^X a^X \rightsquigarrow g^X a^X \equiv u^X$.

3. Assume $t \equiv \pi_i(< a_1, a_2 >) \rightsquigarrow a_i \equiv u$. Then $t^X \equiv \pi_i(< a_1^X, a_2^X >) \rightsquigarrow a_i^X \equiv u^X$.

4. Assume $t \equiv \pi_i(a) \rightsquigarrow \pi_i(b) \equiv u$ and $a \rightsquigarrow b$. By induction hypothesis we have $a^X \rightsquigarrow b^X$. We conclude that $t^X \equiv \pi_i(a^X) \rightsquigarrow \pi_i(b^X) \equiv u^X$.

5. Assume $t \equiv \mathtt{r}\vec{r}\mathtt{cons}(f) \rightsquigarrow r_i(\mathtt{r}\vec{r} \circ f) \equiv u$. Then $t^X \equiv \mathtt{r}\vec{r^X}\mathtt{cons}(f^X) \rightsquigarrow r_i^X(\mathtt{r}\vec{r^X} \circ f^X) \equiv u^X$.

6. Assume $t \equiv \mathtt{r}\vec{r}c(f) \rightsquigarrow r_i c(\mathtt{r}\vec{r} \circ f) \equiv u$ for some $c \equiv \mathtt{f}, \mathtt{j}_Y$ and some Y. Then $c^X \equiv \mathtt{f}, \mathtt{j}_Z$ for some $Z = Y, X$. We conclude $t^X \equiv \mathtt{r}\vec{r^X}c^X(f^X) \rightsquigarrow r_n^X c^X(\mathtt{r}\vec{r^X} \circ f^X) \equiv u^X$.

7. Assume $t \equiv \mathtt{F}_{k,D}.c(f)$. We reason by cases on c.

   Assume $c \not\equiv \mathtt{f}$. Then $t \rightsquigarrow c(\mathtt{F}_{k,D}.f) \equiv u$. By definition of $(.)^{-m}$, for some k', m' depending on k, m we have $t^{-m} \equiv \mathtt{F}_{k',D}.c(f^{-m'})$ and $u^X \equiv c(\mathtt{F}_{k',D}.f^{-m'})$. From $c \not\equiv \mathtt{f}$ conclude that $t^X \rightsquigarrow u^X$.

   Assume $c \equiv \mathtt{f}_{h,F,B}$. We distinguish the sub-cases $h < k$, $h = k$, $h > k$.

   (a) Assume $h < k$. Then $t \rightsquigarrow \mathtt{f}_h(\mathtt{F}_{k,D}.f) \equiv u$.
   If $h < k \leq m$, then $t^X \equiv \mathtt{F}_{k,D}.\mathtt{f}_h(f^{-(m+1)}) \rightsquigarrow \mathtt{f}_h(\mathtt{F}_{k,D}.f^{-(m+1)}) \equiv u^X$.
   If $h < m < k$, then $t^X \equiv \mathtt{F}_{k-1,D}.\mathtt{f}_h(f^{-m}) \rightsquigarrow \mathtt{f}_h(\mathtt{F}_{k-1,D}.f^{-m}) \equiv u^X$.
   If $m = h < k$, then $t^X \equiv \mathtt{F}_{k-1,D}.\mathtt{f}_{h-1}(f^{-m}) \rightsquigarrow \mathtt{f}_{h-1}(\mathtt{F}_{k-1,D}.f^{-m}) \equiv u^X$.
   If $m < h < k$, then $t^X \equiv \mathtt{F}_{k-1,D}.\mathtt{cons}_n(f^{-m}) \rightsquigarrow \mathtt{cons}_n(\mathtt{F}_{k-1,D}.f^{-m}) \equiv u^X$.

   (b) Assume $k = h$. Then $t \rightsquigarrow \mathtt{cons}_n(\mathtt{F}_{k,D}.f) \equiv u$.
   If $h = k \leq m$, then $t^X \equiv \mathtt{F}_{k,D}.\mathtt{f}_k(f^{-(m+1)})$ (because $k < m+1$), hence $t^X \rightsquigarrow \mathtt{cons}_n(\mathtt{F}_{k,D}.f^{-(m+1)}) \equiv u^X$.
   If $h = k > m$, then $t^{-m} \equiv \mathtt{F}_{k-1,D}.\mathtt{f}_{k-1}(f^{-m}) \rightsquigarrow \mathtt{cons}_n(\mathtt{F}_{k-1,D}.f^{-m}) \equiv u^X$.

(c) Assume $h > k$. Then $t \rightsquigarrow \mathtt{f}_{h-1}(\mathtt{F}_{k,D}.f) \equiv u$.
If $k < h < m{+}1$, then $t^{-m} \equiv \mathtt{F}_{k,D}.\mathtt{f}_h(f^{-(m+1)}) \rightsquigarrow \mathtt{f}_{h-1}(\mathtt{F}_{k,D}.f^{-(m+1)}) \equiv u^X$.
If $k < h = m{+}1$, then $t^X \equiv \mathtt{F}_{k,D}.\mathtt{cons}_n(f^{-(m+1)}) \rightsquigarrow \mathtt{cons}_n(\mathtt{F}_{k,D}.f^{-(m+1)}) \equiv u^X$ (because $h - 1 = m$).
If $k \leq m < m{+}1 < h$, then $t^X \equiv \mathtt{F}_{k,D}.\mathtt{f}_{h-1}(f^{-(m+1)}) \rightsquigarrow \mathtt{f}_{h-2}(\mathtt{F}_{k,D}.f^{-(m+1)}) \equiv u^X$ (because $h - 1 > m \geq k$).
If $m < k < h$, then $t^X \equiv \mathtt{F}_{k-1,D}.\mathtt{f}_{h-1}(f^{-m}) \rightsquigarrow \mathtt{f}_{h-2}(\mathtt{F}_{k,D}.f^{-m}) \equiv u^X$ (because $h - 1 > m, k - 1$).

8. Assume $t \equiv \mathtt{F}_{k,D}.a \rightsquigarrow \mathtt{F}_{k,D}.b \equiv u$ for some $a \rightsquigarrow b$. Then for some k', m' depending on k, m we have $t^X \equiv \mathtt{F}_{k',D}.a^{-m'}$ and $u^X \equiv \mathtt{F}_{k',D}.b^{-m'}$. By induction hypothesis we have $a^{-m'} \rightsquigarrow b^{-m'}$. We conclude that $t^X \rightsquigarrow u^X$.

9. Assume $t \equiv (\mathtt{F}_{k,D}.f)(a) \rightsquigarrow \mathtt{F}_{k,D}.f(a^{k,D}) \equiv u$. We distinguish the cases $k > m, k \leq m$. Assume $k > m$. Then $t^X \equiv (\mathtt{F}_{k-1,D}.f^{-m})(a^{-m})$ and $u^X \equiv \mathtt{F}_{k-1,D}.f^{-m}((a^{k,D})^{-m})$. By Lemma 3.2.1: $(a^{k,D})^{-m} \equiv (a^{-m})^{k-1,D})$
We deduce $u^{-m} \equiv \mathtt{F}_{k-1,D}.f^{-m}((a^{-m})^{k-1,D})$, hence $t^X \rightsquigarrow u^X$.
Assume $k \leq m$. Then $t^X \equiv (\mathtt{F}_{k,D}.f^{-(m+1)})(a^{-m})$ and $u^X \equiv \mathtt{F}_{k,D}.f^{-(m+1)}((a^{k,D})^{-(m+1)})$. By Lemma 3.2.2: $(a^{k,D})^{-(m+1)} \equiv (a^{-m})^{k,D})$
We deduce $u^{-m} \equiv \mathtt{F}_{k,D}.f^{-(m+1)}((a^{-(m+1)})^{k,D})$, hence $t^X \rightsquigarrow u^X$.

