On Extensions of AF2 with Monotone and Clausular (Co)inductive Definitions

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften an der Fakultät für
Mathematik, Informatik und Statistik der
Ludwig-Maximilians-Universität München

vorgelegt von

Favio Ezequiel Miranda Perea
aus Mexiko-Stadt

im September 2004

1. Berichterstatter: Prof. Dr. Helmut Schwichtenberg
2. Berichterstatter: Prof. Dr. Wilfried Buchholz
Tag des Rigorosums: 12. November 2004

Contents

Abstract vii

Zusammenfassung ix

Acknowledgements xi

Agradecimientos xiii

Introduction xv

1 Preliminaries 1

1.1 Categorical Interlude 1

1.1.1 M-(Co)algebras 5

1.1.2 Dialgebras 9

1.2 The Type System F 12

1.2.1 Adding Sum and Product Types 14

1.2.2 Adding Existential Types 22

1.2.3 On Embeddings 23

1.3 Second Order Logic AF2 24

1.3.1 Definition of the System 24

1.3.2 Strong Normalization of AF2 29

1.3.3 Adding Conjunctions and Disjunctions 29

2 Extensions of System F with Monotone (Co)inductive Types 31

2.1 From Categories to Types 31

2.1.1 Representing (Co)algebras 32

2.1.2 Representing Dialgebras 35

2.2 The System MICT 37

2.2.1 Definition of the System 37

2.2.2 Strong Normalization of MICT 39

2.3 The System MCICT 52

2.3.1 Definition of the System 52

2.3.2 Strong Normalization of MCICT 57

2.3.3 On η-rules 62

2.3.4 Canonical Monotonicity Witnesses 65

2.3.5 (Co)recursive Programming in MCICT 68

2.4 The System MCICT_M 75

2.4.1 Definition of the System 75
2.4.2 Strong Normalization of MCICT_M 76
2.5 The Hybrid System $\mathsf{MCICT}_{\mu M\nu}$ 77

3 Monotone and Clausular (Co)inductive Definitions 79
3.1 Fixed-Point Theory 79
3.2 The Logic MCICD 80
3.3 Strong Normalization of MCICD 86
3.4 Canonical Monotonicity Witnesses 87

4 Realizability for MCICD 93
4.1 The Logic $\mathsf{MCICD}^\star$ 93
4.1.1 Definition of the Logic 93
4.1.2 Strong Normalization of $\mathsf{MCICD}^\star$ 95
4.1.3 Subject Reduction for $\mathsf{MCICD}^\star$ 95
4.2 The Realizability Interpretation 104
4.2.1 Realizing the Axioms 106
4.2.2 The Soundness Theorem 120

5 Programming with Proofs 127
5.1 Semantics 127
5.1.1 Syntactical Models for the Term System 127
5.1.2 Semantics for the Logic $\mathsf{MCICD}^\star$ 132
5.2 Formal Data Types 142
5.2.1 A Connection with Modified Realizability 143
5.2.2 The Canonical Model 144
5.2.3 Examples of Data Types 145
5.3 Programming with Proofs in MCICD 150
5.3.1 Programming Functions with Iteration or Recursion . . . 151
5.3.2 Programming Functions with Coiteration or Corecursion . 156

6 A System with Mendler-style Coinduction 159
6.1 Fixed-Point Theory 159
6.2 The Logic $\mathsf{MCICD}_{\mu M\nu}$ 160
6.3 Realizability for $\mathsf{MCICD}_{\mu M\nu}$ 162
6.3.1 Realizing the Axioms 162
6.3.2 The Soundness Theorem 164
6.4 Semantics 165
6.5 Programming with Proofs in $\mathsf{MCICD}_{\mu M\nu}$ 169
6.5.1 Data types with Equality 170
6.5.2 Programming with Mendler-style Coiteration or Corecursion 175

7 Conclusions and Future Work 179
7.1 Conclusions . . . 179
7.2 Related Work . . . 181
7.3 Future Work . . . 182

Bibliography 187

Symbol Index 193

Index 195

Lebenslauf 199

Abstract

This thesis discusses some extensions of second-order logic (AF2) with primitive constructors representing leastand greatest fixed points of monotone operators, which allow to define predicates by induction and coinduction. Though the expressive power of second-order logic has been well-known for a long time and suffices to define (co)inductive predicates by means of its (co)induction principles, it is more user-friendly to have a direct way of defining predicates inductively. Moreover recent applications in computer science oblige to consider also coinductive definitions useful for handling infinite objects, the most prominent example being the data type of streams or infinite lists. Main features of our approach are the use clauses in the (co)inductive definition mechanism, concept which simplifies the syntactic shape of the predicates, as well as the inclusion of not only (co)iteration but also primitive (co)recursion principles and in the case of coinductive definitions an inversion principle. For sake of generality we consider full monotone, and not only positive definitions —after all positivity is only used to ensure monotonicity.

Working towards practical use of our systems we give them realizability interpretations where the systems of realizers are strongly normalizing extensions of the second-order polymorphic lambda calculus, system F, in Curry-style, with (co)inductive types corresponding directly to the logical systems via the Curry-Howard correspondence. Such realizability interpretations are therefore not reductive: the definition of realizability for a (co)inductive definition is again a (co)inductive definition. As main application of realizability we extend the so-called programming-with-proofs paradigm of Krivine and Parigot to our logics, by means of which a correct program of the lambda calculus can be extracted from a proof in the logic.

Zusammenfassung

Diese Dissertation beschäftigt sich mit Erweiterungen der Logik zweiter Stufe (AF2) mit primitiven Konstruktoren, die kleinste und größte Fixpunkte monotoner Operatoren repräsentieren, mit denen Prädikate durch Induktion und Koinduktion lassen sich definieren. Obwohl die Ausdrucksfähigkeiten der zweitstufiger Logik schon seit lange Zeit bekannt sind und reichen um (ko)induktive Prädikate, mittels ihre (ko)induktion Prinzipien zu definieren, es ist freundlicher, eine direkte Weise zu haben, Prädikate induktiv zu definieren. Darüber hinaus fordern letzte Anwendungen in der Informatik koinduktive Definitionen zu betrachten, welche nützlich für die Behandlung unendlicher Objekte sind, das bedeutendste Beispiel sei die Datentyp von Ströme oder unendliche Listen. Hauptbeiträge unsere Behandlung sind der Gebrauch von Klauseln in dem Mechanismus (ko)induktiver Definierung. Konzept, das die syntaktische Form der Prädikate vereinfacht, sowie die Betrachtung nicht nur von (Ko)iteration sondern auch von Prinzipien primitiver (Ko)rekursion. Im Interesse der Allgemeinheit, betrachten wir voll monoton, und nicht nur positive Definitionen, immerhin die syntaktische Beschränkung zu positiven Definitionen ist nur verwendet, um Monotonie sicherzustellen.
In Richtung praktischer Anwendungen unserer Systemen geben wir ihnen Realisierbarkeitsinterpretationen, wobei die Systeme von Realisierern stark normalisierende Erweiterungen des polymorphen Lambda Kalküls zweiter Stufe, System F á la Curry, mit (ko)induktive Typen sind, die direkt den logischen Systemen durch die Curry-Howard Korrespondenz entsprechen. Solche Realisierbarkeitsinterpretationen sind folglich nicht reduktive: die Definition der Realisierbarkeit für eine (ko)induktive Definition ist wieder eine (ko)induktive Definition. Als Hauptanwendung der Realisierbarkeit werde das sogenannte programmieren-mit-Beweise Verfahren von Krivine und Parigot auf unsere Logik erweitert, mit welchem ein korrektes Programm des Lambda-Kalküls aus einem Beweis in der Logik gewonnen werden kann.

Acknowledgements

When the moment came for me to choose where to go to pursuit my doctoral studies my main concern was the fact that up to that moment I had enjoyed a big freedom in my academical life and certainly wanted to keep it. Now after four and a half years and with this piece of work under my arm I can only say that I made the right decision by coming to München. I am very thankful to Prof. Dr. Helmut Schwichtenberg for accepting me as Ph. D. student but mainly for the excellent research environment he has formed at the Mathematics Institute of the Ludwig-Maximilians-Universität in München in the Theresienstraße. Specially for the very famous "Mitarbeiterbesprechung" every thursday where more than once I got inspiring ideas which are part of this work.

Dr. Ralph Matthes deserves a special mention. This work would have never been finished without his help, by explaining to me concepts which now seem a child's game but that on the beginning of my research were so difficult like the correct use of prepositions and declensions in the german language still is, and specially for telling me to start my research with something very easy, even trivial, but something that I could tell to be mine. I am also thankful to him for getting me an office at the chair for Theoretical Computer Science in the Computer Science institute at the Oettingenstraße. Room D.11 has been a very comfortable scientific home during this time and I will certainly miss it.

I thank Prof. Dr. Wilfried Buchholz in München and Prof. Dr. Michel Parigot in Paris for taking the time to reviewing my work.

Being an associated member of the *Graduiertenkolleg Logik in der Informatik (GKLI)* allowed me to attend several summer schools and conferences which improved substantially my spirit of research. The GKLI's colloquium every friday morning provided me with a deep overview of the very different branchs of research on logic and theoretical computer science.

Dr. Olha Shkaravska provided me with an oasis full of jokes and nonsense which took the stress away in many occasions.

Last but not least I dedicate this work to my family and friends in Mexico and München, for supporting me even in the darkest moments when I thought I would never finish this research.

I gratefully acknowledge the funding of the joint program between the Mexican National Council of Science and Technology (CONACYT) and the German Academic Exchange Service (DAAD) by credit grant 154186.

Agradecimientos

Este trabajo no es sólo mio, en cada página y cada símbolo matemático están escondidos muchos momentos, momentos de alegria y tristeza, de calma y desesperación, de incertidumbre y seguridad, los cuales se crearon gracias a muchas personas que me encontré en el camino sin las cuales jamás hubiera llegado al final. Ellos me sacaron del mundo matemático, de diversas maneras, en los momentos más obscuros cuando parecía que jamás podría terminar mi investigación. Agradezco a todos aquellos que estuvieron conmigo en algun momento, y sobre todo a los que siguen ahí, en especial a Aura Mireles por los tragos y los partidos de Scrabble, pero sobre todo por la amistad, A Maria Angelica y Erwin Fellermier por darme no sólo una habitación, sino un hogar en la Dobmannstraße 10 en diversas ocasiones. A Helen Briseño, Jimie, Daniel y Jaime Roura por las inolvidables tertulias en Germering. A Giovanni e Ivonne Barrios Salas por las parrandas y el delicioso sancocho. A Jorge Medina por los partidos de "gana pierde" y las exquisitas cenas los fines de semana en la Auenstraße 104, Al Dr. Jorge Galindo por dejarme hechar un vistazo al mundo de las ciencias sociales, pero sobre todo por las profundas discusiones frente al televisor en la Finauerstraße 4.

En México agradezco a mi familia y amigos quienes, cada vez que volví de vacaciones, me hicieron sentir como si nunca me hubiera ido. En especial dedico este trabajo a mamá quien me enseño los primeros números y a papá quien me enseño lógica y teoría de conjuntos por primera vez, gracias por el infinito amor que he recibido. A Lupilla por todo el amor que me has dado, por enseñarme el mundo del teatro y sobre todo por seguir ahí a pesar de la distancia. A la facultad de ciencias de la UNAM, en especial a mis maestros José Alfredo Amor, Carlos Torres y Elisa Viso por su amistad y apoyo continuo, y a mi alumna más brillante Liliana Reyes por tu sonrisa.

Negrita de mis pesares, ojos de papel volando,
a todos díles que sí, pero no les digas cuando,
así me dijiste a mí, por éso vivo penando!
Son de la negra, Jalisco México

Habich ist ein schöner Vogel, Hättich nur ein Nestling.
Deutsches Sprichwort

Introduction

The Curry-Howard correspondence ([Ho80]) or formulas-as-types paradigm is a powerful tool relating logical systems, handling mathematical proofs, with the world of programs, represented as systems of lambda calculi. It considers specifications of programs as formulas in a given logic and allows to extract programs, written as lambda terms, from proofs of these formulas.

This thesis addresses some extensions of this famous correspondence with (co)inductive types/definitions. First we extend the second-order polymorphic lambda calculus, system F, with (co)inductive types taking as motivation the categorical approach ([JaRu97]): an inductive type represents a (weak) initial algebra of a functor, whereas a coinductive type dually represents a (weak) final coalgebra. Our main extension, inspired by [Mat98] and [Hag87a], includes full-monotone types with a clausular feature. Moreover we define also an extension with Mendler-style co(induction) principles ([Men87, Men91]).

Next we move to the realm of formulas, introducing a concept of monotone and clausular (co)inductive definition/predicate and extending the Curry-Howard correspondence by defining an extension of second-order logic $\mathsf{AF2}$ with primitive constructors for (co)inductive definitions representing least and greatest fixed points of monotone, and not only positive, operators.

Our choice for the system of second-order logic is not accidental. $\mathsf{AF2}$, being a constructive logic, has been proved suitable for extracting programs from proofs. Building on the work of Krivine and Parigot [KrPa90, Par92] we extend their so-called *programming-with-proofs* paradigm to our system of (co)inductive definitions. The importance of such paradigm is that it neccesarily produces programs which do what one expects, not magically, but for mathematical reasons. A cornerstone of the method is the use of realizability ([Tro98]) (called "semantic notion of type" in [KrPa90]), this is an important technique to make explicit the computational content hidden in a proof. If a logic is constructive[1] and has a sound realizability interpretation we can produce a program and its verification proof effectively from a proof of the specification formula using the realizability interpretation.

The *programming-with-proofs* paradigm uses some kind of semantics, in our case a tarskian one, to formulate, in a given model, a concept of formal data type, which is a unary predicate having a special property with respect to realizability, namely the inhabitants of the data type are realizers of its own inhabitation.

[1] Some research has been done also on extracting programs from classical proofs, see for example [BBS02]

This self-realizing property allows to obtain programs without calculate realizers explicitly.

To finish this introduction we give an overview of the contributions and an outline of the contents.

Contributions

As the main contributions of this thesis I consider:

- A formulation of a strongly normalizing type system in Curry-style MCICT, extending system F, with both inductive and coinductive types including (co)iteration, (co)recursion and inversion principles as well as monotone and clausular features.
- The concept of monotone and clausular (co)inductive definition is introduced and added to second-order logic AF2 getting an extension MCICD corresponding to the type-system MCICT under the Curry-Howard correspondence.
- A realizability interpretation for the system of (co)inductive definitions MCICD using as term language the corresponding system of (co)inductive types, i.e. the realizers are terms of MCICT. This interpretation is not reductive, meaning that the realizability of a (co)inductive predicate is again defined (co)inductively.
- As main application of our realizability interpretation the extension of the programming with proofs method to MCICD.
- Formulation of a system of (co)inductive definitions with coinduction principles in Mendler-style together with its realizability interpretation suitable for extracting programs from proofs of specifications including coinductive definitions.

We give more details of the contributions on chapter 7.

Chapter Outline

Chapter one introduces the basic concepts on category theory, lambda calculus and logic needed later, in particular we present basic definitions on (co)algebras and dialgebras, the definition of system F including a direct strong normalization proof which will be extended later to the basic system with (co)inductive types, and the basics about the second-order logic AF2.
Chapter two is devoted to type systems, in the spirit of [Mat98, Mat99] we present two extensions of system F with monotone (co)inductive types: the first one, called MICT, includes traditional (co)inductive types of the form $\mu\alpha\rho, \nu\alpha\rho$.

To prove the strong normalization of this system we proceed directly extending the proof for F via saturated sets and the SN-method. The second system, called MCICT extends F with monotone (co)inductive types of the form $\mu\alpha(\rho_1, \ldots, \rho_k), \nu\alpha(\rho_1, \ldots, \rho_k)$ in a similar way to the extension of simply typed lambda calculus presented in [Hag87a]. Using the categorical concept of dialgebra as background we obtain the main feature of the type system, the definition of (co)inductive types by means of a tuple of types avoiding the use of sums or products. We call these types *clausular* in analogy to the clausular definitions presented later on chapter three. The strong normalization of the system is proved by embedding it into the first system MICT. Furthermore we sketch another two useful extensions, the first one, $\mathsf{MCICT}_{\boldsymbol{M}}$, includes only Mendler-style (co)induction principles whereas the second one, $\mathsf{MCICT}_{\boldsymbol{\mu M\nu}}$ is a hybrid system with conventional induction and Mendler-style coinduction principles.

On chapter three we present the first part of the main contribution of this work, an extension of $\mathsf{AF2}$ with monotone and clausular (co)inductive definitions called MCICD which corresponds to the type system MCICT under the Curry-Howard correspondence. The use of clauses in the mechanism of (co)inductive definitions allows to set a direct analogy with informal (co)inductive definitions and simplifies the definition of predicates.

The second part of our main contribution, a realizability interpretation for the logic MCICD, is presented on chapter four, the target logic being an extension of MCICD with existential and restricted formulas and where the term language, that is the language of realizers, is nothing but our term system MCICT. Instead of the more frequently used modified realizability interpretation, we use a version of realizability where the first-order universal formulas do not have computational content. A nice application of the realizability soundness theorem is the extension of Krivine and Parigot's *programming with proofs* method to our logic. This method, first presented in [KrPa90], allows to obtain programs over formal data types from proofs in the logic without calculate a single realizer. To illustrate the method a serie of examples is provided.

Problems arised when trying to obtain programs from proofs involving coinductive definitions lead us to chapter six where a solution is provided by means of a hybrid logical system $\mathsf{MCICD}_{\boldsymbol{\mu M\nu}}$ wich includes conventional induction principles and Mendler-style coinduction principles corresponding, of course, to the type system $\mathsf{MCICT}_{\boldsymbol{\mu M\nu}}$. We present the system and give it a realizability interpretation used again to program with proofs. This time specifications involving coinductive definitions are satisfactory programmed.

The thesis concludes with chapter seven which presents some conclusions, related work as well as some suggerences for future work.

No quiso escribir más. Fijó, nuevamente, los ojos en el sol. Se sintió pequeño y ridículo; pequeños y ridículos debían sentirse cuantos trataran de explicar algo de este país. ¿ Explicarlo ? No -se dijo –, creerlo, nada más. México no se explica; en México se cree, con furia, con pasión, con desaliento. Dobló sus cuartillas y se puso de pie.

Carlos Fuentes, La región más transparente.

1
Preliminaries

This chapter is devoted to recall concepts needed later, we assume knowledge of basic logic (in a natural deduction approach) and lambda calculus. Every non-defined concept is assumed to be known. When in doubt the reader should consult the given references.

1.1 Categorical Interlude

We assume some knowledge of category theory, here we only state the basic concepts needed later, for full details on category theory see for example [Mac98]. We will use the categorical approach to (co)induction to formulate our systems of (co)inductive types, this can be briefly stated as follows:

- Induction is the use of initiality for algebras
- Coinduction is the use of finality for coalgebras

For an excellent tutorial for (co)induction from the categorical point of view see [JaRu97], here we give only the basic definitions.

Fix a category $\mathcal{C}$, with products and coproducts for our purposes.

Definition 1.1 *Let $T : \mathcal{C} \to \mathcal{C}$ be a functor. A T-algebra is a pair $\langle A, f\rangle$ such that $f : TA \to A$. Analogously a T-coalgebra is a pair $\langle B, g\rangle$ with $g : B \to TB$.*

Definition 1.2 *Given two T-algebras $\langle A, f\rangle, \langle B, g\rangle$ a morphism from $\langle A, f\rangle$ to*

$\langle B, g\rangle$ is a $\mathcal{C}$-morphism $h : A \to B$ such that the following diagram commutes:

$$\begin{tikzcd}
TA \arrow[r, "f"] \arrow[d, "Th"'] & A \arrow[d, "h"] \\
TB \arrow[r, "g"] & B
\end{tikzcd}$$

We say that the algebra $\langle A, f\rangle$ is initial if it is the initial object of the category of T-algebras, i.e., if for every given algebra $\langle B, g\rangle$ there is a unique h such that the above diagram commutes, in this case the h is denoted It_g and called the iteratively defined morphism with step function g.
If exists, the initial T-algebra is unique and is denoted as $\langle \mu T, \mathsf{in}_T\rangle$, so that $\mathsf{It}_g : \mu T \to B$ and

$$\mathsf{It}_g \circ \mathsf{in}_T = g \circ T(\mathsf{It}_g) \tag{1.1}$$

this equation is called principle of iteration.
Dually a morphism of coalgebras from $\langle B, g\rangle$ to $\langle A, f\rangle$ is a $\mathcal{C}$-morphism $h : B \to A$ such that the following diagram commutes:

$$\begin{tikzcd}
A \arrow[r, "f"] & TA \\
B \arrow[u, "h"] \arrow[r, "g"] & TB \arrow[u, "Th"']
\end{tikzcd}$$

We say that the coalgebra $\langle A, f\rangle$ is final if it is the final object of the category of T-algebras, i.e., if for every given coalgebra $\langle B, g\rangle$ there is a unique h such that the above diagram commutes., in this case we denote such h with CoIt_g and call it the coiteratively defined morphism with step function g.
If exists, the final T-coalgebra is unique and denoted with $\langle \nu T, \mathsf{out}_T\rangle$, so that $\mathsf{CoIt}_g : B \to \nu T$ and

$$\mathsf{out}_T \circ \mathsf{CoIt}_g = F(\mathsf{CoIt}_g) \circ g \tag{1.2}$$

this equation is called principle of coiteration.

Proposition 1.1 $\mathsf{in}_T, \mathsf{out}_T$ *are isomorphisms, therefore there exist inverse morphisms* ${\mathsf{in}_T}^{-1}, {\mathsf{out}_T}^{-1}$ *such that* ${\mathsf{in}_T}^{-1} \circ \mathsf{in}_T = \mathsf{Id}_{T\mu T}$ *and* $\mathsf{out}_T \circ {\mathsf{out}_T}^{-1} = \mathsf{Id}_{\nu T}$*. These equations are called the principle of inductive and coinductive inversion respectively.*
Proof.

Consider the following diagram

$$\begin{tikzcd}
{} & T\mu T \arrow[r, "\mathsf{in}_T"] \arrow[dl, "T(\mathsf{in}_T \circ h)"'] & \mu T \arrow[dl, "\mathsf{in}_T \circ h"] \\
T\mu T \arrow[r, "\mathsf{in}_T"] \arrow[d, "T(h)"'] & \mu T \arrow[d, "h"] & \\
T(T\mu T) \arrow[r, "T(\mathsf{in}_T)"] & T\mu T &
\end{tikzcd}$$

The lower diagram commutes by the universal property of the initial algebra, therefore we have

$$h \circ \mathsf{in}_T = T(\mathsf{in}_T) \circ T(h) = T(\mathsf{in}_T \circ h)$$

the second equality due to the second functor law.

The upper diagram commutes, with help of the lower one, as follows:

$$(\mathsf{in}_T \circ h) \circ \mathsf{in}_T = \mathsf{in}_T \circ (h \circ \mathsf{in}_T) = \mathsf{in}_T \circ (T(\mathsf{in}_T \circ h))$$

Next observe that the upper diagram also commutes with Id instead of $\mathsf{in}_T \circ h$, which by the universal property of the initial algebra implies $\mathsf{in}_T \circ h = \mathsf{Id}$, which implies

$$h \circ \mathsf{in}_T = T(\mathsf{in}_T \circ h) = T(\mathsf{Id}) = \mathsf{Id}$$

the last equality given by the first functor law.

Therefore h is an inverse for in_T and we denote it with ${\mathsf{in}_T}^{-1}$.

The case for the final coalgebra is analogous. ⊣

The extended (co)induction principles will be justified by means of (co)recursive algebras:

Definition 1.3 *Define $\mathsf{\Pi}_D : \mathcal{C} \to \mathcal{C}$ as $\mathsf{\Pi}_D C := C \times D$. We say that the T-algebra $\langle A, f\rangle$ is recursive if for every $T\mathsf{\Pi}_A$-algebra $\langle B, g\rangle$ there exists a morphism $h : A \to B$ such that:*

$$\begin{tikzcd}
TA \arrow[r, "f"] \arrow[d, "T\langle \mathsf{Id}, h\rangle"'] & A \arrow[d, "h"] \\
T(A \times B) \arrow[r, "g"] & B
\end{tikzcd} \tag{1.3}$$

Set $\Sigma_D : \mathcal{C} \to \mathcal{C}$ with $\Sigma_D C := C + D$. We say that the T-coalgebra $\langle A, f\rangle$ is corecursive if for every $T\Sigma_A$-coalgebra $\langle B, g\rangle$ there exists a morphism $h : B \to A$ such that:

$$\begin{tikzcd}
A \arrow[r, "f"] & TA \\
B \arrow[u, "h"] \arrow[r, "g"] & T(A+B) \arrow[u, "{T[\mathsf{Id}, h]}"']
\end{tikzcd} \tag{1.4}$$

Proposition 1.2 *$\langle \mu T, \mathsf{in}_T\rangle$ is recursive and $\langle \nu T, \mathsf{out}_T\rangle$ is corecursive.*
Proof. Let $\langle B, g\rangle$ be a $T\Pi_{\mu T}$-algebra, i.e. $g : T(\mu T \times B) \to B$. It is easy to see that $\mathsf{in}_T \circ T(\pi_1) : T(\mu T \times B) \to \mu T$, so that we get the following T-algebra:

$$\langle \mathsf{in}_T \circ T(\pi_1), g\rangle : T(\mu T \times B) \to \mu T \times B$$

Therefore by iteration there is a unique $h : \mu T \to \mu T \times B$ such that

$$h \circ \mathsf{in}_T = \big\langle \mathsf{in}_T \circ T(\pi_1), g\big\rangle \circ T(h) \tag{1.5}$$

The goal is to show that for the given g there is a $h' : \mu T \to B$ such that

$$\begin{tikzcd}
T\mu T \arrow[r, "\mathsf{in}_T"] \arrow[d, "T(\langle \mathsf{id}, h'\rangle)"'] & \mu T \arrow[d, "h'"] \\
T(\mu T \times B) \arrow[r, "g"] & B
\end{tikzcd} \tag{1.6}$$

Set $h' : \mu T \to B$ defined as $h' := \pi_2 \circ h$, we will show that the diagram commutes, i.e.,

$$h' \circ \mathsf{in}_T = g \circ T(\langle \mathsf{Id}, h'\rangle)$$

First we show that $\pi_1 \circ h = \mathsf{Id}$, by initiality, i.e. we have to show that the following diagram commute:

$$\begin{tikzcd}
T\mu T \arrow[r, "\mathsf{in}_T"] \arrow[d, "T(\pi_1 \circ h)"'] & \mu T \arrow[d, "\pi_1 \circ h"] \\
T\mu T \arrow[r, "\mathsf{in}_T"] & \mu T
\end{tikzcd}$$

we have by equation (1.5)

$$(\pi_1 \circ h) \circ \mathsf{in}_T = \pi_1 \circ (h \circ \mathsf{in}_T) = \pi_1 \circ \Big(\langle \mathsf{in}_T \circ T(\pi_1), g \rangle \circ T(h) \Big) =$$

$$= \big(\pi_1 \circ \langle \mathsf{in}_T \circ T(\pi_1), g\rangle\big) \circ T(h) = \big(\, \mathsf{in}_T \circ T(\pi_1)\big) \circ T(h) = \mathsf{in}_T \circ T(\pi_1 \circ h)$$

Therefore the diagram commutes and by uniqueness we have $\pi_1 \circ h = \mathsf{Id}$.

Next observe that $h = \langle \pi_1 \circ h, \pi_2 \circ h \rangle = \langle \mathsf{Id}, h' \rangle$. Now we can show that diagram (1.6) commutes:

$$\begin{array}{rcl} h' \circ \mathsf{in}_T & = & \big(\pi_2 \circ h\big) \circ \mathsf{in}_T \\ & = & \pi_2 \circ \big(h \circ \mathsf{in}_T\,\big) \\ & = & \pi_2 \circ \Big(\langle \mathsf{in}_T \circ T(\pi_1), g\rangle \circ T(h)\Big) \\ & = & \Big(\pi_2 \circ \langle \mathsf{in}_T \circ T(\pi_1), g\rangle\Big) \circ T(h) \\ & = & g \circ T(h) \\ & = & g \circ T(\langle \mathsf{Id}, h' \rangle) \end{array}$$

Therefore diagram (1.6) commutes.

The case for the final coalgebra is similar.

⊣

For the cases of the initial algebra and the final coalgebra, the h that makes diagrams (1.3), (1.4) commute is denoted $\mathsf{Rec}_g, \mathsf{CoRec}_g$ respectively and we refer to them as the (co)recursively defined morphism with step function g, so that we have $\mathsf{Rec}_g : \mu T \to B$, $\mathsf{CoRec}_g : B \to \nu T$ such that the following principles hold:

- Principle of Primitive Recursion

$$\mathsf{Rec}_g \circ \mathsf{in}_T = g \circ T(\langle \mathsf{Id}, \mathsf{Rec}_g \rangle) \tag{1.7}$$

- Principle of Primitive Corecursion

$$\mathsf{out}_T \circ \mathsf{CoRec}_g = T([\mathsf{Id}, \mathsf{CoRec}_g]) \circ g \tag{1.8}$$

1.1.1 M-(Co)algebras

In this section we justify categorically the concept of Mendler-style (co)induction ([Men87, Men91]),which will provide an important tool for coinductive programming. For a deep explanation of Mendler-style from the categorical point of view see [UV99, UV00].