10. Assume $t \equiv \pi_i(\mathtt{F}_{k,D}.f) \rightsquigarrow \mathtt{F}_{k,D}.(\pi_i \circ f) \equiv u$. Then for some k', m' depending on k, m we have $t^X \equiv \pi_i(\mathtt{F}_{k',D}.f^{-m'})$ and $u^X \equiv \mathtt{F}_{k',D}.f^{-m'}(a^{-m'})$. We conclude that $t^X \rightsquigarrow u^X$.

We may prove by cases that if t is normal then t^X is normal.

Chapter 4

The normalization proof for the class of models Mod

We prove that all relations $\sim_{\mathcal{A}}$ are reflexive, for all $\mathcal{A} \in \mathtt{Mod}$.

4.1. Rules for $\sim$ The relation $\sim_{\mathcal{A}}$ in a model (weakly) decrease when the model increases, and it is invariant under reductions (in both directions).

Proposition 4.1.1 (Totality Rules) *1.* Antimonotonicity rule. *Assume $\mathcal{A} \subseteq \mathcal{B}$. If $a_1, a_2 \in \mathcal{A}$ and $a_1 \sim_{\mathcal{B}} a_2$ then $a_1 \sim_{\mathcal{A}} a_2$.*

2. Reduction rule. *Assume $t_1, t_2 \in (\Gamma \vdash A)_{\mathcal{A}}$ and $t_j \rightsquigarrow^* t'_j$ for $j = 1, 2$. Then $t_1 \sim_{\mathcal{A}} t_2 \Leftrightarrow t'_1 \sim_{\mathcal{A}} t'_2$.*

Proof 1. *Antimonotonicity rule.* Assume $a_1, a_2 \in (\Gamma \vdash A)_{\mathcal{A}} a_2$ and $a_1 \sim_{\mathcal{B}} a_2$. We prove $a \sim_{\mathcal{A}}$ by induction on the definition of $\sim_{\mathcal{B}}$.

(a) Assume $A = D \in \mathtt{D}$, and for some $c \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$, $f_1 \sim_{\mathcal{B}} f_2$ we have: $a_j \rightsquigarrow^* c(f_j)$ for $j = 1, 2$. By definition, for all $e_1 \mathtt{I}_{\mathcal{B}}(c) e_2$, $f_1(e_1^\Gamma) \sim_{\mathcal{B}} f_2(e_2^\Gamma)$. Then for all $(e_1, e_2) \in \mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{\mathcal{B}}(c)$, $f_1(e_1^\Gamma) \sim_{\mathcal{B}} f_2(e_2^\Gamma)$, and by induction hypothesis $f_1(e_1^\Gamma) \sim_{\mathcal{A}} f_2(e_2^\Gamma)$. We conclude that $c(f_1) \sim_{\mathcal{A}} c(f_2)$, that is, $a_1 \sim_{\mathcal{A}} a_2$.

(b) Assume $A = B \times C$ and for $j = 1, 2$, $\pi_j(a_1) \sim_{\mathcal{B}} \pi_j(a_2)$. By induction hypothesis we deduce that $\pi_j(a_1) \sim_{\mathcal{A}} \pi_j(a_2)$. We conclude that $a_1 \sim_{\mathcal{A}} a_2$.

(c) Assume $A = B \to C$ and for all $b_1, b_2 \in (\Gamma \vdash B)_{\mathcal{B}}$ if $b_1 \sim b_2$ then $a_1(b_1) \sim_{\mathcal{B}} a_2(b_2)$. Then for all $b_1, b_2 \in (\Gamma \vdash B)_{\mathcal{A}} \subseteq (\Gamma \vdash B)_{\mathcal{B}}$ if $b_1 \sim b_2$ then $a_1(b_1) \sim_{\mathcal{B}} a_2(b_2)$. By induction hypothesis we deduce

that $a_1(b_1) \sim_{\mathcal{A}} a_2(b_2)$, under the same hypothesis. We conclude that $a_1 \sim_{\mathcal{A}} a_2$.

2. *Reduction rule.* By induction on A.

   (a) Assume $A \equiv D \in \mathtt{D}$. Then $t_j \rightsquigarrow^* t'_j$ implies that t_j, t'_j have the same normal form $c(f_j)$, if they have any. Thus, the thesis holds by definition.

   (b) Assume $A \equiv B \times C$. Then: $t_1 \sim_{\mathcal{A}} t_2 \Leftrightarrow$ (by definition) for all $i = 1, 2$: $\pi_j(t_1) \sim_{\mathcal{A}} \pi_j(t_2) \Leftrightarrow$ (by $\pi_i(t_j) \rightsquigarrow^* \pi_i(t'_j)$ for $j = 1, 2$ and induction hypothesis on B, C) for all $i = 1, 2$: $\pi_i(t'_1) \sim_{\mathcal{A}} \pi_i(t'_2) \Leftrightarrow$ (by definition) $t'_1 \sim_{\mathcal{A}} t'_2$.

   (c) Assume $A \equiv B \to C$. Then: $t_1 \sim_{\mathcal{A}} t_2 \Leftrightarrow$ (by definition) for all $u_1, u_2 \in (\Gamma \vdash T)_{\mathcal{A}}$, if $u_1 \sim u_2$ then $t_1(u_1) \sim_{\mathcal{A}} t_2(u_2) \Leftrightarrow$ (by $t_j(u_j) \rightsquigarrow^* t'_j(u_j)$ and induction hypothesis on C) for all $u_1, u_2 \in (\Gamma \vdash T)_{\mathcal{A}}$, if $u_1 \sim u_2$ then $t'_1(u_1) \sim_{\mathcal{A}} t'_1(u_1)$. By definition, this latter is equivalent $t'_1 \sim_{\mathcal{A}} t'_2$.

4.2. Pruning and the relation $\sim$ Let $\Gamma = E_0, \ldots, E_{m-1}$ be context and $\mathcal{A} : I \to \mathcal{N} + \mathcal{I}$ be a model. Suppose $j = 0, \ldots, m-1$. Let $I_j = \{i \in I | \mathcal{A}(i) \in (E_j)_{\mathcal{A}}\}$: I_j is the set of indexes of terms of $(E_j)_{\mathcal{A}}$. Let $X_j = \{\mathcal{A}(i) | i \in I_j\}$ be the listing of $(E_j)_{\mathcal{A}}$ obtained by restricting the listing $\mathcal{A} : I \to \mathcal{N} + \mathcal{I}$ to I_j. We write $(.)^{-m,E,\mathcal{A}}$ for $(.)^{-m,E,X_j}$.

We will prove that the relation $\sim$ is invariant under $(.)^{-m,E,\mathcal{A}}$: for all terms t in the context Γ, E, Δ, $t_1 \sim t_2 \Leftrightarrow$ for all $\mathcal{A}$ we have $t_1^{-m,E,\mathcal{A}} \sim t_2^{-m,E,\mathcal{A}}$.