Definition 1.4 *A T-Mendler-style-algebra, for short T-*M*-algebra, is a pair $\langle A, \Phi \rangle$ with*

$$\Phi : \mathsf{Hom}(\cdot, A) \to \mathsf{Hom}(T\cdot, A)$$

that is, Φ is a transformation taking morphisms $f : C \to A$ to morphisms $\Phi f : TC \to A$, for every object C, and such that for every object B and morphism $g : B \to A$ we have:

$$\begin{tikzcd}
TB \arrow[r, "\Phi(g)"] \arrow[d, "Tg"'] & A \\
TA \arrow[ur, "\Phi(\mathsf{Id})"'] &
\end{tikzcd}$$

$$\Phi(g) = \Phi(\mathsf{Id}) \circ Tg$$

*A morphism of T-*M*-algebras $\langle D, \Psi\rangle, \langle C, \Phi\rangle$ is a morphism $h : D \to C$ such that:*

$$\begin{tikzcd}
TD \arrow[r, "\Psi(\mathsf{Id})"] \arrow[dr, "\Phi(h)"'] & D \arrow[d, "h"] \\
& C
\end{tikzcd}$$

$$h \circ \Psi(\mathsf{Id}) = \Phi(h)$$

Proposition 1.3 *Let T be a functor with initial algebra $\langle \mu T, \mathsf{in}_T\rangle$. Then for every T-*M*-algebra $\langle C, \Phi\rangle$ there is a unique morphism $\mathsf{MIt}_\Phi : \mu T \to C$ such that*

$$\begin{tikzcd}
T\mu T \arrow[r, "\mathsf{in}_T"] \arrow[dr, "\Phi(\mathsf{MIt}_\Phi)"'] & \mu T \arrow[d, "\mathsf{MIt}_\Phi"] \\
& C
\end{tikzcd}$$

so that the principle of Mendler-style iteration holds:

$$\mathsf{MIt}_\Phi \circ \mathsf{in}_T = \Phi(\mathsf{MIt}_\Phi) \tag{1.9}$$

MIt_Φ is the morphism defined by Mendler-style iteration with step function Φ.

Definition 1.5 *A T-algebra $\langle A, f\rangle$ is* M*-recursive if for every object C and every transformation*

$$\Phi : \mathsf{Hom}(\cdot, A) \to \mathsf{Hom}(\cdot, C) \to \mathsf{Hom}(T\cdot, C)$$

there exists an $h : A \to C$ such that:

$$\begin{tikzcd}
TA \arrow[r, "f"] \arrow[dr, "\Phi(\mathsf{Id})(h)"'] & A \arrow[d, "h"] \\
 & C
\end{tikzcd} \tag{1.10}$$

$$f \circ h = \Phi(\mathsf{Id})(h)$$

Proposition 1.4 *The initial algebra $\langle \mu T, \mathsf{in}_T\rangle$ is* M*-recursive.In this case the h of diagram* (1.10) *is denoted* MRec_Φ *and the equation*

$$\mathsf{in}_T \circ \mathsf{MRec}_\Phi = \Phi(\mathsf{Id})(\mathsf{MRec}_\Phi) \tag{1.11}$$

is called the principle of Mendler-style recursion, whereas MRec_Φ *is called the morphism defined by Mendler-style recursion with step function Φ.*

Dualizing the previous definitions we justify Mendler-style coinduction.

Definition 1.6 *A T-Mendler-style-coalgebra, for short T-*M*-coalgebra, is a pair $\langle D, \Phi\rangle$ with*

$$\Phi : \mathsf{Hom}(D, \cdot) \to \mathsf{Hom}(D, T\cdot)$$

and such that for every object A and morphism $g : D \to A$ we have:

$$\begin{tikzcd}
D \arrow[r, "\Phi(g)"] \arrow[d, "\Phi(\mathsf{Id})"'] & TA \\
TD \arrow[ur, "Tg"'] &
\end{tikzcd}$$

$$\Phi(g) = Tg \circ \Phi(\mathsf{Id})$$

*A morphism of T-*M*-coalgebras $\langle D, \Phi\rangle, \langle E, \Psi\rangle$ is a morphism $h : D \to E$ such that*

$$\begin{tikzcd}
D \arrow[r, "\Phi(\mathsf{Id})"] \arrow[d, "h"'] & TD \\
E \arrow[ur, "\Psi(h)"'] &
\end{tikzcd}$$

$$\Psi(h) \circ h = \Phi(\mathsf{Id})$$

Proposition 1.5 *Let T be a functor with final coalgebra $\langle \nu T, \mathsf{out}_T\rangle$. Then for every T-*M*-coalgebra $\langle D, \Phi\rangle$ there is a unique morphism $\mathsf{MCoIt}_\Phi : D \to \nu T$ such that*

$$\begin{tikzcd}
T\nu T & D \arrow[l, "\Phi(\mathsf{MCoIt}_\Phi)"'] \arrow[d, "\mathsf{MCoIt}_\Phi"] \\
& \nu T \arrow[ul, "\mathsf{out}_T"]
\end{tikzcd}$$

so that the principle of Mendler-style coiteration holds:

$$\mathsf{out}_T \circ \mathsf{MCoIt}_\Phi = \Phi(\mathsf{MCoIt}_\Phi) \tag{1.12}$$

MCoIt_Φ *is the morphism defined by Mendler-style iteration with step function Φ.*

Definition 1.7 *A T-coalgebra $\langle A, f\rangle$ is* M*-corecursive if for every object D and every transformation*

$$\Phi : \mathsf{Hom}(A, \cdot) \to \mathsf{Hom}(D, \cdot) \to \mathsf{Hom}(D, T\cdot)$$

there exists an $h : D \to A$ such that

$$\begin{tikzcd}
A \arrow[r, "f"] & TA \\
& D \arrow[ul, "h"] \arrow[u, "\Phi(\mathsf{Id})(h)"']
\end{tikzcd} \tag{1.13}$$

$$f \circ h = \Phi(\mathsf{Id})(h)$$

Proposition 1.6 *The final coalgebra $\langle \nu T, \mathsf{out}_T \rangle$ is* M*-corecursive. In this case the h of diagram* (1.13) *is denoted* MCoRec_Φ *and the equation*

$$\mathsf{out}_T \circ \mathsf{MCoRec}_\Phi = \Phi(\mathsf{Id})(\mathsf{MCoRec}_\Phi) \tag{1.14}$$

is called the principle of Mendler-style corecursion,whereas MCoRec_Φ *is called the morphism defined by Mendler-style corecursion with step function Φ.*

1.1.2 Dialgebras

The concept of dialgebra, introduced in [Hag87b], is a straightforward generalization of (co)algebras with stronger expressive power (see [PZ01]). With dialgebras we can represent products, coproducts and even exponential objects (see [DM93]). We will serve later from this concept to justify the clausular feature of our type/logic systems.

Definition 1.8 *Let $F, G : \mathcal{C} \to \mathcal{D}$ covariant functors between two categories $\mathcal{C}, \mathcal{D}$. A F, G-dialgebra is a pair $\langle A, f \rangle$ where A is a $\mathcal{C}$-object and $f : FA \to GA$ is a D-morphism.*

Definition 1.9 *A morphism between two F, G-dialgebras $\langle A, f \rangle, \langle B, g \rangle$ is a $\mathcal{C}$-morphism $h : A \to B$ such that:*

$$\begin{tikzcd}
FA \arrow[r, "f"] \arrow[d, "Fh"'] & GA \arrow[d, "Gh"] \\
FB \arrow[r, "g"] & GB
\end{tikzcd}$$

Observe that if I is the identity functor then a T, I-dialgebra $\langle A, f \rangle$ is a T-algebra and a I, T-dialgebra is a T-coalgebra.

We are specially interested in dialgebras where the functors $F, G : \mathcal{C} \to \mathcal{C}^n$ are of the form

$$F \equiv \langle F_1, \ldots, F_n \rangle \qquad G \equiv \langle I, \ldots, I \rangle$$

with $F_i : \mathcal{C} \to \mathcal{C}$.

The final G, F-dialgebra, if exists, will be denoted with $\langle \nu(F_1, \ldots, F_n), \mathsf{out}_n \rangle$

The finality of $\nu(F_1, \ldots, F_n)$ is given by the following diagram, where $V := \nu(F_1, \ldots, F_n)$

$$\begin{tikzcd}
\langle V, \dots, V \rangle \arrow[r, "\mathsf{out}_n"] & \langle F_1 V, \dots, F_n V \rangle \\
\langle B, \dots, B \rangle \arrow[u, "{\langle h, \dots, h \rangle}"] \arrow[r, "g"] & \langle F_1 B, \dots, F_n B \rangle \arrow[u, swap, "{\langle F_1 h, \dots, F_n h \rangle}"]
\end{tikzcd}$$

where $h : B \to V$ is the unique function such that:

$$\mathsf{out}_n \circ \langle h, \dots, h \rangle = \langle F_1 h, \dots, F_n h \rangle \circ g$$

Observing that the morphisms out_n, g are neccesary of the form

$$\mathsf{out}_n = \langle \mathsf{out}_{n,1}, \dots, \mathsf{out}_{n,n} \rangle \qquad g = \langle g_1, \dots, g_n \rangle.$$

The previous diagram can be splitted into the following n-diagrams, denoting with CoIt^n_g to the unique h above.

$$\begin{tikzcd}
\nu(F_1, \dots, F_n) \arrow[r, "\mathsf{out}_{n,i}"] & F_i\big(\nu(F_1, \dots, F_n)\big) \\
B \arrow[u, "\mathsf{CoIt}^n_g"] \arrow[r, "g_i"] & F_i B \arrow[u, swap, "F_i(\mathsf{CoIt}^n_g)"]
\end{tikzcd}$$

$$\mathsf{out}_{n,i} \circ \mathsf{CoIt}^n_g = F_i(\mathsf{CoIt}^n_g) \circ g_i \tag{1.15}$$

These equations represent the coiteration principle on dialgebras
Analogously corecursion is introduced by the following n-diagrams :

$$\begin{tikzcd}
\nu(F_1, \dots, F_n) \arrow[r, "\mathsf{out}_{n,i}"] & F_i\big(\nu(F_1, \dots, F_n)\big) \\
B \arrow[u, "\mathsf{CoRec}^n_g"] \arrow[r, "g_i"] & F_i\big(\nu(F_1, \dots, F_n) + B\big) \arrow[u, swap, "{F_i\big([\mathsf{Id}, \mathsf{CoRec}^n_g]\big)}"]
\end{tikzcd}$$

$$\mathsf{out}_{n,i} \circ \mathsf{CoRec}^n_g = F_i\big([\mathsf{Id}, \mathsf{CoRec}^n_g]\big) \circ g_i \tag{1.16}$$

This equations represent the principle of primitive corecursion on dialgebras

Finally the coinductive inversion principle is given by this equations:

$$\mathsf{out}_k \circ \mathsf{out}_k^{-1} = \mathsf{Id}_{\langle F_1 V, \ldots, F_n V\rangle} \tag{1.17}$$

Similarly denoting with $\langle \mu(F_1, \ldots, F_n), \mathsf{in}_n\rangle$ the initial F, G-dialgebra we arrive to the following diagrams:

$$\begin{tikzcd}
F_i\big(\mu(F_1, \ldots, F_n)\big) \arrow[r, "\mathsf{in}_{n,i}"] \arrow[d, "F_i(\mathsf{It}^n_g)"'] & \mu(F_1, \ldots, F_n) \arrow[d, "\mathsf{It}^n_g"] \\
F_i B \arrow[r, "g_i"] & B
\end{tikzcd}$$

representing the iteration principle:

$$\mathsf{It}^n_g \circ \mathsf{in}_{n,i} = g_i \circ F_i(\mathsf{It}^n_g) \tag{1.18}$$

$$\begin{tikzcd}
F_i\big(\mu(F_1, \ldots, F_n)\big) \arrow[r, "\mathsf{in}_{n,i}"] \arrow[d, "F_i\big(\langle \mathsf{Id}, \mathsf{Rec}^n_g\rangle\big)"'] & \mu(F_1, \ldots, F_n) \arrow[d, "\mathsf{Rec}^n_g"] \\
F_i\big(\mu(F_1, \ldots, F_n) \times B\big) \arrow[r, "g_i"] & B
\end{tikzcd}$$

representing the recursion principle

$$\mathsf{Rec}^n_g \circ \mathsf{in}_{n,i} = g_i \circ F_i\big(\langle \mathsf{Id}, \mathsf{Rec}^n_g\rangle\big) \tag{1.19}$$

Finally the inductive inversion principle is given by:

$$\mathsf{in}_k^{-1} \circ \mathsf{in}_k = \mathsf{Id}_{\langle \mu(F_1, \ldots, F_n), \ldots, \mu(F_1, \ldots, F_n)\rangle} \tag{1.20}$$

Mendler Style (Co)induction on Dialgebras

In an analogous way to the results in section 1.1.1 we can define Mendler-style (co)iteration and (co)recursion on dialgebras, here we only state such principles. For $F = \langle F_1, \ldots, F_n \rangle, G = \langle I, \ldots, I \rangle$ with initial F, G-dialgebra $\langle \mu(F_1, \ldots, F_n), \mathsf{in}_n \rangle$ where $\mathsf{in}_n = \langle \mathsf{in}_{n,1}, \ldots, \mathsf{in}_{n,n} \rangle$, given the step function $\Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$ we have the following principles:

- Mendler-Style Iteration

$$\mathsf{MIt}^n_\Phi \circ \mathsf{in}_{n,i} = \Phi_i(\mathsf{MIt}^n_\Phi) \tag{1.21}$$

- Mendler-Style Recursion

$$\mathsf{MRec}^n_\Phi \circ \mathsf{in}_{n,i} = \Phi_i(\mathsf{Id})(\mathsf{MRec}^n_\Phi) \tag{1.22}$$

Analogously for the final G, F-dialgebra $\langle \nu(F_1, \ldots, F_n), \mathsf{out}_n \rangle$ where $\mathsf{out}_n = \langle \mathsf{out}_{n,1}, \ldots, \mathsf{out}_{n,n} \rangle$ and given the step function $\Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$ we have the following principles:

- Mendler-Style Coiteration

$$\mathsf{out}_{n,i} \circ \mathsf{MCoIt}^n_\Phi = \Phi_i(\mathsf{MCoIt}^n_\Phi) \tag{1.23}$$

- Mendler-Style Corecursion

$$\mathsf{out}_{n,i} \circ \mathsf{MCoRec}^n_\Phi = \Phi_i(\mathsf{Id})(\mathsf{MCoRec}^n_\Phi) \tag{1.24}$$

This finishes our categorical interlude, in the next two section we introduce our basic type system as well as the second-order logic AF2.

1.2 The Type System F

Our basic type system is the second order polymorphic lambda calculus, also known as system F, introduced independently by Girard [Gir72] (see also [GLT89]) and Reynolds [Rey74]. Like all systems in this work we present system F in Curry-style (see [Bar93] for an explanation of this terminology), that is, as a type assignment system.