From now on, we denote with $\mathcal{B}^X$ (an arbitrary enumeration of) the closure of $\mathcal{A} \cup \mathcal{B}$ w.r.t. $(.)^X$: by definition, $\mathcal{B}^X$ is a model of $\mathcal{N}$ and we have $\mathcal{A}, \mathcal{B} \subseteq \mathcal{B}^X$.

Lemma 4.2.1 (Pruning and Index Set) *Assume $\mathcal{A} \in \mathtt{Mod}$. Denote with $X = X_j$ the listing of* all *terms of $E_{\mathcal{A}}$. Denote with $\mathcal{B}^X$ one enumeration of the closure of $\mathcal{A}, \mathcal{B}$ w.r.t. $(.)^X$. Assume $E, D \in \mathtt{D}$ are data types and $e : E$. Let $c = \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$. Then*

$$\mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{(\mathcal{B}^X)}(c^X) \subseteq \mathtt{I}_{(\mathcal{B}^X)}(c)$$

Proof By $\mathcal{A} \subseteq \mathcal{B}^{\mathcal{A}}$ we have $\mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{(\mathcal{B}^X)}(c)$. Now we argue by cases on c.

1. Assume $c = \mathtt{f}_{k,D}$ for some $m + 1 \leq k$. Then $c^X \equiv \mathtt{f}_{k-1,D}$, therefore $\mathtt{I}_{(\mathcal{B}^X)}(c^X) = \mathtt{I}_{(\mathcal{B}^X)}(c)$. We conclude $\mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{(\mathcal{B}^X)}(c) = \mathtt{I}_{(\mathcal{B}^X)}(c^X)$.

2. Assume $c = \mathtt{f}_{m,D}$. Then $c^X \equiv \mathtt{j}_X$, and by $X = E_{\mathcal{A}}$: $\mathtt{I}_{\mathcal{A}}(c) = E_{\mathcal{A}} \times E_{\mathcal{A}} \cap \sim \; = X \times X \cap \sim \; = \mathtt{I}_{(\mathcal{B}^X)}(c^X)$. Besides, we have $\mathtt{I}_{(\mathcal{B}^X)}(c^X) = X \times X \cap \sim \; = E_{\mathcal{A}} \times E_{\mathcal{A}} \cap \sim \; \subseteq E_{(\mathcal{B}^X)} \cap \sim \; = \mathtt{I}_{(\mathcal{B}^X)}(c)$.

3. In all other cases we have $c^X \equiv c$, therefore $\mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{(\mathcal{B}^X)}(c) = \mathtt{I}_{(\mathcal{B}^X)}(c^X)$.

Lemma 4.2.2 (Simulation) *Assume $\mathcal{A} \in \mathtt{Mod}$. Let $(.)^{\mathcal{A}} = (.)^{-m,E,X_j}$ with X_j a list of* all *terms of $E_{\mathcal{A}}$, and denote with X any listing of some terms of $E_{\mathcal{A}}$. Denote with $\mathcal{B}^{\mathcal{A}}$ one enumeration of the closure of $\mathcal{A}, \mathcal{B}$ w.r.t. $(.)^{\mathcal{A}}$. Assume $E, D \in \mathtt{D}$ are data types and $e : E$, $t : D$.*

1. *t normalizes to $u \Leftrightarrow t^X$ normalizes to u^X.*
2. *If t or t^X normalize, then both normalize and either*
   (a) *$t \equiv \mathtt{f}_{m,E}(f)$ and $t^X \equiv \mathtt{j}_{X_m}(f)$ or*
   (b) *$t \equiv \mathtt{f}_{k+1,E}(f)$ for some $k \geq m$ in t and $t^X \equiv \mathtt{f}_{k,E}(f^X)$ or*
   (c) *$t \equiv c(f)$, for some $c = \mathtt{cons}, \mathtt{f}_{k,E}, \mathtt{j}_X$ and some $k < m$, and $t^X \equiv c(f^X)$.*
3. *If $e_1 \sim_{\mathcal{B}^X} e_2$ then $e_1^X \sim_{\mathcal{B}^X} e_2^X$*
4. *Let $X = E_{\mathcal{A}}$ and write $t^{\mathcal{A}}$ for $t^{E_{\mathcal{A}}}$. If $e_1^{\mathcal{A}} \sim_{\mathcal{B}^{\mathcal{A}}} e_2^{\mathcal{A}}$ and $e_1, e_2 \in \mathcal{A}$ then $e_1 \sim_{\mathcal{A}} e_2$*
5. *Let $D \in \mathtt{D}$, $a_1, a_2 \in (\Gamma, E, \Delta \vdash D)_{\mathcal{A}}$. Then:*
   $a_1 \sim a_2 \Leftrightarrow$ for all $\mathcal{A} \in \mathtt{Mod}$: $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$
6. *Let $A \in \mathtt{Tp}$, $a_1, a_2 \in (\Gamma, E, \Delta \vdash A)_{\mathcal{A}}$, $b_1, b_2 \in (\Gamma, \Delta \vdash A)_{\mathcal{A}}$. Then:*
   (a) *$a_1 \sim a_2 \Leftrightarrow$ for all $\mathcal{A} \in \mathtt{Mod}$: $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$.*
   (b) *$b_1 \sim b_2 \Leftrightarrow b_1^{i,E} \sim b_2^{i,E}$.*

Proof 1. Assume $t \rightsquigarrow^n u$ and u is normal. By Lemma 3.3.1, the reduction path from $t^{\mathcal{A}}$ has length $m = n$ and terminates in $u^{\mathcal{A}}$ which is normal.

Assume $t^X \rightsquigarrow^m v$ and v is normal. By Lemma 3.3.1, the longest reduction path from t has length $n \leq m$, therefore $t \rightsquigarrow^n u$ and u is normal, hence $t^{\mathcal{A}} \rightsquigarrow^n u^{\mathcal{A}}$ and $u^{\mathcal{A}}$ is normal.

2. If t or $t^{\mathcal{A}}$ normalize, then both normalize by point 1 above. The rest of the thesis follows by definition of $(.)^{\mathcal{A}}$.

3. Assume $d_1, d_2 \in (\Gamma, E, \Delta \vdash D)_{\mathcal{B}^X}$, $d_1 \sim_{\mathcal{B}^X} d_2$, in order to prove $d_1^X \sim_{\mathcal{B}^X} d_2^X$ in the context Γ, Δ. We argue by induction on the definition of $\sim_{\mathcal{B}^X}$.

   By assumption, for some $c \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$ we have $d_j \rightsquigarrow^* c(f_j)$, hence $d_j^X \rightsquigarrow^* c^X f_j^X$, and for all $e_1, e_2 \in \mathtt{I}_{\mathcal{B}^X}(c)$, $f_1(e_1^{\Gamma,E,\Delta}) \sim_{\mathcal{B}^X} f_2(e_2^{\Gamma,E,\Delta})$. By induction hypothesis, $f_1^X((e_1^{\Gamma,E,\Delta})^X) \sim_{\mathcal{B}^X} f_2^X((e_2^{\Gamma,E,\Delta})^X)$. By Corollary 3.2.4.2 we have $(e_j^{\Gamma,E,\Delta})^X \equiv e_j^{\Gamma,\Delta}$. We deduce $f_1^X(e_1^{\Gamma,\Delta}) \sim_{\mathcal{B}^X} f_2^X(e_2^{\Gamma,\Delta})$, for all $e_1(\mathtt{I}_{\mathcal{B}^X}(c))e_2$, and with more reason for all $e_1(\mathtt{I}_{\mathcal{B}^X}(c^X))e_2$ because $\mathtt{I}_{\mathcal{B}^X}(c^X) \subseteq \mathtt{I}_{\mathcal{B}^X}(c)$. By definition of $\sim_{\mathcal{B}^X}$ we deduce $c^X(f_1^X) \sim_{\mathcal{B}^X} c^X(f_2^X)$, that is, $d_1^X \sim_{\mathcal{B}^X} d_2^X$.