The types are generated by the following grammar:

$$\sigma, \rho ::= \alpha \mid \sigma \to \rho \mid \forall \alpha \sigma$$

The set of free variables of σ denoted $FV(\sigma)$ is defined as usual, as well as the substitution concept $\rho[\vec{\alpha} := \vec{\sigma}\,]$ avoding the capture of bounded variables.

The terms are defined as follows:

$$t, r, s \quad ::= \quad x \mid \lambda x r \mid rs$$

The set $FV(t)$ and the concept $t[\vec{x} := \vec{s}]$ are defined as usual.

The type assignment relation between a context $\Sigma = \{x_1 : \rho_1, \ldots, x_k : \rho_k\}$ a term t and a type ρ, denoted

$$\Sigma \rhd t : \rho$$

which can be read as “The term t inhabits the type ρ in the context Σ“, is defined as usual:

$$\frac{}{\Sigma, x : \sigma \rhd x : \sigma}\ (Var)$$

$$\frac{\Sigma, x : \sigma \rhd t : \rho}{\Sigma \rhd \lambda x t : \sigma \to \rho}\ (\to I) \qquad \frac{\Sigma \rhd r : \sigma \to \rho \quad \Sigma \rhd s : \sigma}{\Sigma \rhd rs : \rho}\ (\to E)$$

$$\frac{\Sigma \rhd t : \rho \quad \alpha \notin FV(\Sigma)}{\Sigma \rhd t : \forall \alpha \rho}\ (\forall I) \qquad \frac{\Sigma \rhd t : \forall \alpha \rho}{\Sigma \rhd t : \rho[\alpha := \sigma]}\ (\forall E)$$

The reduction relation $\to_\beta$, which provides the operational semantics of the language, is the term closure of the following relation $\mapsto_\beta$ between terms:

$$(\lambda x r)s \quad \mapsto_\beta \quad r[x := s]$$

As our presentation is in Curry-style the pure term system corresponds to the untyped lambda calculus, there is neither type decorations in terms like $\lambda x^\rho r$ nor type abstractions or applications like $\Lambda \alpha r, r\sigma$.

This finish the definition of our language as a typed term rewrite system $\langle \mathsf{F}, \to_\beta, \rhd \rangle$.

To show the expressive power of F we give some examples of interesting types

Natural Numbers in F

Define the type of natural numbers as follows:

$$\mathsf{nat} := \forall \alpha . \alpha \to (\alpha \to \alpha) \to \alpha$$

The canonical inhabitants of this type are the Church numerals defined as:

$$\widetilde{n} := \lambda x \lambda f . f^n(x)$$

where $f^0(x) := x$ and $f^{n+1}(x) := f(f^n(x))$.

The constructors of nat are defined as:

$$\begin{aligned} 0 &:= \widetilde{0} := \lambda x \lambda f . x \\ s &:= \lambda n \lambda x \lambda f . f(nxf) \end{aligned}$$

It is easy to check that $\rhd 0 : \mathsf{nat}$ and $\rhd s : \mathsf{nat} \to \mathsf{nat}$.

Streams in F

The type of streams (infinite lists) of objects of type ρ is defined as follows:

$$\mathsf{stream}(\rho) := \forall\gamma.\big(\forall\alpha.(\alpha\to\rho)\to(\alpha\to\alpha)\to\alpha\to\gamma\big)\to\gamma$$

with destructors $\mathsf{tail}, \mathsf{head}$ defined as:

$$\begin{aligned}\mathsf{head} &:= \lambda s.s\big(\lambda h\lambda t\lambda x.hx\big)\\ \mathsf{tail} &:= \lambda s.s\big(\lambda h\lambda t\lambda x.\mathsf{build}ht(tx)\big)\end{aligned}$$

where $\mathsf{build} := \lambda h\lambda t\lambda x\lambda f.(fhtx)$ with

$$\rhd\mathsf{build} : \forall\alpha.(\alpha\to\rho)\to(\alpha\to\alpha)\to\alpha\to\mathsf{stream}(\rho).$$

With this definitions we get $\rhd\,\mathsf{head} : \mathsf{stream}(\rho)\to\rho$ and $\rhd\,\mathsf{tail} : \mathsf{stream}(\rho)\to\mathsf{stream}(\rho)$.

Two very important properties of typed term rewrite systems are strong normalization (termination of rewriting) and subject reduction (type preservation), the latter property being not trivial in type assignment systems, like the ones considered here, because of the presence of implicit polymorphism, a typed term $\Sigma \rhd t : \forall\alpha\rho$ inhabits also all the instances of ρ, i.e., $\Sigma \rhd t : \rho[\alpha := \sigma]$ for every σ, due to this feature the application of an introduction or elimination rule for universal types cannot be traced by only looking at the terms, such rules are called *non-traceable.*

Let us recall the definitions of both properties.

Definition 1.10 *A typed term rewrite system $\langle\mathcal{T},\rightsquigarrow,\rhd\rangle$ has subject reduction if the following holds: If $\Sigma \rhd r : \rho$ and $r \rightsquigarrow s$ then $\Sigma \rhd s : \rho$.*

Definition 1.11 *A typed term rewrite system $\langle\mathcal{T},\rightsquigarrow,\rhd\rangle$ is strongly normalizing if for every typable term $\Sigma \rhd t : \sigma$ there is no infinite reduction sequence $(r_i)_{i\in\mathbb{N}}$ with $r_0 \equiv t$ and $r_i \rightsquigarrow r_{i+1}$ for every $i \in \mathbb{N}$. That is, every reduction sequence starting in t terminates.*

It is well-known that system F enjoys subject reduction and strongly normalizes (see for example [Kri93], a direct proof of strong normalization is given by corollary 1.3 below).

1.2.1 Adding Sum and Product Types

Although system F is highly expressive we prefer to add sum and product types to our basic framework for comfort and because of some technicalities that will be clear later.

Extend system F as follows:

Types:

$$\sigma, \rho ::= \ldots \mid \sigma + \rho \mid \sigma \times \rho$$

Terms :

$$\begin{array}{rcl} t, r, s & ::= & \ldots \mid \mathsf{inl}\, r \mid \mathsf{inr}\, s \mid \mathsf{case}(r, x.s, y.t) \mid \\ & & \langle r, s\rangle \mid \pi_1 r \mid \pi_2 r \end{array}$$

Type Assignment:

$$\frac{\Sigma \rhd r : \rho}{\Sigma \rhd \mathsf{inl}\, r : \rho + \sigma}\ (+I_L) \qquad \frac{\Sigma \rhd r : \sigma}{\Sigma \rhd \mathsf{inr}\, r : \rho + \sigma}\ (+I_R)$$

$$\frac{\Sigma \rhd r : \rho + \sigma \quad \Sigma, x : \rho \rhd s : \tau \quad \Sigma, y : \sigma \rhd t : \tau}{\Sigma \rhd \mathsf{case}(r, x.s, y.t) : \tau}\ (+E)$$

$$\frac{\Sigma \rhd r : \rho \quad \Sigma \rhd s : \sigma}{\Sigma \rhd \langle r, s\rangle : \rho \times \sigma}\ (\times I) \quad \frac{\Sigma \rhd s : \rho \times \sigma}{\Sigma \rhd \pi_1 s : \rho}\ (\times E_L) \quad \frac{\Sigma \rhd s : \rho \times \sigma}{\Sigma \rhd \pi_2 s : \sigma}\ (\times E_R)$$

Reduction Relation:

$$\begin{array}{rcl} \mathsf{case}(\mathsf{inl}\, r, x.s, y.t) & \mapsto_\beta & s[x := r] \\ \mathsf{case}(\mathsf{inr}\, r, x.s, y.t) & \mapsto_\beta & t[y := r] \\ \pi_1\langle r, s\rangle & \mapsto_\beta & r \\ \pi_2\langle r, s\rangle & \mapsto_\beta & s \end{array}$$

We call to this extension $\mathsf{F}^{+,\times}$.

$\mathsf{F}^{+,\times}$ enjoys subject reduction which can be proven by adapting the method for system F in [Kri93].
Strong normalization can be proved by embedding it into system F. Nevertheless and in the spirit of modularity we reprove strong normalization via Matthes' SN-method developed in [Mat98], later we will extend this proof to the basic system of (co)inductive types.

Strong Normalization for $\mathsf{F}^{+,\times}$

Definition 1.12 *Let $\star$ be a new symbol. An elimination is an expression of one of the following forms:*

$$\star s,\ \mathsf{case}(\star, x.t, y.r),\ \pi_1\star,\ \pi_2\star$$

eliminations are denoted with the letter e.

Definition 1.13 *Multiple eliminations are defined as follows:*

$$E ::= \star \mid e[\star := E]$$

where $e[\star := E]$ is defined as if $\star$ were a term variable. From now on we will use $E[r]$ to denote $E[\star := r]$.

Definition 1.14 *The set* SN *is inductively defined as follows:*

$$\frac{}{x \in \mathsf{SN}} \qquad \frac{E[x], s \in \mathsf{SN}}{E[x]s \in \mathsf{SN}} \qquad \frac{E[x], s, t \in \mathsf{SN}}{\mathsf{case}(E[x], x.s, y.t) \in \mathsf{SN}}$$

$$\frac{E[x] \in \mathsf{SN}}{\pi_1(E[x]) \in \mathsf{SN}} \qquad \frac{E[x] \in \mathsf{SN}}{\pi_2(E[x]) \in \mathsf{SN}}$$

$$\frac{r \in \mathsf{SN}}{\lambda xr \in \mathsf{SN}} \qquad \frac{E\big[r[x := s]\big], s \in \mathsf{SN}}{E[(\lambda xr)s] \in \mathsf{SN}} \qquad \frac{t \in \mathsf{SN}}{\mathsf{inl}\, t \in \mathsf{SN}} \qquad \frac{t \in \mathsf{SN}}{\mathsf{inr}\, t \in \mathsf{SN}}$$

$$\frac{E\big[r[x := t]\big], s \in \mathsf{SN}}{E[\mathsf{case}(\mathsf{inl}\, t, x.r, y.s)] \in \mathsf{SN}} \qquad \frac{E\big[s[y := t]\big], r \in \mathsf{SN}}{E\big[\mathsf{case}(\mathsf{inr}\, t, x.r, y.s)\big] \in \mathsf{SN}}$$

$$\frac{r, s \in \mathsf{SN}}{\langle r, s\rangle \in \mathsf{SN}} \qquad \frac{E[r], s \in \mathsf{SN}}{E[\pi_1\langle r, s\rangle] \in \mathsf{SN}} \qquad \frac{E[s], r \in \mathsf{SN}}{E[\pi_2\langle r, s\rangle] \in \mathsf{SN}}$$

Definition 1.15 (Saturated Set) *A set $\mathcal{M}$ of terms is saturated iff:*

$$\mathcal{M} \subseteq \mathsf{SN}$$

and if the following closure conditions hold:

$$\frac{E[x] \in \mathsf{SN}}{E[x] \in \mathcal{M}}$$

$$\frac{E\big[r[x := s]\big] \in \mathcal{M} \quad s \in \mathsf{SN}}{E[(\lambda xr)s] \in \mathcal{M}}$$

$$\frac{E\big[r[x := t]\big] \in \mathcal{M} \quad s \in \mathsf{SN}}{E[\mathsf{case}(\mathsf{inl}\, t, x.r, y.s)] \in \mathcal{M}} \qquad \frac{E\big[s[y := t]\big] \in \mathcal{M} \quad r \in \mathsf{SN}}{E\big[\mathsf{case}(\mathsf{inr}\, t, x.r, y.s)\big] \in \mathcal{M}}$$

$$\frac{E[r] \in \mathcal{M} \quad s \in \mathsf{SN}}{E[\pi_1\langle r, s\rangle] \in \mathcal{M}} \qquad \frac{E[s] \in \mathcal{M} \quad r \in \mathsf{SN}}{E[\pi_2\langle r, s\rangle] \in \mathcal{M}}$$

the set of saturated sets will be denoted with SAT*,*

Lemma 1.1 *The following holds:*

- $\mathsf{SN} \in \mathsf{SAT}$
- *If* $\mathcal{U} \subseteq \mathsf{SAT}$ *then* $\bigcap \mathcal{U} \in \mathsf{SAT}$

Proof. Straightforward ⊣

Definition 1.16 *Given a set of terms M we define the saturated closure of M as follows:*

$$\mathsf{cl}(M) := \bigcap\{\mathcal{N} \in \mathsf{SAT} \mid M \cap \mathsf{SN} \subseteq \mathcal{N}\}$$

$\mathsf{cl}(M)$ is the least saturated set which contains $M \cap \mathsf{SN}$. Observe that $M \subseteq \mathsf{cl}(M)$ if and only if $M \subseteq \mathsf{SN}$.

Definition 1.17 *Given a variable x and $\mathcal{M}, \mathcal{N} \in \mathsf{SAT}$ we define*

$$\mathsf{S}_x(\mathcal{M}, \mathcal{N}) := \{t \mid \forall s \in \mathcal{M}.\ t[x := s] \in \mathcal{N}\}$$

Definition 1.18 *Given $\mathcal{M}, \mathcal{N} \in \mathsf{SAT}$, we define the following sets:*

$$\begin{array}{rcl}
\mathcal{I}_{\to}(\mathcal{M}, \mathcal{N}) & := & \{\lambda x t \mid t \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})\} \\
\mathcal{I}_{+}(\mathcal{M}, \mathcal{N}) & := & \{\mathsf{inl}\, t \mid t \in \mathcal{M}\} \cup \{\mathsf{inr}\, t \mid t \in \mathcal{N}\} \\
\mathcal{I}_{\times}(\mathcal{M}, \mathcal{N}) & := & \{\langle s, t\rangle \mid s \in \mathcal{M} \textit{ and } t \in \mathcal{N}\} \\
\mathcal{M} \to \mathcal{N} & := & \mathsf{cl}(\mathcal{I}_{\to}(\mathcal{M}, \mathcal{N})) \\
\mathcal{M} + \mathcal{N} & := & \mathsf{cl}(\mathcal{I}_{+}(\mathcal{M}, \mathcal{N})) \\
\mathcal{M} \times \mathcal{N} & := & \mathsf{cl}(\mathcal{I}_{\times}(\mathcal{M}, \mathcal{N})) \\
\mathcal{E}_{\to}(\mathcal{M}, \mathcal{N}) & := & \{r \in \mathsf{SN} \mid \forall s \in \mathcal{M}.\ rs \in \mathcal{N}\}
\end{array}$$

$$\begin{array}{rcl}
\mathcal{E}_{+}(\mathcal{M}, \mathcal{N}) & := & \{r \in \mathsf{SN} \mid \quad \forall \mathcal{P} \forall x \forall s \in \mathsf{S}_x(\mathcal{M}, \mathcal{P}) \forall y \forall t \in \mathsf{S}_y(\mathcal{N}, \mathcal{P}). \\
 & & \qquad \mathsf{case}(r, x.s, y.t) \in \mathcal{P}\} \\
\mathcal{E}_{\times}(\mathcal{M}, \mathcal{N}) & := & \{r \in \mathsf{SN} \mid \pi_1 r \in \mathcal{M} \textit{ and } \pi_2 r \in \mathcal{N}\}
\end{array}$$

Lemma 1.2 *For $\diamond \in \{\to, +, \times\}$ we have $\mathcal{I}_\diamond(\mathcal{M}, \mathcal{N}) \subseteq \mathsf{SN}$.*
Proof. The proof is straightforward, as example we show the case $\diamond = \times$. Take $\langle s, t\rangle \in \mathcal{I}_{\times}(\mathcal{M}, \mathcal{N})$, i.e., $s \in \mathcal{M}$ and $t \in \mathcal{N}$, but as $\mathcal{M}, \mathcal{N} \in \mathsf{SAT}$ we have $\mathcal{M}, \mathcal{N} \subseteq \mathsf{SN}$. Therefore $s, t \in \mathsf{SN}$ which implies $\langle s, t\rangle \in \mathsf{SN}$. ⊣

Corollary 1.1 *For $\diamond \in \{\to, +, \times\}$ and the same in both choices, we have*

$$\mathcal{I}_\diamond(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} \diamond \mathcal{N}.$$

Proof. Again we only treat the case for $\diamond = \times$. We have to show that $\mathcal{I}_\times(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} \times \mathcal{N}$, but by definition $\mathcal{M} \times \mathcal{N} = \mathsf{cl}(\mathcal{I}_\times(\mathcal{M}, \mathcal{N}))$ and we know that $\mathcal{I}_\times(\mathcal{M}, \mathcal{N}) \cap \mathsf{SN} \subseteq \mathsf{cl}(\mathcal{I}_\times(\mathcal{M}, \mathcal{N}))$, which by the previous lemma is the same as $\mathcal{I}_\times(\mathcal{M}, \mathcal{N}) \subseteq \mathsf{cl}(\mathcal{I}_\times(\mathcal{M}, \mathcal{N}))$ and we are done. $\dashv$

Lemma 1.3 *For $\diamond \in \{\to, +, \times\}$ we have $\mathcal{E}_\diamond(\mathcal{M}, \mathcal{N}) \in \mathsf{SAT}$.*
Proof. The proof is straightforward, as example we treat the case for $\diamond = \times$. $\mathcal{E}_\times(\mathcal{M}, \mathcal{N}) \subseteq \mathsf{SN}$ is clear. Take $E[\big[r[x := s]\big]] \in \mathcal{E}_\times(\mathcal{M}, \mathcal{N})$ and $s \in \mathsf{SN}$, we have to show $E[(\lambda x r)s] \in \mathcal{E}_\times(\mathcal{M}, \mathcal{N})$. As $E[\big[r[x := s]\big]] \in \mathcal{E}_\times(\mathcal{M}, \mathcal{N})$ we have $\pi_1(E[\big[r[x := s]\big]]) \in \mathcal{M}$ and $\pi_2(E[\big[r[x := s]\big]]) \in \mathcal{N}$. Observe that $\pi_1(E\big[r[x := s]\big]) \equiv (\pi_1 \star)\big[\star := E\big[r[x := s]\big]\big] \equiv (\pi_1 \star)[\star := E]\big[r[x := s]\big]$ and that $(\pi_1 \star)[\star := E]$ is again a multiple elimination say E', therefore we have $E'\big[r[x := s]\big] \in \mathcal{M}$, and as $s \in \mathsf{SN}$ and $\mathcal{M} \in \mathsf{SAT}$ we get $E'[(\lambda x r)s] \in \mathsf{SN}$, i.e., $\pi_1(E[(\lambda x r)s]) \in \mathcal{M}$. Analogously we show that $\pi_2(E[(\lambda x r)s]) \in \mathcal{N}$. Therefore $E[(\lambda x r)s] \in \mathcal{E}_\times(\mathcal{M}, \mathcal{N})$. The other rules for SAT sets are proved similarly. $\dashv$

Lemma 1.4 *For $\diamond \in \{\to, +, \times\}$ and the same in both choices, we have*

$$\mathcal{I}_\diamond(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{E}_\diamond(\mathcal{M}, \mathcal{N}).$$

Proof. For $\diamond = \times$ take $\langle s, t\rangle \in \mathcal{I}_\times(\mathcal{M}, \mathcal{N})$. We have to show that $\langle s, t\rangle \in \mathcal{E}_\times(\mathcal{M}, \mathcal{N})$, i.e., $\pi_1\langle s, t\rangle \in \mathcal{M}$ and $\pi_2\langle s, t\rangle \in \mathcal{N}$. As $\langle s, t\rangle \in \mathcal{I}_\times(\mathcal{M}, \mathcal{N})$ we have $s \in \mathcal{M}$ and $t \in \mathcal{N}$. Observe that $s \equiv \star[s] \in \mathcal{M}$ is a multiple elimination and $t \in \mathsf{SN}$, because $\mathcal{N} \subseteq \mathsf{SN}$. Therefore as $\mathcal{M} \in \mathsf{SAT}$, we have $\star[\pi_1\langle s, t\rangle] \in \mathcal{M}$. i.e., $\pi_1\langle s, t\rangle \in \mathcal{M}$ and analogously $\pi_2\langle s, t\rangle \in \mathcal{N}$. $\dashv$

Corollary 1.2 *For $\diamond \in \{\to, +, \times\}$ and the same in both choices, we have*

$$\mathcal{M} \diamond \mathcal{N} \subseteq \mathcal{E}_\diamond(\mathcal{M}, \mathcal{N}).$$

Proof. We have to show that $\mathcal{M} \diamond \mathcal{N} \equiv \mathsf{cl}(\mathcal{I}_\diamond(\mathcal{M}, \mathcal{N})) \subseteq \mathcal{E}_\diamond(\mathcal{M}, \mathcal{N})$. But by the previous lemmas we have that $\mathcal{I}_\diamond(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{E}_\diamond(\mathcal{M}, \mathcal{N})$ and that $\mathcal{E}_\diamond(\mathcal{M}, \mathcal{N}) \in \mathsf{SAT}$ therefore by minimality of the closure we are done. $\dashv$

Proposition 1.7 (Saturated Sets Properties) *Assume $\mathcal{M}, \mathcal{N} \in \mathsf{SAT}$, then*

1. $\mathcal{M} \to \mathcal{N} \in \mathsf{SAT}$
2. *If $r \in \mathcal{M} \to \mathcal{N}$ and $s \in \mathcal{M}$ then $rs \in \mathcal{N}$.*
3. *If $t \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})$ then $\lambda x t \in \mathcal{M} \to \mathcal{N}$.*

4. $\mathcal{M} + \mathcal{N} \in \mathsf{SAT}$

5. *If* $t \in \mathcal{M}$ *then* $\mathsf{inl}\, t \in \mathcal{M} + \mathcal{N}$*.*

6. *If* $t \in \mathcal{N}$ *then* $\mathsf{inr}\, t \in \mathcal{M} + \mathcal{N}$*.*

7. *If* $r \in \mathcal{M} + \mathcal{N},\ s \in \mathsf{S}_x(\mathcal{M}, \mathcal{P}),\ t \in \mathsf{S}_y(\mathcal{N}, \mathcal{P})$ *then* $\mathsf{case}(r, x.s, y.t) \in \mathcal{P}$

8. $\mathcal{M} \times \mathcal{N} \in \mathsf{SAT}$

9. *If* $s \in \mathcal{M}$ *and* $t \in \mathcal{N}$ *then* $\langle s, t\rangle \in \mathcal{M} \times \mathcal{N}$

10. *If* $r \in \mathcal{M} \times \mathcal{N}$ *then* $\pi_1 r \in \mathcal{M}$ *and* $\pi_2 r \in \mathcal{N}$

Proof.

1. Clear.

2. Immediate from $\mathcal{M} \to \mathcal{N} \subseteq \mathcal{E}_{\to}(\mathcal{M}, \mathcal{N})$.

3. Take $t \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})$, this implies $\lambda x t \in \mathcal{I}_{\to}(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} \to \mathcal{N}$.

4. Clear.

5. $t \in \mathcal{M}$ implies $\mathsf{inl}\, t \in \mathcal{I}_{+}(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} + \mathcal{N}$.

6. $t \in \mathcal{N}$ implies $\mathsf{inr}\, t \in \mathcal{I}_{+}(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} + \mathcal{N}$.

7. Immediate from $\mathcal{M} + \mathcal{N} \subseteq \mathcal{E}_{+}(\mathcal{M}, \mathcal{N})$.

8. Clear.

9. $s \in \mathcal{M},\ t \in \mathcal{N}$ imply $\langle s, t\rangle \in \mathcal{I}_{\times}(\mathcal{M}, \mathcal{N}) \subseteq \mathcal{M} \times \mathcal{N}$.

10. Immediate from $\mathcal{M} \times \mathcal{N} \subseteq \mathcal{E}_{\times}(\mathcal{M}, \mathcal{N})$.

$\dashv$

Definition 1.19 *A candidate assignment is a finite set of pairs of the form* $\alpha : \mathcal{M}$ *where* α *is a type variable and* $\mathcal{M} \in \mathsf{SAT}$ *such that no type variable occurs twice. Candidate assignments are denoted with* Γ*, in the expression* $\Gamma, \alpha : \mathcal{M}$ *is understood that* $\alpha \notin \Gamma$*.*

Definition 1.20 (Strong Computability Predicates) *Given a type* ρ *and a candidate assigment* Γ *we define the saturated set of strongly computable terms* *with respect to ρ and Γ,denoted $\mathsf{SC}^{\rho}[\Gamma]$, as follows:*

$$
\begin{array}{rcl}
\mathsf{SC}^{\alpha}[\Gamma] & := & \left\{ \begin{array}{ll} \mathcal{M} & \textit{if } \alpha : \mathcal{M} \in \Gamma \\ \mathsf{SN} & \quad \textit{otherwise.} \end{array} \right. \\[2ex]
\mathsf{SC}^{\rho \to \sigma}[\Gamma] & := & \mathsf{SC}^{\rho}[\Gamma] \to \mathsf{SC}^{\sigma}[\Gamma] \\[2ex]
\mathsf{SC}^{\rho + \sigma}[\Gamma] & := & \mathsf{SC}^{\rho}[\Gamma] + \mathsf{SC}^{\sigma}[\Gamma] \\[2ex]
\mathsf{SC}^{\rho \times \sigma}[\Gamma] & := & \mathsf{SC}^{\rho}[\Gamma] \times \mathsf{SC}^{\sigma}[\Gamma] \\[2ex]
\mathsf{SC}^{\forall \alpha \rho}[\Gamma] & := & \bigcap_{\mathcal{M} \in \mathsf{SAT}} \mathsf{SC}^{\rho}[\Gamma, \alpha : \mathcal{M}]
\end{array}
$$

Lemma 1.5 (Coincidence) *If $\alpha \notin FV(\rho)$ then $\mathsf{SC}^{\rho}[\Gamma, \alpha : \mathcal{M}] = \mathsf{SC}^{\rho}[\Gamma]$.*
Proof. Induction on ρ.
If $\rho \equiv \beta \neq \alpha$ we have two possibilites, if $\beta : \mathcal{N} \in \Gamma$ then $\mathsf{SC}^{\beta}[\Gamma, \alpha : \mathcal{M}] = \mathcal{N} = \mathsf{SC}^{\beta}[\Gamma]$, otherwise $\mathsf{SC}^{\beta}[\Gamma, \alpha : \mathcal{M}] = \mathsf{SN} = \mathsf{SC}^{\beta}[\Gamma]$. For $\rho \equiv \forall \beta \sigma$, we can assume $\beta \notin \Gamma$ and $\alpha \neq \beta$, then $\mathsf{SC}^{\forall \beta \sigma}[\Gamma, \alpha : \mathcal{M}] = \bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\sigma}[\Gamma, \alpha : \mathcal{M}, \beta : \mathcal{N}]$ which by IH, as $\alpha \notin FV(\sigma)$, equals $\bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\sigma}[\Gamma, \beta : \mathcal{N}] = \mathsf{SC}^{\forall \beta \sigma}[\Gamma]$. ⊣

Lemma 1.6 (Substitution) $\mathsf{SC}^{\rho[\alpha := \sigma]}[\Gamma] = \mathsf{SC}^{\rho}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$.
Proof. Induction on ρ. If $\rho = \alpha$ then $\mathsf{SC}^{\alpha[\alpha := \sigma]}[\Gamma] = \mathsf{SC}^{\sigma}[\Gamma]$ which by definition is the same as $\mathsf{SC}^{\alpha}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$. If $\rho \equiv \beta \neq \alpha$ we have $\mathsf{SC}^{\beta[\alpha := \sigma]}[\Gamma] \equiv \mathsf{SC}^{\beta}[\Gamma]$ which by the coincidence lemma is the same as $\mathsf{SC}^{\beta}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$.
Case $\rho \equiv \forall \beta \tau$. We can assume $\beta \neq \alpha$ and $\beta \notin FV(\sigma)$. $\mathsf{SC}^{(\forall \beta \tau)[\alpha := \sigma]}[\Gamma] = \bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\tau[\alpha := \sigma]}[\Gamma, \beta : \mathcal{N}]$, which by IH equals $\bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\tau}[\Gamma, \beta : \mathcal{N}, \alpha : \mathsf{SC}^{\sigma}[\Gamma, \beta : \mathcal{N}]] = \bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma, \beta : \mathcal{N}], \beta : \mathcal{N}]$, which using the coincidence lemma ($\beta \notin FV(\sigma)$) simplifies to $\bigcap_{\mathcal{N} \in \mathsf{SAT}} \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma], \beta : \mathcal{N}]$. But this is exactly $\mathsf{SC}^{\forall \beta \tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$.
⊣

Lemma 1.7 (Main Lemma) *If $\boldsymbol{\Sigma} \rhd r : \rho$ with $\boldsymbol{\Sigma} = \{x_1 : \rho_1, \ldots, x_k : \rho_k\}$ and $s_i \in \mathsf{SC}^{\rho_i}[\Gamma]$, for $1 \leq i \leq k$, then $r[\vec{x} := \vec{s}\,] \in \mathsf{SC}^{\rho}[\Gamma]$.*
Proof. Induction on $\rhd$. Case $(\to I)$ Assume $\boldsymbol{\Sigma} \rhd \lambda x t : \rho \to \sigma$ from $\boldsymbol{\Sigma}, x : \rho \rhd t : \sigma$. Our goal is $(\lambda x t)[\vec{x} := \vec{s}] \in \mathsf{SC}^{\rho \to \sigma}[\Gamma]$, i.e., $\lambda x.t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\rho}[\Gamma] \to \mathsf{SC}^{\sigma}[\Gamma]$. Using the proposition 1.7, part 3, it suffices to show $t[\vec{x} := \vec{s}] \in \mathsf{S}_x(\mathsf{SC}^{\rho}[\Gamma], \mathsf{SC}^{\sigma}[\Gamma])$. Take $r \in \mathsf{SC}^{\rho}[\Gamma]$, we have to prove that $t[\vec{x} := \vec{s}][x := r] \in \mathsf{SC}^{\sigma}[\Gamma]$. The IH implies $t[\vec{x}, x := \vec{s}, r] \in \mathsf{SC}^{\sigma}[\Gamma]$, but we have $x \notin \vec{x}$ and w.l.o.g. also $x \notin FV(\vec{s})$ therefore $t[\vec{x}, x := \vec{x}, r] \equiv t[\vec{x} := \vec{s}][x := r]$ and we are done.
Case $(\forall I)$ Assume $\boldsymbol{\Sigma} \rhd t : \forall \alpha \tau$ from $\boldsymbol{\Sigma} \rhd t : \tau$ and $\alpha \notin FV(\boldsymbol{\Sigma})$. $s_i \in \mathsf{SC}^{\rho_i}[\Gamma]$ and $\alpha \notin FV(\rho_i)$ imply by the coincidence lemma $s_i \in \mathsf{SC}^{\rho_i}[\Gamma, \alpha : \mathcal{M}]$, which by IH implies $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathcal{M}]$ for all $\mathcal{M} \in \mathsf{SAT}$, i.e., $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\forall \alpha \tau}[\Gamma]$.
Case $(\forall E)$. Assume $\boldsymbol{\Sigma} \rhd t : \tau[\alpha := \sigma]$ from $\boldsymbol{\Sigma} \rhd t : \forall \alpha \tau$. By IH we have $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\forall \alpha \tau}[\Gamma]$ which implies $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathcal{M}]$ for all $\mathcal{M} \in \mathsf{SAT}$.