4. Assume $d_1^{\mathcal{A}} \sim_{\mathcal{B}^{\mathcal{A}}} d_2^{\mathcal{A}}$ and $d_1, d_2 \in \mathcal{A}$, in order to prove that $d \sim_{\mathcal{A}}$. We argue by induction on the definition of $\sim_{\mathcal{B}^{\mathcal{A}}}$. For $j = 1, 2$, by assumption $d_j^{\mathcal{A}}$ normalizes, therefore d_j normalizes: for some $c \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$ we have $d_j \rightsquigarrow^* c(f_j)$ and $d_j^{\mathcal{A}} \rightsquigarrow^* c^{\mathcal{A}}(f_j^{\mathcal{A}})$. By assumption we have $f_1^{\mathcal{A}}(e_1^{\Gamma,\Delta}) \sim_{\mathcal{B}^{\mathcal{A}}} f_2^{\mathcal{A}}(e_2^{\Gamma,\Delta})$, for all $(e_1, e_2) \in \mathtt{I}_{\mathcal{B}^{\mathcal{A}}}(c^{\mathcal{A}})$. For all $(e_1, e_2) \in \mathtt{I}_{\mathcal{B}^{\mathcal{A}}}(c^{\mathcal{A}})$, from $(e_j^{\Gamma,E,\Delta})^{\mathcal{A}} \equiv e_j^{\Gamma,\Delta}$ we deduce $f_1^{\mathcal{A}}((e_1^{\Gamma,E,\Delta})^{\mathcal{A}}) \sim_{\mathcal{B}^{\mathcal{A}}} f_2^{\mathcal{A}}((e_2^{\Gamma,E,\Delta})^{\mathcal{A}})$. The same conclusion, with more reason, holds for all $(e_1, e_2) \in \mathtt{I}_{\mathcal{A}}(c) \subseteq \mathtt{I}_{\mathcal{B}^{\mathcal{A}}}(c^{\mathcal{A}})$. By induction hypothesis we deduce $f_1(e_1^{\Gamma,E,\Delta}) \sim_{\mathcal{A}} f_2(e_2^{\Gamma,E,\Delta})$. By definition, this is equivalent to $d_1 \sim_{\mathcal{A}} d_2$.

5. Let $a_1, a_2 : D \in \mathtt{D}$. We have to prove that $a_1 \sim a_2 \Leftrightarrow$ for all $\mathcal{A} \in \mathtt{Mod}$: $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$.

   (a) $\Rightarrow$. Assume $a_1 \sim a_2$, and take any $\mathcal{A}, \mathcal{B} \in \mathtt{Mod}$ s.t. $a_1^{\mathcal{A}}, a_2^{\mathcal{A}} \in \mathcal{B}$. By assumption, $a_1 \sim_{\mathcal{B}^{\mathcal{A}}} a_2$, and by point 3, $a_1^{\mathcal{A}} \sim_{\mathcal{B}^{\mathcal{A}}} a_1^{\mathcal{A}}$. By Anti-monotonicity and $a_1^{\mathcal{A}}, a_2^{\mathcal{A}} \in \mathcal{B} \subseteq \mathcal{B}^{\mathcal{A}}$ we deduce $a_1^{\mathcal{A}} \sim_{\mathcal{B}} a_2^{\mathcal{A}}$. This is true for any any $\mathcal{A}, \mathcal{B} \in \mathtt{Mod}$ such that $a_1^{\mathcal{A}}, a_2^{\mathcal{A}} \in \mathcal{B} \in \mathtt{Mod}$, therefore $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$.

   (b) $\Leftarrow$. Assume $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$ for all $\mathcal{A} \in \mathtt{Mod}$, and take any $\mathcal{A} \in \mathtt{Mod}$ s.t. $a_1, a_2 \in \mathcal{A}$. Then $a_1^{\mathcal{A}} \in \mathcal{A}^{\mathcal{A}} a_2^{\mathcal{A}}$, and by $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$ we have $a_1^{\mathcal{A}} \sim_{\mathcal{A}^{\mathcal{A}}} a_2^{\mathcal{A}}$. By point 4 and $a_1, a_2 \in \mathcal{A}$ we deduce $a_1 \sim_{\mathcal{A}} a_2$. This is true for all $a_1, a_2 \in \mathcal{A} \in \mathtt{Mod}$, therefore $a_1 \sim a_2$.

6. We argue by induction on $A \in \mathtt{Tp}$. We first prove that for all $a_1, a_2 : A$: $a_1 \sim a_2 \Leftrightarrow$ for all $\mathcal{A} \in \mathtt{Mod}$: $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$.

   (a) Assume $A \in \mathtt{D}$. The thesis is point 5 above.

   (b) Assume $A \equiv B \times C$. $a_1 \sim a_2 \Leftrightarrow$ for $i = 1, 2$ $\pi_i(a_1) \sim \pi_i(a_2) \Leftrightarrow$ (by induction hypothesis) for all $i = 1, 2$, $\mathcal{A} \in \mathtt{Mod}$: $\pi_i(a_1)^{\mathcal{A}} \sim \pi_i(a_2)^{\mathcal{A}}$ $\Leftrightarrow$ (by $\pi_i(a_j)^{\mathcal{A}} \equiv \pi_i(a_j^{\mathcal{A}})$) for all $i = 1, 2$, $\mathcal{A} \in \mathtt{Mod}$: $\pi_i(a_j^{\mathcal{A}}) \sim \pi_i(a_j^{\mathcal{A}})$ $\Leftrightarrow$ for all $\mathcal{A} \in \mathtt{Mod}$: $a_1^{\mathcal{A}} \sim a_2^{\mathcal{A}}$.

(c) Assume $A \equiv B \to C$.

$\Leftarrow$. Assume $f_1 \sim f_2 : B \to C$ in Γ, E, Δ, in order to prove that $f_1^{\mathcal{A}} \sim f_2^{\mathcal{A}}$ in Γ, Δ. For all $b_1 \sim b_2 : B$ in Γ, Δ, by induction hypothesis on B we have that $b_1^{i,E} \sim b_2^{i,E}$ in Γ, E, Δ. Thus, $f_1(b_1^{i,E}) \sim f_2(b_2^{i,E}) : C$ in Γ, E, Δ, and by induction hypothesis on C, for all $\mathcal{A} \in \mathtt{Mod}$: $f_1(b_1^{i,E})^{\mathcal{A}} \sim f_2(b_2^{i,E})^{\mathcal{A}}$ in Γ, Δ. For $j = 1, 2$, we have $f_j(b_j^{i,E})^{\mathcal{A}} \equiv f_j^{\mathcal{A}}((b_j^{i,E})^{\mathcal{A}}) \equiv$ (by Lemma 3.2.3) $f_j^{\mathcal{A}}(b_j)$. Thus, $f_1^{\mathcal{A}}(b_1) \sim f_2^{\mathcal{A}}(b_2)$ in Γ, Δ for all $b_1 \sim b_2 : B$ in Γ, Δ. We conclude that for all $\mathcal{A} \in \mathtt{Mod}$: $f_1^{\mathcal{A}} \sim f_2^{\mathcal{A}}$.