In particular we have $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$ which, using the substitution lemma, is the same as $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\tau[\alpha:=\sigma]}[\Gamma]$.

⊣

Proposition 1.8 *If* $\mathbb{\Sigma} \rhd r : \rho$ *then* $r \in \mathsf{SN}$.
Proof. Assume $\mathbb{\Sigma} = \{x_1 : \rho_1, \dots, x_k : \rho_k\}$. As the set of variables is contained in every saturated set we have $x_i \in \mathsf{SC}^{\rho_i}[\varnothing]$ therefore as $\mathbb{\Sigma} \rhd r : \rho$ the main lemma yields $r[\vec{x} := \vec{x}] \in \mathsf{SC}^{\rho}[\varnothing] \subseteq \mathsf{SN}$. Therefore $r \in \mathsf{SN}$. ⊣

Terms in SN are Strongly Normalizing

Definition 1.21 *The set* sn *of strongly normalizing terms is inductively defined as follows:*

$$\textit{If for every } r' \textit{ such that } r \to_\beta r' \textit{ we have } r' \in \mathsf{sn} \textit{ then } r \in \mathsf{sn}.$$

Lemma 1.8 sn *is the set of terms* r *such that there is no infinite* β*-reduction sequence starting in* r.
Proof. To prove that given a term $r \in \mathsf{sn}$ there is no infinite reduction sequence starting in r we simply do induction on $r \in \mathsf{sn}$. For the reverse inclusion use bar induction, i.e., show that $\{s | r \to_\beta^\star s\} \subseteq \mathsf{sn}$ by induction on $\to_\beta$. ⊣

Lemma 1.9 *Variables belong to* sn.
Proof. Clear ⊣

Lemma 1.10 *If* $E[x], s \in \mathsf{sn}$ *then* $E[x]s \in \mathsf{sn}$.
Proof. Main Induction on $E[x] \in \mathsf{sn}$, side induction on $s \in \mathsf{sn}$. ⊣

Lemma 1.11 *If* $r \in \mathsf{sn}$ *then* $\lambda x r \in \mathsf{sn}$.
Proof. Induction on $r \in \mathsf{sn}$. ⊣

Lemma 1.12 *If* $E[r[x := s]], s \in \mathsf{sn}$ *then* $E[(\lambda x r)s] \in \mathsf{sn}$.
Proof. Main Induction on $s \in \mathsf{sn}$, side induction on $E[r[x := s]] \in \mathsf{sn}$. ⊣

Lemma 1.13 *If* $r, s \in \mathsf{sn}$ *then* $\langle r, s\rangle \in \mathsf{sn}$.
Proof. Main Induction on $r \in \mathsf{sn}$, side induction on $s \in \mathsf{sn}$. ⊣

Lemma 1.14 *If* $E[x] \in \mathsf{sn}$ *then* $\pi_1(E[x]) \in \mathsf{sn}$ *and* $\pi_2(E[x]) \in \mathsf{sn}$.
Proof. Induction on $E[x] \in \mathsf{sn}$. ⊣

Lemma 1.15 *If* $E[r], s \in \mathsf{sn}$ *then* $E[\pi_1\langle r, s\rangle] \in \mathsf{sn}$
Proof. Main induction on $s \in \mathsf{sn}$, side induction on $E[r] \in \mathsf{sn}$. ⊣

Lemma 1.16 *If* $E[s], r \in \mathsf{sn}$ *then* $E[\pi_2\langle r, s\rangle] \in \mathsf{sn}$
Proof. Analogous to the previous lemma ⊣

Lemma 1.17 *If* $E[x], r, s \in \mathsf{sn}$ *then* $\mathsf{case}(E[x], y.r.z.s) \in \mathsf{sn}$.
Proof. Induction on $E[x], r, s \in \mathsf{sn}$. ⊣

Lemma 1.18 *If* $r \in \mathsf{sn}$ *then* $\mathsf{inl}\, r \in \mathsf{sn}$ *and* $\mathsf{inr}\, r \in \mathsf{sn}$.
Proof. Induction on $r \in \mathsf{sn}$. ⊣

Lemma 1.19 *If* $E[s[y := t]] \in \mathsf{sn}$ *and* $r \in \mathsf{sn}$ *then* $E[\mathsf{case}(\mathsf{inr}\, t, x.r, y.s)] \in \mathsf{sn}$
Proof. Main induction on $r \in \mathsf{sn}$, side induction on $E[s[y := t]] \in \mathsf{sn}$. ⊣

Lemma 1.20 *If* $E[r[x := t]] \in \mathsf{sn}$ *and* $s \in \mathsf{sn}$ *then* $E[\mathsf{case}(\mathsf{inl}\, t, x.r, y.s)] \in \mathsf{sn}$
Proof. Analogous to the previous lemma ⊣

Proposition 1.9 $\mathsf{SN} \subseteq \mathsf{sn}$
Proof. The above lemmas show that sn is closed under the defining rules of SN, therefore the claim follows by minimality of SN. ⊣

Proposition 1.10 $\mathsf{F}^{+,\times}$ *strongly normalizes.*
Proof. Immediate from propositions 1.8 and 1.9 ⊣

Corollary 1.3 F *is strongly normalizing.*

1.2.2 Adding Existential Types

Another useful extension of system F or of any other system treated in this work will be obtained by adding existential types. This is a not essential extension as existential types can be defined in system F.

Add to system F the following:

- Types: If α is a type variable and ρ is a type then $\exists\alpha\rho$ is a type.
- Terms: $\mathsf{pack}\, r$, $\mathsf{open}(r, x.s)$
- Typing Rules:
$$\frac{\Sigma \rhd r : \rho[\alpha := \sigma]}{\Sigma \rhd \mathsf{pack}\, r : \exists\alpha\rho}\ (\exists I)$$
$$\frac{\Sigma \rhd r : \exists\alpha\rho \quad \Sigma, z : \rho \rhd s : \sigma}{\Sigma \rhd \mathsf{open}(r, z.s) : \sigma}\ (\exists E)$$
The last rule with the proviso $\alpha \notin FV(\Sigma, \sigma)$.
- β-reduction:
$$\mathsf{open}(\mathsf{pack}\, r, z.s) \mapsto_\beta s[z := r]$$

The extension will be denoted with $\mathsf{F}^\exists$. Moreover given a type system T we denote with $\mathsf{T}^\exists$ the extension of T with existential types.
Strong normalization will be proved in the next subsection whereas subject reduction is again obtained by adapting the proof for $\mathsf{AF2}$..

1.2.3 On Embeddings

As we have seen in section 1.2.1 direct proofs of strong normalization are quite complicated. Fortunately we have a simpler technique to get such proofs which will be used frequently later in this work, namely the embedding of typed term rewrite systems. Here we give the definitions and justification of this technique.

Definition 1.22 (Embedding of Typed Term Rewrite Systems) *An embedding from a typed term rewrite system $\langle \mathcal{T}, \rightsquigarrow_{\mathcal{T}}, \rhd_{\mathcal{T}} \rangle$ into a typed term rewrite system $\langle \mathcal{T}^\star, \rightsquigarrow_{\mathcal{T}^\star}, \rhd_{\mathcal{T}^\star} \rangle$ is a function $(\cdot)' : \mathcal{T} \to \mathcal{T}^\star$ which assigns a term $t' \in \mathcal{T}^\star$ to every term $t \in \mathcal{T}$ and a type $\rho' \in \mathcal{T}^\star$ to every type $\rho \in \mathcal{T}$ such that*

- $x' := x$
- $(\cdot)'$ *is type-respecting, i.e. If $\Sigma \rhd_{\mathcal{T}} r : \rho$ then $\Sigma' \rhd_{\mathcal{T}^\star} r' : \rho'$.*
  where if $\Sigma = \{x_1 : \sigma_1, \ldots, x_k : \sigma_k\}$ then $\Sigma' = \{x_1 : \sigma_1', \ldots, x_k : \sigma_k'\}$
- $(\cdot)'$ *is reduction-preserving, i.e. If $s \rightsquigarrow_{\mathcal{T}} t$ then $s' \rightsquigarrow^{+}_{\mathcal{T}^\star} t'$. In words every reduction step in $\mathcal{T}$ is mapped into at least one reduction step in $\mathcal{T}^\star$.*

Proposition 1.11 (Inheritance of Strong Normalization) *Assume the typed term rewrite system $\mathcal{T}^\star$ strongly normalizes and $(\cdot)' : \mathcal{T} \to \mathcal{T}^\star$ is an embedding then $\mathcal{T}$ is strongly normalizing.*
Proof. Clear as an infinite reduction sequence in $\mathcal{T}$ would generate an infinite reduction sequence in $\mathcal{T}^\star$, which is absurd as $\mathcal{T}^\star$ strongly normalizes. ⊣

Strong Normalization for $\mathsf{F}^{\exists}$

We proof now the strong normalization of $\mathsf{F}^{\exists}$ via an embedding into system F.

The non-homomorphic rules of an embedding into system F are:

- Types:
  $$(\exists\alpha\rho)' := \forall\beta.(\forall\alpha.\rho' \to \beta) \to \beta$$
  where $\beta \notin FV(\rho, \alpha)$.
- Terms:
  $$(\mathsf{pack}\, r)' := \lambda x.xr'$$
  $$(\mathsf{open}(r, x.s))' := r'(\lambda z.s')$$

The following lemma will be needed to prove that the above function is really an embedding.

Lemma 1.21 *The following properties hold*

- $\rho[\alpha := \sigma]' = \rho'[\alpha := \sigma']$.
- $r[x := s]' = r'[x := s']$.

Proof. Induction on ρ and r respectively. ⊣

With help of the previous lemma the following is easy to proof.

Proposition 1.12 $\cdot' : \mathsf{F}^{\exists} \to \mathsf{F}$ *is an embedding.*

Finally using prop 1.11 we get

Corollary 1.4 $\mathsf{F}^{\exists}$ *strongly normalizes.*

1.3 Second Order Logic AF2

The basic logic that we will use is the system AF2 due to Leivant [Lei83] and Krivine [Kri93]. It is a natural deduction proof system for second-order logic with a proof trace mechanism by means of terms used as labelsfor formulas, called proof-terms. The main feature of the system is the inclusion of equational reasoning by means of second-order defined Leibniz' equality.

1.3.1 Definition of the System

Formulas are generated as follows:

$$A, B, C ::= X\vec{t} \mid A \to B \mid \forall x A \mid \forall X A$$

where x (X) is first-order (second-order) variable and in $X\vec{t}$, the arity of X is equal to the length of $\vec{t}$. The term system is a static one generated by

$$r, s, t ::= x \mid f\vec{t}$$

where f belongs to a given set of function symbols.

The sets $FV(t)$ and $FV(A)$ of free variables of t and A are defined as usual. Observe that in this case $FV(t)$ consists of all variables occurring in t.

On Substitution

Definition 1.23 *Given a term t, variables $\vec{x}$ and terms $\vec{s}$ we define the simultaneous substitution of $\vec{x}$ with $\vec{s}$ in t denoted $t[\vec{x} := \vec{s}]$ as follows:*

$$x[\vec{x} := \vec{s}] = \begin{cases} s_i & \textit{If } x \equiv x_i \\ \\ x & \textit{If } x \notin \vec{x} \end{cases}$$

$$(f\vec{t})[\vec{x} := \vec{s}] = f(\vec{t}[\vec{x} := \vec{s}])$$

Definition 1.24 *Given a formula A, variables $\vec{x}$ and terms $\vec{s}$ we define the substitution of $\vec{x}$ with $\vec{s}$ in A, denoted $A[\vec{x} := \vec{s}]$ as follows:*

$$(X\vec{t})[\vec{x} := \vec{s}] = X\vec{t}[\vec{x} := \vec{s}].$$

$$(A \rightarrow B)[\vec{x} := \vec{s}] = A[\vec{x} := \vec{s}] \rightarrow B[\vec{x} := \vec{s}]$$

$(\forall x A)[\vec{x} := \vec{s}] = \forall x.A[\vec{x} := \vec{s}]$, *always assuming* $x \notin \vec{x} \cup FV(\vec{s})$.

$$(\forall X A)[\vec{x} := \vec{s}] = \forall X.A[\vec{x} := \vec{s}]$$

The following concept provides an important tool to define sets in second-order logic.

Definition 1.25 *A comprehension predicate is an expression of the form*

$$\lambda\vec{y}F$$

where $\vec{y}$ are first-order variables and F is a formula. With calligraphic letters $\mathcal{F}, \mathcal{G}, \mathcal{H}, \ldots$, we denote the comprehension predicates generated by the formulas $F, G, H, \ldots$, respectively. The arity of $\lambda\vec{y}F$ is the length of $\vec{y}$.
Intuitively $\lambda\vec{y}F$ represents the set $\{\vec{t} \mid F[\vec{y} := \vec{t}\,]\}$, therefore $(\lambda\vec{y}.F)\vec{t}$ should be understood as $F[\vec{y} := \vec{t}\,]$. The set of free variables of $\lambda\vec{y}F$ is defined as $FV(\lambda\vec{y}F) := FV(F) \setminus \{\vec{y}\}$.
A predicate is either a second-order variable or a comprehension predicate.