$\Rightarrow$. Assume for all $\mathcal{A} \in \mathtt{Mod}$: $f_1^{\mathcal{A}} \sim f_2^{\mathcal{A}} : B \to C$ in Γ, Δ, in order to prove that $f_1 \sim f_2$ in Γ, E, Δ. We have to prove that for all $b_1, b_2 : B$, $b_1 \sim b_2$ in Γ, E, Δ, $f_1(b_1) \sim f_2(b_2)$ in Γ, E, Δ. By induction hypothesis on B, for all $\mathcal{A} \in \mathtt{Mod}$: $b_1^{\mathcal{A}} \sim b_2^{\mathcal{A}}$. By assumption, $f_1^{\mathcal{A}}(b_1^{\mathcal{A}}) \sim f_2^{\mathcal{A}}(b_2^{\mathcal{A}})$ in Γ, Δ, and by $f_j(b_j)^{\mathcal{A}} \equiv f_j^{\mathcal{A}}(b_j^{\mathcal{A}})$, we deduce that $f_1(b_1)^{\mathcal{A}} \sim f_2(b_2)^{\mathcal{A}}$ in Γ, Δ, for all $\mathcal{A} \in \mathtt{Mod}$. By induction hypothesis on C, $f_1(b_1) \sim f_2(b_2)$ in Γ, E, Δ. We conclude that $f_1 \sim f_2$.

We now prove that $b_1^{i,E} \sim b_2^{i,E} : A \Leftrightarrow b_1 \sim b_2 : A$. By what we just proved for $(.)^{\mathcal{A}}$ and A, $b_1^{i,E} \sim b_1^{i,E}$ if and only for all $\mathcal{A} \in \mathtt{Mod}$: $(b_1^{i,E})^{\mathcal{A}} \sim (b_2^{i,E})^{\mathcal{A}}$. By Lemma 3.2.3: $(b_j^{i,E})^{\mathcal{A}} \equiv b_j$ we conclude our thesis.

4.3. Existence of a model of Extendable Recursion

Lemma 4.3.1 (Lifting Lemma) *Assume* $E, D \in \mathtt{D}$, $f_1, f_2 : E \to D$, $c \equiv \mathtt{cons}, \mathtt{f}$ *and* $c(f_1) \sim c(f_2) : D$, *in the context* Γ.

1. *If* $c(f_1) \sim c(f_2)$ *then* $f_1 \sim f_2$.
2. *If* $f_1 \sim f_2$ *then* $c(f_1) \sim c(f_2)$.

Proof of Lifting Lemma 1. Assume $c(f_1) \sim c(f_2)$, in order to prove that $f_1 \sim f_2$. By Lemma **??**.6, it is enough to prove that $f_1^{-\Gamma} \sim f_2^{-\Gamma}$ in the context $\mathtt{nil}$. Assume that $\mathcal{A} \in \mathtt{Mod}$, $f_1, f_2 \in \mathcal{A}$, $(e_1, e_2) \in \mathtt{I}_{\mathcal{A}}(c)$ in the context $\mathtt{nil}$, in order to prove that $f_1^{-\Gamma}(e_1) \sim_{\mathcal{A}} f_2^{-\Gamma}(e_2)$. For $j = 1, 2$ we have: $f_j^{-\Gamma}(e_j) \equiv$ (by Corollary 3.2.4.1) $f_j^{-\Gamma}((e_j^{\Gamma})^{-\Gamma}) \equiv f_j(e_j^{\Gamma})^{-\Gamma}$. Thus, by Lemma **??**.6, it is enough to prove that $f_1(e_1^{\Gamma}) \sim_{\mathcal{A}} f_2(e_2^{\Gamma})$, for all $(e_1, e_2) \in \mathtt{I}_{\mathcal{A}}(c)$. This is true by definition of $c(f_1) \sim c(f_2)$.

2. Assume that $f_1 \sim f_2$ in order to prove that $c(f_1) \sim c(f_2)$. By Lemma Pruning.6, for all $(e_1, e_2) \in \mathtt{I}(c) = E_{\mathcal{A}} \cap \sim$ we have $e_1^{\Gamma} \sim e_1^{\Gamma}$. By $f_1 \sim f_2$ we deduce that $f_1(e_1^{\Gamma}) \sim f_2(e_2^{\Gamma})$. Thus, by definition of $\sim$ we conclude that $c(f_1) \sim c(f_2)$.

Lemma 4.3.2 (F Lemma) $\sim$ *is compatible with* $\mathtt{F} \equiv \mathtt{F}_{i,E,A}$.

Proof Assume $u_1, u_2 \in (\Gamma, E, \Delta \vdash A)_{\mathcal{A}}$ and $u_1 \sim u_2$ in order to prove that $\mathtt{F}.u_1 \sim \mathtt{F}.u_2$. We argue by induction on the definition of $u_1 \sim u_2$.

1. Assume $A \equiv D \in \mathtt{D}$. There are some $c \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X$, $f_1 \sim f_2$ such that for $j = 1, 2$ we have $u_j \rightsquigarrow^* c(f_j)$. There is some c' such that, for $j = 1, 2$: $\mathtt{I}_{\mathcal{A}}(c') = \mathtt{I}_{\mathcal{A}}(c)$ and $\mathtt{F}.u_j \rightsquigarrow c'(\mathtt{F}.f_j)$. Indeed, if $\mathtt{F}$ changes c, then $\mathtt{F}$ changes the index of a constant $\mathtt{f}$ or changes a constant future to $\mathtt{cons}$, but in both cases $\mathtt{F}$ preserves type and therefore the index set in $\mathcal{A}$. By definition of $\sim$, we have to prove that for all $(e_1, e_2) \in \mathtt{I}_{\mathcal{A}}(c') = \mathtt{I}_{\mathcal{A}}(c)$ we have $(\mathtt{F}.f_1)(e_1^{\Gamma,\Delta}) \sim_{\mathcal{A}} (\mathtt{F}.f_2)(e_2^{\Gamma,\Delta})$. By Reduction rule, this is equivalent to prove that $\mathtt{F}.f_1((e_1^{\Gamma,\Delta})^{i,E}) \sim_{\mathcal{A}} \mathtt{F}.f_2((e_2^{\Gamma,\Delta})^{i,E})$.

   By Corollary 3.2.4.2, for $j = 1, 2$ we have $(e_j^{\Gamma,\Delta})^{i,E} \equiv e_j^{\Gamma,E,\Delta}$. Thus, we have to prove that $\mathtt{F}.f_1(e_1^{\Gamma,E,\Delta}) \sim_{\mathcal{A}} \mathtt{F}.f_2(e_2^{\Gamma,E,\Delta})$. This follows from $f_1(e_1^{\Gamma,E,\Delta}) \sim f_2(e_2^{\Gamma,E,\Delta})$ and induction hypothesis.

2. By definition of $\sim$, have to prove that for all $i = 1, 2$ we have $\pi_i(\mathtt{F}.u_1) \sim_{\mathcal{A}} \pi_i(\mathtt{F}.u_2)$ for all $\mathcal{A} \ni \mathtt{F}.u$. By Reduction rule it is equivalent to prove that $\mathtt{F}.\pi_i(u_1) \sim_{\mathcal{A}} \mathtt{F}.\pi_i(u_2)$. By the assumption $u_1 \sim u_2$, we have $\pi_i(u_1) \sim \pi_i(u_1)$. The thesis follows by induction hypothesis.

3. Assume $A \equiv B \to C$. By definition we have to prove that for all $a_1, a_2 \in (\Gamma, \Delta \vdash B)_{\mathcal{A}}$, $a_1 \sim a_2$ we have $(\mathtt{F}.u_1)(a_1) \sim_{\mathcal{A}} (\mathtt{F}.u_2)(a_2)$. By Reduction rule, it is enough to prove that $\mathtt{F}.u_1(a^{i,E}) \sim_{\mathcal{A}} \mathtt{F}.u_2(a_2^{i,E})$. By $a_1 \sim a_2$ and Lemma **??**, point 6, we have $a_1^{i,E} \sim a_2^{i,E}$. By the assumption $u_1 \sim u_2$ we deduce $u_1(a_1^{i,E}) \sim u_2(a_2^{i,E})$, then $\mathtt{F}.u_1(a_1^{i,E}) \sim_{\mathcal{A}} \mathtt{F}.u_2(a_2^{i,E})$ by induction hypothesis.

Theorem 4.3.3 $\sim$ *is an equivalence relation.*

Proof By induction on the definition of the term.

1. Assume $c \equiv C, \pi_i, <\_,\_>$ for some combinator C. We use the Reduction rule.

2. Assume $c \equiv \mathtt{r}$. We prove that for all $\mathcal{A} \ni \vec{r_1}, d_1, \mathtt{r}(\vec{r_2}), d_2$ with $\vec{r_1}, d_1 \sim \vec{r_2}, d_2$ we have $\mathtt{r}(\vec{r_1})(d_1) \sim_{\mathcal{A}} \mathtt{r}(\vec{r_2})(d_2)$. We argue by induction on the definition of $d_1 \sim_{\mathcal{A}} d_2$. By $d_1 \sim_{\mathcal{A}} d_2$ we have $d_j \rightsquigarrow^* c(f_j)$ for $j = 1, 2$. We need Lemma 4.3.1.2 in order prove that if $c \equiv \mathtt{cons}, \mathtt{f}$ and $c(f_1) \sim c(f_2)$ then $f_1 \sim f_2$, then closure under application, and eventually Reduction rule. We conclude $\mathtt{r}(\vec{r_1})(d_1) \sim \mathtt{r}(\vec{r_2})(d_2)$.

3. Assume $c \equiv \mathtt{cons}, \mathtt{f}, \mathtt{j}_X : (E \to A) \to A$. We prove that c is total by induction on A.

   (a) Assume $A \equiv D \in \mathtt{D}$. If $c \equiv \mathtt{cons}, \mathtt{f}$, we need Lemma 4.3.1.1 in order prove that if $f_1 \sim f_2$ then $c(f_1) \sim_{\mathcal{A}} c(f_2)$ for all $\mathcal{A} \in \mathtt{Mod}$. If $c \equiv \mathtt{j}_X$ we use definition unfolding.

   (b) Assume $A = B \times C$ and $c \equiv \mathtt{f}, \mathtt{j}_X$. If $f_1 \sim f_2$ then for $i = 1, 2$: $\pi_i \circ f_1 \sim_{\mathcal{A}} \pi_i \circ f_2$. By induction hypothesis on B, C, for all $\mathcal{A} \in \mathtt{Mod}$, $i = 1, 2$ we deduce $c(\pi_i \circ f_1) \sim_{\mathcal{A}} c(\pi_i \circ f_2)$. By Reduction rule and $\pi_i c(f) \rightsquigarrow c(\pi_i \circ f)$ we obtain $\pi_i c(f) \sim_{\mathcal{A}}$. We conclude that $c(f) \sim_{\mathcal{A}}$.

   (c) Assume $A = B \to C$ and $c \equiv \mathtt{f}, \mathtt{j}_X$. Let $P(b, g) = g(b)$. If $f_1 \sim f_2$ and $a_1 \sim a_2$, then $f_1(a_1) \sim f_2(a_2)$ and for $j = 1, 2$: $P(a_j)(f_j)(b_j) \rightsquigarrow^* f_j(a_j)(b_j) : C$ for all $b_1 \sim b_2$. By Reduction rule we deduce $P(a_1) \circ f_1 \sim P(a_2) \circ f_2$. By induction hypothesis on C we derive $c(P(a_1) \circ f_1) \sim_{\mathcal{A}} c(P(a_2) \circ f_2)$. For $j = 1, 2$ we have $c(f_j)(a_j) \rightsquigarrow c(P(a_j) \circ f_j)$. By Reduction rule we get $c(f_1)(a_1) \sim_{\mathcal{A}} c(f_2)(a_2)$ We conclude that $c(f_1) \sim_{\mathcal{A}} c(f_2)$.

4. By Lemma 4.3.2, if $u_1 \in (\Gamma, E, \Delta \vdash A)_{\mathcal{A}}$ and $u_1 \sim u_1$ then $\mathtt{F}.u_1 \sim \mathtt{F}.u_1$.

5. Total terms are closed under applications: by definition if $f \sim f$ and $a \sim a$ then $f(a) \sim f(a)$.

Corollary 4.3.4 (Normalization for $\mathcal{N}$) *1. For any data type $D \in \mathtt{D}$, any term $d : D$ of $\mathcal{N} + \mathcal{I}$ normalizes and denotes a well-founded tree.*

2. The quotient $(\mathcal{N}+\mathcal{I})/\sim$ is an Extensional Model of simply typed lambda calculus with primitive recursion in all types.

Proof 1. By Theorem **??**, the term d is total: we have $d \sim d$. By definition of $\sim$, d has a normal form and denotes a well-founded tree.

2. Again by Theorem **??**, the relation $\sim$ is an equivalence relation. By definition, $\sim$ is compatible with application. By Lemma 4.3.2, if $u_1 \sim u_2$ then $\mathtt{F}.u_1 \sim \mathtt{F}.u_2$. By definition, and the fact that it is an equivalence relation, $\sim$ is extensional: $\sim$ equates two terms $d, e : D$ in the context $\mathtt{nil}$ if and only if they denote the same tree, two maps if they are equal pointwise, and two pairs if they are equal componentwise.

In order to define a model we have to define a term $\mathtt{lift} : (A \to B) \to (A@E \to B@E)$ and a restriction map $(.\lceil X) : A@E \to A$ for any $X \subseteq |E|$.

We first define a map $\mathtt{B}_A : A@D \to A$ in the context $\Gamma = \{E\}$, by induction on A. Assume $D = (D_0, \ldots, D_{n-1})$. We define $\mathtt{B}_D = \mathtt{r}(\vec{R})$, with

$\vec{R} = R_0, \ldots, R_{n+1}$, with $R_i = \mathtt{cons}_i$ for $i < n$ and $R_n = \mathtt{f}_{0,E}$ and $R_{n+1} = \mathtt{id}_D$. By definition unfolding, $\mathtt{B}_D$ hereditarily replaces the constructor $\mathtt{cons}_n$ with $\mathtt{f}_{0,E}$. We extend $\mathtt{B}$ componentwise and pointwise. If C is the combinator defined by $C(f_1, f_2)(a) = \langle f(\pi_1(a)), f(\pi_2(a))\rangle$ we set $\mathtt{B}_{A\times B} = C(\mathtt{B}_A, \mathtt{B}_B)$: by definition unfolding we have $\pi_i(\mathtt{B}_{A_1\times A_2})(c) = \mathtt{B}_{A_i}(\pi_i(c))$. If C is the combinator which does composition, we set $\mathtt{B}_{A\to B} = C(\mathtt{B}_B)$: by definition unfolding we have $\mathtt{B}_{A\to B}(f)(a) = \mathtt{B}_B(f(a))$.