Definition 1.26 *Given a formula A, variables $\vec{X}$ and predicates $\vec{\mathcal{F}}$ we define the substitution of $\vec{X}$ with $\vec{\mathcal{F}}$ in A, denoted $A[\vec{X} := \vec{\mathcal{F}}\,]$ as follows:*

$$(X\vec{t})[\vec{X} := \vec{\mathcal{F}}\,] = \begin{cases} \mathcal{F}_i\vec{t} & \text{If } X \equiv X_i \\ \\ X\vec{t} & \text{If } X \notin \vec{X} \end{cases}$$

$$(A \rightarrow B)[\vec{X} := \vec{\mathcal{F}}\,] = A[\vec{X} := \vec{\mathcal{F}}\,] \rightarrow B[\vec{X} := \vec{\mathcal{F}}\,]$$

$(\forall x A)[\vec{X} := \vec{\mathcal{F}}\,] = \forall x.A[\vec{X} := \vec{\mathcal{F}}\,]$, *always assuming* $x \notin FV(\vec{\mathcal{F}}\,)$.

$(\forall X A)[\vec{X} := \vec{\mathcal{F}}\,] = \forall X.A[\vec{X} := \vec{\mathcal{F}}\,]$, *always assuming* $X \notin \vec{X} \cup FV(\vec{\mathcal{F}}\,)$.

Lemma 1.22 (Substitution Properties) *The following properties hold:*

- *If $\vec{x} \notin \vec{y} \cup FV(\vec{s})$ then*

$$t[\vec{x} := \vec{r}][\vec{y} := \vec{s}] = t[\vec{y} := \vec{s}][\vec{x} := \vec{r}[\vec{y} := \vec{s}]] \quad (SwP1)$$

- *If $\vec{\beta} \notin \vec{\gamma} \cup FV(\vec{\zeta})$ then*

$$A[\vec{\beta} := \vec{\chi}][\vec{\gamma} := \vec{\zeta}] \;\equiv\; A[\vec{\gamma} := \vec{\zeta}][\vec{\beta} := \vec{\chi}[\vec{\gamma} := \vec{\zeta}]] \quad (SwP2)$$

where $\vec{\beta}, \vec{\gamma}$ can be first or second order variables and $\vec{\chi}, \vec{\zeta}$ are terms or comprehension predicates respectively, so that every substitution makes sense.

Proof. Induction on t and A respectively. ⊣

The particular feature of AF2 is the use of equations between terms $s = t$ defined in the next section. The judgments of the logic are of the form

$$\Gamma \vdash_{\mathbb{E}} t : A$$

where

- A is a formula.
- Γ is a given context of formulas of the form $\{x_1 : A_1, \dots, x_n : A_n\}$.
- $\mathbb{E}$ is a given context of equations of the form $\{s_1 = t_1, \dots, s_k = t_k\}$.
- t is a lambda-term encoding the derivation of A. Such terms are called proof-terms.

The relation $\Gamma \vdash_{\mathbb{E}} t : A$, read as "the formula A is derivable from the assumptions $\Gamma, \mathbb{E}$ and the term t is a code for such derivation", is inductively defined from

$$\Gamma, x : A \vdash_{\mathbb{E}} x : A \;\; (Var) \qquad \frac{s = t \in \mathbb{E}}{\Gamma \vdash_{\mathbb{E}} s = t} \; (start)$$

as follows:

$$\frac{\Gamma, x : A \vdash_{\mathbb{E}} r : B}{\Gamma \vdash_{\mathbb{E}} \lambda xr : A \to B} \; (\to I) \quad \frac{\Gamma \vdash_{\mathbb{E}} r : A \to B \quad \Gamma \vdash_{\mathbb{E}} s : A}{\Gamma \vdash_{\mathbb{E}} rs : B} \; (\to E)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A}{\Gamma \vdash_{\mathbb{E}} t : \forall x A} \; (\forall I) \quad \frac{\Gamma \vdash_{\mathbb{E}} t : \forall x A}{\Gamma \vdash_{\mathbb{E}} t : A[x := s]} \; (\forall E)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A}{\Gamma \vdash_{\mathbb{E}} t : \forall X A} \; (\forall^2 I) \quad \frac{\Gamma \vdash_{\mathbb{E}} t : \forall X A}{\Gamma \vdash_{\mathbb{E}} t : A[X := \mathcal{F}]} \; (\forall^2 E)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A[x := s] \quad \Gamma \vdash_{\mathbb{E}} s = t}{\Gamma \vdash_{\mathbb{E}} r : A[x := t]} \; (Eq)$$

Important remarks are:

- In the rule $(\forall I)$, $x \notin FV(\Gamma, \mathbb{E})$.
- In the rule $(\forall^2 I)$, $X \notin FV(\Gamma)$ (Observe that $X \notin FV(\mathbb{E})$ always holds).
- In the rule (Eq), $\Gamma \vdash_{\mathbb{E}} s = t$ means nothing but a derivation with the rules being defined with the difference that we get rid of the proof-terms. Indeed we could isolate the context of equalities and perform only derivations of the form $\mathbb{E} \vdash s = t$ but in extensions of the system needed later this is not possible anymore, therefore we prefer this general formulation.

- ○ Although we make no syntactic distinction between object and proof-term variables we consider both sets as disjunct.
- ○ From now on we will make explicit the context $\mathbb{E}$ only if neccesary, but usually we will only write $\vdash$ instead of $\vdash_{\mathbb{E}}$.
- ○ Rules like (Eq) and the four rules for $\forall, \forall^2$ whose application is not reflected in the proof-term system are called *non-traceable.*, in other case a rule is called *traceable*.

The proof reduction is given by the following β-reduction rule between proof-terms:

$$(\lambda x r)s \quad \mapsto_\beta \quad r[x := s]$$

To see the expressive power of AF2 we define natural numbers and streams.

Natural Numbers in AF2

Given a constant symbol 0 and a unary function symbol s, we define the unary predicate of natural numbers as:

$$\mathbb{N} := \lambda z.\forall X.X0 \to (\forall x.Xx \to Xsx) \to Xz$$

It is easy to see that $\vdash \widetilde{0} : \mathbb{N}0$ and $\vdash \widetilde{s} : \forall x.\mathbb{N}x \to \mathbb{N}sx$. where $\widetilde{0} := \lambda x\lambda f.x$ and $\widetilde{s} : \lambda n\lambda x\lambda f.f(nxf)$.

Streams in AF2

Given unary function symbols $\mathsf{head}, \mathsf{tail}$, we define the unary predicate of streams of elements of the predicate $\mathcal{A}$ as:

$$\mathcal{S}_{\mathcal{A}} := \quad \lambda u.\forall Z.\Big(\forall X.(\forall x.Xx \to \mathcal{A}\,\mathsf{head}\,x) \to (\forall x.Xx \to X\,\mathsf{tail}\,x) \to \forall x.Xx \to Zx\Big) \to Zu$$

We can see that $\vdash \widetilde{\mathsf{head}} : \forall x.\mathcal{S}_{\mathcal{A}}x \to \mathcal{A}\,\mathsf{head}\,x$ and $\vdash \widetilde{\mathsf{tail}} : \forall x.\mathcal{S}_{\mathcal{A}}x \to \mathcal{S}_{\mathcal{A}}\,\mathsf{tail}\,x$, where $\widetilde{\mathsf{head}} := \lambda s.s(\lambda h\lambda t\lambda x.hx)$ and $\widetilde{\mathsf{tail}} := \lambda s.s(\lambda h\lambda t\lambda x\lambda f.fht(tx))$.

On Leibniz' Equality

The particular feature of AF2 is the use of Leibniz' equality, which is defined for given terms s, t as:

$$s = t := \forall X.Xs \to Xt$$

A formula of the form $s = t$ will be called equation.

The following derived rules will be very useful when handling equations:

$$\frac{}{\Gamma \vdash_{\mathbb{E}} t = t}\ (refl)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} s = t}{\Gamma \vdash_{\mathbb{E}} t = s}\ (symm) \quad \frac{\Gamma \vdash_{\mathbb{E}} r = s \quad \Gamma \vdash_{\mathbb{E}} s = t}{\Gamma \vdash_{\mathbb{E}} r = t}\ (trans)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} s_i = t_i,\ 1 \leq i \leq k}{\Gamma \vdash_{\mathbb{E}} f\vec{s} = f\vec{t}}\ (comp)$$

Proposition 1.13 *The above rules for equational reasoning can be derived in* AF2.
Proof. We derive each rule

- $(refl)$. Clearly $\Gamma \vdash_{\mathbb{E}} \forall X.Xx \rightarrow Xx$.
- $(trans)$. It suffices to show

  $$\Gamma, \forall X.Xr \rightarrow Xs, \forall X.Xs \rightarrow Xt \vdash_{\mathbb{E}} \forall X.Xr \rightarrow Xt,$$

  which is clear.
- $(symm)$. It suffices to show $\Gamma \vdash_{\mathbb{E}} s = t \rightarrow t = s$. The goal is then

  $$\Gamma, \forall X.Xs \rightarrow Xt \vdash_{\mathbb{E}} t = s.$$

  We have by $(\forall E)$

  $$\Gamma, \forall X.Xs \rightarrow Xt \vdash_{\mathbb{E}} (Xs \rightarrow Xt)[X := \lambda z.z = s],$$

  i.e.,

  $$\Gamma, \forall X.Xs \rightarrow Xt \vdash_{\mathbb{E}} s = s \rightarrow t = s$$

  Finally using $(refl)$ we can eliminate the implication getting

  $$\Gamma, \forall X.Xs \rightarrow Xt \vdash_{\mathbb{E}} t = s$$

  which was the goal.
- $(comp)$. Assume $\Gamma \vdash_{\mathbb{E}} s_i = t_i$ for $1 \leq i \leq k$. In particular we have $\Gamma \vdash_{\mathbb{E}} s_1 = t_1$, which implies $\Gamma \vdash_{\mathbb{E}} (Xs_1 \rightarrow Xt_1)[X := \lambda z.Xfzs_2 \ldots s_k]$, that is $\Gamma \vdash_{\mathbb{E}} Xf\vec{s} \rightarrow Xft_1s_2 \ldots s_k$, which can be rewritten as

  $$\Gamma \vdash_{\mathbb{E}} (Xf\vec{s} \rightarrow Xft_1z_2 \ldots z_k)[z_2 := s_2] \ldots [z_k := s_k]$$

  Therefore as the $\vec{z}$ are fresh variables then after applying the rule (Eq) with $\Gamma \vdash_{\mathbb{E}} s_j = t_j$ for $2 \leq j \leq k$ and permuting some substitutions we get

  $$\Gamma \vdash_{\mathbb{E}} (Xf\vec{s} \rightarrow Xft_1z_2 \ldots z_k)[z_2 := t_2] \ldots [z_k := t_k]$$

i.e.,

$$\Gamma \vdash_{\mathbb{E}} Xf\vec{s} \rightarrow Xft_1t_2 \ldots t_k$$

Finally by $(\forall^2 I)$ as $X \notin FV(\Gamma)$, we get $\Gamma \vdash_{\mathbb{E}} \forall X.Xf\vec{s} \rightarrow Xf\vec{t}$, which is the same as $\Gamma \vdash_{\mathbb{E}} f\vec{s} = f\vec{t}$. ⊣

Subject Reduction

This important property was proved in [Kri93].

1.3.2 Strong Normalization of AF2

The logic AF2 considered as a term rewrite system $\langle \mathsf{AF2}, \rightarrow_\beta, \vdash \rangle$ will be embedded into the strongly normalizing system $\langle \mathsf{F}, \rightarrow_\beta, \triangleright \rangle$.

The embedding will be the first-order forgetful map on formulas, defined as:

$$\begin{array}{rcl} r' & := & r \\ (X\vec{t})' & := & X \\ (A \to B)' & := & A' \to B' \\ (\forall x A)' & := & A' \\ (\forall X A)' & := & \forall X.A' \end{array}$$

where on the right-hand side the X is a type variable with the same name as the predicate variable X on the left-hand side, which can be assumed w.l.o.g. Observe that the embedding in proof-terms is the identity.

To prove that we really have an embedding we need the following

Lemma 1.23 *The following properties hold,*

- $A[\vec{x} := \vec{t}]' = A'$
- $A[X := \mathcal{F}]' = A'[X := \mathcal{F}']$, *where* $(\lambda\vec{y}F)' := \lambda\vec{y}.F'$.

Proof. Induction on A $\dashv$

The following two lemmas prove that we have an embedding.

Lemma 1.24 *If* $\Gamma \vdash_{\mathbb{E}} t : A$ *then* $\Gamma' \triangleright t : A'$.
Proof. Induction on $\vdash$. Observe that an application of the rules $(\forall I), (\forall E), (Eq)$ dissapear in system F. $\dashv$

Lemma 1.25 *If* $r \rightarrow_\beta^{\mathsf{AF2}} s$ *then* $r \rightarrow_\beta^{\mathsf{F}} s$.
Proof. Trivial as the embedding on terms is the identity. $\dashv$

Proposition 1.14 AF2 *strongly normalizes.*
Proof. Immediate from prop 1.11 and lemmas 1.24 and 1.25. $\dashv$

1.3.3 Adding Conjunctions and Disjunctions

Although disjunction and conjunction can be defined within AF2 we prefer to have them as primitives getting a system $\mathsf{AF2}^{\wedge,\vee}$.

The additional inference rules are:

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A \quad \Gamma \vdash_{\mathbb{E}} s : B}{\Gamma \vdash_{\mathbb{E}} \langle r, s \rangle : A \wedge B} (\wedge I) \quad \frac{\Gamma \vdash_{\mathbb{E}} r : A \wedge B}{\Gamma \vdash_{\mathbb{E}} \pi_1 r : A} (\wedge_1 E) \quad \frac{\Gamma \vdash_{\mathbb{E}} r : A \wedge B}{\Gamma \vdash_{\mathbb{E}} \pi_2 r : B} (\wedge_2 E)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} s : A}{\Gamma \vdash_{\mathbb{E}} \mathsf{inl}\, s : A \vee B}\,(\vee_L I) \quad \frac{\Gamma \vdash_{\mathbb{E}} s : B}{\Gamma \vdash_{\mathbb{E}} \mathsf{inr}\, s : A \vee B}\,(\vee_R I)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A \vee B \quad \Gamma, y : A \vdash_{\mathbb{E}} s : C \quad \Gamma, z : B \vdash_{\mathbb{E}} t : C}{\Gamma \vdash_{\mathbb{E}} \mathsf{case}(r, y.s, z.t) : C}\,(\vee E)$$

The additional proof-reduction rules are given by:

$$\begin{array}{rcl} \mathsf{case}(\mathsf{inl}\, r, x.s, y.t) & \mapsto_\beta & s[x := r] \\ \mathsf{case}(\mathsf{inr}\, r, x.s, y.t) & \mapsto_\beta & t[y := r] \\ \pi_1\langle r, s\rangle & \mapsto_\beta & r \\ \pi_2\langle r, s\rangle & \mapsto_\beta & s \end{array}$$

This system is also strongly normalizing, suffices to extend the embedding for AF2 as follows:

$$\begin{array}{rcl} (A \wedge B)' & := & A' \times B' \\ (A \vee B)' & := & A' + B' \end{array}$$

Lemmas 1.24,1.25 are still valid, therefore we have an embedding from $\mathsf{AF2}^{\wedge,\vee}$ into $\mathsf{F}^{+,\times}$, which by proposition 1.10 strongly normalizes.