We define the map $(.\lceil X) : A@D \to A$ in the context $\mathtt{nil}$ in the same way, except that we set $R_n = \mathtt{j}_X$.

We define $\mathtt{lift}$. Let $C(x, y, z) = y(x(z))$ be reverse composition. We set $\mathtt{lift}_{A,B} = \mathtt{F}_{0,E}(C(\mathtt{B}_A))$, in the context $\mathtt{nil}$, because $C(\mathtt{B}_A)$ is in the context $\{E\}$. By definition unfolding, we have $\mathtt{lift}(f)(a) = \mathtt{F}_{0,E}(C(\mathtt{B})(f^{0,E})(a^{0,E})) = \mathtt{F}_{0,E}(f^{0,E}(\mathtt{B}(a^{0,E})))$, in the context $\mathtt{nil}$.

Theorem 4.3.5 (Extendable Recursion) *The quotient $(\mathcal{N} + \mathcal{I})/\sim$ is a model of Extendable Recursion.*

Proof We check that we have $\mathtt{lift}$ and $(.\lceil X)$ satisfying the required equations.

1. First, we remark that $\mathtt{B}(v^{0,E})^{-0,E,X} = \mathtt{B}^{-0,E,X}((v^{0,E})^{-0,E,X}) = \lceil_X(v)$.
2. Second, we prove that if $v = \mathtt{F}_{0,E}.(u)$, then $\mathtt{B}(v^{0,E})^{-0,E,X} = u^{-0,E,X}$. Assume the expression has type $A \in \mathtt{Tp}$. If $A \in \mathtt{D}$, the proof is by induction on the tree denotesd by u, otherwise the proof is by induction on A.

Using the two points above, we deduce that $F^*(t)\lceil X = \mathtt{lift}(F)(t)\lceil X = \mathtt{F}(F^{0,E}(\mathtt{B}(t^{0,E})))\lceil_X$. Let $u = F^{0,E}(\mathtt{B}(t^{0,E}))$ and $v = \mathtt{F}(u)$: then $F^*(t)\lceil X = v\lceil_X =$ (by point 1 above) $\mathtt{B}(v^{0,E})^{-0,E,X} =$ (by point 2) $u^{-0,E,X} = F^{0,E}(\mathtt{B}(t^{0,E}))^{-0,E,X} =$ (by $(t^{0,E})^{-0,E,X} = t$) $F(\mathtt{B}^{-0,E,X}(t)) = F(\lceil_X(t))$. Thus, we proved that $F^*(t)\lceil X = F(\lceil_X(t))$, as wished.

Chapter 5

Discussing the definition of $\mathcal{N}$

5.1. Examples of Data Types in $\mathcal{N}$ Here are some examples of arities.

1. Assume D_i is empty. Then c has arity 0, f is a dummy map with domain the empty set and cf is a one-node tree (whose root has no children).

2. Assume $D_i = \{e_0, \ldots, e_{k-1}\}$ has $k \in \mathtt{N}$ elements. Then $f : D_i \to D$ is any enumeration $f(e_0), \ldots, f(e_{k-1})$ of k elements of D and cf is a tree whose root has n children. The child number j is $f(e_j)$, with index e_j.

3. Assume D_i represents the set $\mathtt{N}$ natural numbers. Then cf is is a tree whose root has countably many of children, indexed on $n : \mathtt{N}$.

Here are some examples of data types $D \in \mathtt{D}$.

1. The data type $\perp\!\!\!\perp = ()$ has 0 constructors and denotes $\emptyset$

2. The data type $\mathtt{Unit} = (\perp\!\!\!\perp)$ has 1 constructor of arity 0, and denote the singleton set

3. The data types $\mathtt{Bool} = (\perp\!\!\!\perp, \perp\!\!\!\perp)$ has 2 constructors of arity 0, and denote the two-elements set

4. The data type with constructors of arity $0, 1$ is (isomorphic to) $\mathtt{N}$.

5. The data type with constructors of arity $0, 1, 2$ is (isomorphic to) the set Ω_0 of finite binary trees.

6. The data type with constructors of arity $0, 1, 2, \omega$ denotes the set Ω_1 of well-founded trees whose nodes have 0 or 1 or 2 or ω children.

7. We may nest the definition of data types: each data type may be the index set of a constructor. For instance, we may form the data types Ω_2 of well-founded trees whose nodes have 0 or 1 or 2 or ω children, or children indexed over the set Ω_1.

5.2. A circularity problem To define the elements of a data type D in a context Γ poses the following *circularity problem*. The elements of D may be defined out of some future constructor $\mathtt{f} : (E \to D) \to D$: but E may be D itself, which is not yet defined. This is a problem in proving that computations in $\mathcal{N}$ terminate and all denotable trees are well-founded. We may explain this loop by saying that, after defining D and a map $f : D \to D$ by recursion, we may come back to a time before the definition of D, and use a future constructor of index set D to define some new element $e \in D$ which was not there when we defined D and f. We would like to prove that the computation of $f(e)$ terminate, and that $f(e) : D$ is a well-founded tree, but we have no evidence for these fact. We break this circularity by providing D with approximation $\mathtt{j}$ of future constructors, in such a way that every future constructor $\mathtt{f}$ in some model $\mathcal{A}$ of $\mathcal{N}$, say the one defining e, is equal to some approximation $\mathtt{j}$ in some model $\mathcal{B} \supset \mathcal{A}$ of $\mathcal{N}$. Expanding the model of $\mathcal{N}$ from $\mathcal{A}$ to $\mathcal{B}$ provides the required evidences for the computation of $f(e)$, because the tree denoted by e is isomorphic to the tree denoted by some element $e^{\mathcal{A}}$ which (in $\mathcal{B}$, at least) was in D in the moment we defined D and f. $f(e)$ is a terminating computation and denotes a well-founded tree because this is the case for $f(e^{\mathcal{A}})$. The expansion from $\mathcal{A}$ to $\mathcal{B}$ may be seen as a second temporal coordinate, recording the modification producing by coming back to a time before the definition of D. Events in this second temporal coordinate are new evidences for termination, and they produce no observable effect to the computation itself of a term. Thus, no time-travel inconsistencies arises: the choice of coming back before the definition of D is not modified by the adding $\mathtt{j}$ to the model $\mathcal{A}$ and forming the model $\mathcal{B}$ containing $e^{\mathcal{A}}$. The computation history of $f(e)$ in $\mathcal{A}$ and in the modified model $\mathcal{B}$ are the same. The loop in time is only a way of proving termination of a computation and does not affect the computation itself.

This feature of $\mathcal{N}$ is similar to the literary device called *retroactive continuity*, in which established facts in a fictional work are adjusted by a subsequently published work. In our case, however, inconsistencies cannot arise, because we only add new correct knowledge, instead of arising and being ignored.

Approximation terms $\mathtt{j}$ are only used in the semantics and for termination and well-foundedness proofs, and they do not arise in computations. We may explain better why this is the case by informally describing how we compute

with future constructors in $\mathcal{N}$. A constructor $c : (D_i \to D) \to D$ belongs to one specific data type D, while a future constructor may be added to any data type E, and may be extended componentwise/pairwise to any type A. If $E \in \mathtt{D}$, $j < m$ and $f : E_j \to E$, a term $\mathtt{f}_j f : E$ denotes a tree whose root has children all fe indexed on $e : E_j$. All indexes e are supposed to be "not known yet", which is consistent with our idea of future constructor. The only way to act on $\mathtt{f}_j f$ is to act in the same way on all children fe, that is, to compose the map f with some map $g : E \to B$, obtaining some tree $\mathtt{f}_j(g \circ f) : B$ of type B. We included in $\mathcal{N}$ a constant $\mathtt{u}$ which acts on $\mathtt{f}_j f$ in this way.

5.3. The role of constants $\mathtt{j}_X$ The difference between the tree $\mathtt{j}_{X,E,A} f$ and the tree $\mathtt{f}_{i,E,A} f$ is that the index set of $\mathtt{j}_{X,E,A} f$, necessarily, is only some subset of the index set of $\mathtt{f}_{i,E,A} f$. In any model of $\mathcal{N}$, not all elements of E may be listed in $\{t_i | i \in I\}$, because some elements of E may be defined from $\mathtt{j}_{X,E,A} f$ itself, therefore they do not exist yet when $\mathtt{j}_{X,E,A} f$ is defined. This is the same circularity problem we outlines in §**??**. As we anticipated, the problem may be solved as follows. The elements of E in any model $\mathcal{A}$ of $\mathcal{N}$ may be represented by some list $\{t_i | i \in I\}$ in some other model $\mathcal{B} \supseteq \mathcal{A}$. For instance, from the fact that a map terminates on all approximations of future extensions and all models we may infer the the fact that the same map terminates on all future extensions and all models. Adding approximations to a model $\mathcal{A}$ increases our knowledge about existing terms, without changing the behaviour of the terms we already have.

We have less operations on approximations than on future constructors. Approximations and future constructors and are treated in the same way in a recursive definition, but a operator $\mathtt{F}$ may modify a future constructor, while it cannot modify an approximation.

Therefore adding approximation constants is of no use in computation. Instead, approximations are experiments intended to show that a map terminates and produce well-founded trees when applied on future constructors. Approximations are only intended to represent, within the current type A and in some future model, some data type E which is possibly defined from A. In the current model, a joint just "approximates" the tree defined by a future constructor. We first prove that a map behaves correctly on all models and all extensions of its domain with joints, but without future constructors, even if in each model joints are smaller than the trees defined by future constructors. In this way we will deduce that the map behaves correctly when its domain include all possible future constructors.

5.4. About our notion of Totality and of Partial Equivalence Relation Our notion of totality modifies the notion of Tait, and our notion of Partial Equivalence Relation Model modifies the one introduced by Longo-Moggi. We discuss our changes in the case of the definition of Totality (the same remarks apply to Partial Equivalence Relation). We define $f \in \mathtt{T}_{\mathcal{A}}$ (f is total in the model $\mathcal{A}$) as $f(a) \in \mathtt{T}_{\mathcal{A}}$ for all $a \in \mathtt{T}$, and not for all $a \in \mathtt{T}_{\mathcal{A}}$ as Tait did.

We explain what is the reason. We allow the map f to be applied to arguments with fresh future constructors. As we already remarked, we cannot require that the map is total on all arguments with fresh future constructors without producing a *circularity*. Instead, we guarantee the correct behavior of f on arguments with future constructors by representing future constructors in a model $\mathcal{A}$ by approximated constructors in some larger model $\mathcal{B} \supseteq \mathcal{A}$. The approximated constructor which we introduce in the definition of f depends on the argument we feed to f. Adding one more approximated constructor to definition if we need it afterwards would be a form of *retro-continuity*, this is why we assume in the definition of f that f is total on all possible approximated constructors in all models.

However, notice that we do not by-pass retro-continuity completely. Indeed, a quantification on all models is impredicative, we also quantify on models which are not defined yet. In a sense, in an impredicative definition we add new sets whenever we need them afterward. Thus, the impredicative definition itself could be seen as a form of retro-continuity. We cannot bypass retro-continuity completely, we may only isolate the part of it which is used to deduce the results we want to deduce, termination and that fact that all trees we define are well-founded.

5.5. Comparing the models of $\mathcal{N}$ and Kripke models Typically, a property P is monotonic in a family $\mathtt{Mod}$ of Kripke models: for every $\mathcal{A}$ we have an interpretation $P_{\mathcal{A}} \subseteq \mathcal{A}$, and if $\mathcal{A} \subseteq \mathcal{B}$ then $P_{\mathcal{A}} \subseteq P_{\mathcal{B}} \cap \mathcal{A}$. This means that when a model $\mathcal{A}$ increases to $\mathcal{B}$ the properties of individuals becomes weaker. In any Kripke model $\mathcal{A} \in \mathtt{Mod}$ we define $f \in (P \to Q)_{\mathcal{A}}$ if and only if $\forall a.(\forall \mathcal{B} \supseteq \mathcal{A}.a \in P_{\mathcal{B}} \to f(a) \in Q_{\mathcal{B}})$.

In our definition of total terms, we make the opposite request: the property $P = \mathtt{T}$ is anti-monotonic. For every $\mathcal{A}$ we have an interpretation $\mathtt{T}_{\mathcal{A}} \subseteq \mathcal{A}$, and if $\mathcal{A} \subseteq \mathcal{B}$ then $\mathtt{T}_{\mathcal{A}} \supseteq \mathtt{T}_{\mathcal{B}} \cap \mathcal{A}$ (actually, we did not prove this property, but the corresponding property for $\sim$). This means that when a model $\mathcal{A}$ increases to $\mathcal{B}$ the properties of individuals becomes stronger. In any our model $\mathcal{A} \in \mathtt{Mod}$ we define $f \in (P \to Q)_{\mathcal{A}}$ if and only if $\forall a.(\forall \mathcal{B} \supseteq \mathcal{A}.a \in P_{\mathcal{B}}) \to f(a) \in Q_{\mathcal{A}}$. The the scope of the quantification $\forall \mathcal{B} \supseteq \mathcal{A} \ldots$ is smaller

in our definition than in Kripke's.

To be accurate, in our definition of totality we use a statement apparently weaker than the one we wrote above: we asked $f \in (P \to Q)_{\mathcal{A}}$ if and only if $\forall a.(\forall \mathcal{B} \ni a.a \in P_{\mathcal{B}}) \to f(a) \in Q_{\mathcal{A}}$. However, if we assume anti-monotonicity the two definitions are equivalent. Indeed, assume $(\forall \mathcal{B} \supseteq \mathcal{A}.a \in P_{\mathcal{B}})$. Assume $a \in \mathcal{C}$ in order to prove $a \in P_{\mathcal{C}}$. Let $\mathcal{B}$ be the smallest model including $\mathcal{A} \cup \mathcal{C}$. Then $\mathcal{B} \supseteq \mathcal{A}$, hence $a \in P_{\mathcal{B}}$, and by anti-monotonicity and $a \in \mathcal{C}$ we conclude $a \in P_{\mathcal{B}} \cap \mathcal{C} \subseteq P_{\mathcal{C}}$, as wished.