Subject Reduction

It can be proven by extending the proof for AF2.

Ein Bier, das macht den Durst erst schön,
Drum nehmt das Glas und trinket!
Wie herrlich ist es anzusehn,
Wenn golden im Glase es blinket!

Deutsches Trinklied

2

Extensions of System F with Monotone (Co)inductive Types

In [Mat98, Mat99] Matthes presents several extensions of system F with inductive and coinductive types in Church-style. We take the basic ideas of that work and present some extensions of system F with monotone and clausular (co)inductive types in Curry-style, which model the (co)iteration/(co)recursion principles given in section 1.1.

2.1 From Categories to Types

Let us adopt an informal categorical view of our typable term language, the types will be objects of a category $\mathcal{C}$, such categories and its features are well-known, see for example [Cro93], here we only assume its existence, whereas the morphisms will be functions (terms) from one type to another and composition will be the usual function composition that is, if $f : \sigma \to \rho$ and $g : \rho \to \tau$ then we set $g \circ f := \lambda z.g(fz)$ and get $g \circ f : \sigma \to \tau$.
A functor $T : \mathcal{C} \to \mathcal{C}$ is then a transformation between types. We are specially interested in functors obtained by abstracting type variables, i.e., functors of the form $\lambda\alpha\rho$ where $(\lambda\alpha\rho)\sigma$ means $\rho[\alpha := \sigma]$. Such an abstraction is not immediate a functor because we only know its action on objects (types) but not on morphisms. To ensure that $\lambda\alpha\rho$ behaves really as a functor, specifically to ensure a functorial action on morphisms, the syntactical restriction of α being positive in ρ is usually required —with this proviso there is a canonical definition of what is the action of such functors on morphisms. In our treatment we prefer to follow [Mat98, Mat99] and use full monotonicity instead of positivity:

the functoriality of $\lambda\alpha\rho$ on morphisms is represented internally by means of a term $m : \rho\,\mathsf{mon}\,\alpha$ in a given context, where its type, defined as

$$\rho\,\mathsf{mon}\,\alpha := \forall\alpha\forall\beta.(\alpha \to \beta) \to \rho \to \rho[\alpha := \beta],$$

expresses the fact that $\lambda\alpha\rho$ is monotone (covariant) with respect to α. Such terms are called *monotonicity witnesses.*

Therefore a functor in this framework is a pair $\langle\lambda\alpha\rho, m\rangle$ where m is a term of type $\rho\,\mathsf{mon}\,\alpha$ (in a given context). This way of defining functors is reminiscent of the way functors are defined in some functional programming languages like Haskell, where this concept is captured by the following class definition:

```
class  Functor f  where
    fmap :: (a -> b) -> f a -> f b
```

Therefore a functor is not only a function `f` between categories but a pair composed of a function `f` and a mapping `fmap` who plays the role of the functor on morphisms.

The reader will confirm later that all usual examples of coinductive types are positive. What are then the advantages of using full monotonicity ? Two satisfactory answers are:

- Specific monotonicity witnesses are not involved in proofs, we can even have hypothetical monotonicity, i.e. just an additional assumption $x : \rho\,\mathsf{mon}\,\alpha$ in our context. Therefore the generality of our approach simplifies proofs.

- For higher-order systems there is no fixed concept of positivity. With full monotonicity we can generalize directly the systems presented in this work. Moreover, sometimes different witnesses are useful for programming, see the example on power list reverse in [AMU04].

We have now a fixed definition of functors in type systems, the next step is to represent initial (final) algebras (coalgebras).

2.1.1 Representing (Co)algebras

Representing Initial Algebras

Given a functor $\langle\lambda\alpha\rho, m\rangle$ we denote with $\langle\mu\alpha\rho, \mathsf{in}\rangle$ the (weak) initial algebra of $\lambda\alpha\rho$. The universal property of the universal algebra which corresponds to iteration is represented by the following diagram:

$$\begin{tikzcd}
\rho[\alpha := \mu\alpha\rho] \arrow[r, "\mathsf{in}"] \arrow[d, "m(\mathsf{It}_s)"'] & \mu\alpha\rho \arrow[d, "\mathsf{It}_s"] \\
\rho[\alpha := \sigma] \arrow[r, "s"] & \sigma
\end{tikzcd}$$

which generates the principle of iteration, corresponding to equation (1.1):

$$\mathsf{It}_s \circ \mathsf{in} = s \circ m(\mathsf{It}_s) \tag{2.1}$$

Moreover as the initial algebra $\langle \mu\alpha\rho, \mathsf{in}\rangle$ is recursive the principle of primitive recursion holds:

$$\begin{tikzcd}
\rho[\alpha := \mu\alpha\rho] \arrow[r, "\mathsf{in}"] \arrow[d, "m(\langle \mathsf{Id}, \mathsf{Rec}_s\rangle)"'] & \mu\alpha\rho \arrow[d, "\mathsf{Rec}_s"] \\
\rho[\alpha := \mu\alpha\rho \times \sigma] \arrow[r, "s"] & \sigma
\end{tikzcd}$$

which generates the principle of primitive recursion, corresponding to equation (1.7)

$$\mathsf{Rec}_s \circ \mathsf{in} = s \circ m(\langle \mathsf{Id}, \mathsf{Rec}_s\rangle) \tag{2.2}$$

We can only state the existence of the morphisms $\mathsf{It}, \mathsf{Rec}$, getting only weak algebras, because to model uniqueness would cause some technical problems later. Therefore we cannot get a full inverse in^{-1}. However based on the proof of proposition 1.1 we can get a morphism $\mathsf{in}^{-1} : \mu\alpha\rho \to \rho[\alpha := \mu\alpha\rho]$ such that the principle of inductive inversion holds:

$$\mathsf{in}^{-1} \circ \mathsf{in} = m(\mathsf{Id}). \tag{2.3}$$

Now we are able to develop the formal extension in system F. We will follow a natural deduction approach, so instead of constants $\mathsf{in}, \mathsf{in}^{-1}, \mathsf{It}, \mathsf{Rec}$ we will have a unary term constructor $\mathsf{in}\,\cdot$, a binary constructor $\mathsf{in}^{-1}(\cdot,\cdot)$ and ternary constructors $\mathsf{It}(\cdot,\cdot,\cdot), \mathsf{Rec}(\cdot,\cdot,\cdot)$.

The morphisms are represented as follows:

$$\begin{array}{rcl}
\mathsf{in} & \rightsquigarrow & \lambda z.\,\mathsf{in}\, z \\
\mathsf{in}^{-1} & \rightsquigarrow & \lambda z.\,\mathsf{in}^{-1}(m, z) \\
\mathsf{It}_s & \rightsquigarrow & \lambda z.\mathsf{It}(m, s, z) \\
\mathsf{Rec}_s & \rightsquigarrow & \lambda z.\mathsf{Rec}(m, s, z)
\end{array}$$

To represent the induction and inversion principles we do not use an exact correspondence with the equations above but we apply an argument t to both sides of the equation, we do so for all systems in this work.

The Iteration Principle

Equation (2.1) becomes

$$\mathsf{It}(m, s, \mathsf{in}\, t) \;=\; s\Big(m\big(\lambda x.\mathsf{It}(m, s, x)\big)t\Big)$$

The Primitive Recursion Principle

Equation (2.2) becomes:

$$\mathsf{Rec}(m, s, \mathsf{in}\, t) = s\Big(m\Big(\langle \mathsf{Id}, \lambda z.\mathsf{Rec}(m, s, z)\rangle\Big)t\Big)$$

The Inductive Inversion Principle

Equation (2.3) becomes:

$$\mathsf{in}^{-\mathbf{1}}(m, \mathsf{in}\, t) = m(\lambda z.z)t$$

Representing Final Coalgebras

Dually to the treatment on the previous section given a functor $\langle \lambda\alpha\rho, m\rangle$ the pair $\langle \nu\alpha\rho, \mathsf{out}\rangle$ represents the (weak) final coalgebra of $\lambda\alpha\rho$. Here we state the formal representations together with the corresponding diagrams.

The morphisms are represented as follows:

$$\begin{array}{rcl} \mathsf{out} & \rightsquigarrow & \lambda z.\, \mathsf{out}\, z \\ \mathsf{out}^{-1} & \rightsquigarrow & \lambda z.\, \mathsf{out}^{-1}(m, z) \\ \mathsf{CoIt}_s & \rightsquigarrow & \lambda z.\mathsf{CoIt}(m, s, z) \\ \mathsf{CoRec}_s & \rightsquigarrow & \lambda z.\mathsf{CoRec}(m, s, z) \end{array}$$

The Coiteration Principle

The coiteration principle is represented by the following diagram:

$$\begin{tikzcd}
\rho[\alpha := \sigma] \arrow[d, "m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)"'] & \sigma \arrow[l, "s"] \arrow[d, "\lambda z.\mathsf{CoIt}(m, s, z)"] \\
\rho[\alpha := \nu\alpha\rho] & \nu\alpha\rho \arrow[l, "\lambda z.\, \mathsf{out}\, z"]
\end{tikzcd}$$

which generates the equality:

$$\mathsf{out}\,\mathsf{CoIt}(m,s,t) = m\big(\lambda z.\mathsf{CoIt}(m,s,z)\big)(st)$$

corresponding to equation (1.2).

The Primitive Corecursion Principle

The corecursion principle is represented by the following diagram:

$$\begin{tikzcd}
\rho[\alpha := \nu\alpha\rho + \sigma] \arrow[d, "m\big({[\mathsf{Id}, \lambda x.\mathsf{CoRec}(m,s,x)]}\big)"'] & \sigma \arrow[l, "s"'] \arrow[d, "\lambda z.\mathsf{CoRec}(m,s,z)"] \\
\rho[\alpha := \nu\alpha\rho] & \nu\alpha\rho \arrow[l, "\lambda z.\,\mathsf{out}\,z"]
\end{tikzcd}$$

which generates the equality:

$$\mathsf{out}\,\mathsf{CoRec}(m,s,t) = m\big([\mathsf{Id}, \lambda x.\mathsf{CoRec}(m,s,x)]\big)(st)$$

corresponding to equation (1.8)

The Coinductive Inversion Principle

The equation

$$\mathsf{out}\,\mathsf{out}^{-1}(m,t) = m(\lambda z.z)t$$

representing coinductive inversion is obtained from the dual equation to (2.3)

2.1.2 Representing Dialgebras

The morphisms of dialgebras are represented as follows, for $1 \leq i \leq k$:

$$\begin{array}{rcl}
\mathsf{in}_{k,i} & \rightsquigarrow & \lambda z.\,\mathsf{in}_{k,i}\,z \\
\mathsf{It}^k_s & \rightsquigarrow & \lambda z.\mathsf{It}_k(\vec{m},\vec{s},z) \\
\mathsf{Rec}^k_s & \rightsquigarrow & \lambda z.\mathsf{Rec}_k(\vec{m},\vec{s},z) \\
\mathsf{out}_{k,i} & \rightsquigarrow & \lambda z.\,\mathsf{out}_{k,i}\,z \\
\mathsf{CoIt}^k_s & \rightsquigarrow & \lambda z.\mathsf{CoIt}_k(\vec{m},\vec{s},z) \\
\mathsf{CoRec}^k_s & \rightsquigarrow & \lambda z.\mathsf{CoRec}_k(\vec{m},\vec{s},z)
\end{array}$$

The (co)inductive principles become:

○ Iteration

$$\mathsf{It}_k(\vec{m}, \vec{s}, \mathsf{in}_{k,i}\, t) \;=\; s_i\Big(m_i\big(\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x)\big)t\Big)$$

corresponding to equation (1.18)

- ○ Primitive Recursion:

$$\mathsf{Rec}_k(\vec{m}, \vec{s}, \mathsf{in}_{k,i}\, t) \;=\; s_i\Big(m_i\Big(\langle \mathsf{Id}, \lambda z.\mathsf{Rec}_k(\vec{m}, \vec{s}, z)\rangle\Big)t\Big)$$

corresponding to equation (1.19)

- ○ Coiteration

$$\mathsf{out}_{k,i}\, \mathsf{CoIt}_k(\vec{m}, \vec{s}, t) \;=\; m_i\Big(\lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z)\Big)(s_i t)$$

corresponding to equation (1.15)

- ○ Primitive Corecursion

$$\mathsf{out}_{k,i}\, \mathsf{CoRec}_k(\vec{m}, \vec{s}, t) \;=\; m_i\Big([\mathsf{Id}, \lambda z.\mathsf{CoRec}_k(\vec{m}, \vec{s}, z)]\Big)(s_i t)$$

corresponding to equation (1.16)

A representation of the (co)inductive inversion principles will be discussed in section 2.3.1.

Representing M-dialgebras

The morphisms of M-dialgebras are represented as follows:

$$\begin{array}{rcl} \mathsf{MIt}^k_s & \rightsquigarrow & \lambda z.\mathsf{MIt}_k\vec{s}\,z \\ \mathsf{MRec}^k_s & \rightsquigarrow & \lambda z.\mathsf{MRec}_k\vec{s}\,z \\ \mathsf{MCoIt}^k_s & \rightsquigarrow & \lambda z.\mathsf{MCoIt}_k\vec{s}\,z \\ \mathsf{MCoRec}^k_s & \rightsquigarrow & \lambda z.\mathsf{MCoRec}_k\vec{s}\,z \end{array}$$

The principles are:

- ○ Mendler-Style Iteration. Equation (1.21) becomes

$$\mathsf{MIt}_k\vec{s}(\mathsf{in}_{k,i}\, r) = s_i\big(\mathsf{MIt}_k\vec{s}\big)r$$

- ○ Mendler-Style Recursion. Equation (1.22) becomes

$$\mathsf{MRec}_k\vec{s}(\mathsf{in}_{k,i}\, r) = s_i(\lambda yy)\big(\mathsf{MRec}_k\vec{s}\big)r$$

- ○ Mendler-Style Coiteration. Equation (1.23) becomes

$$\mathsf{out}_{k,i}(\mathsf{MCoIt}_k\vec{s}\,r) = s_i\big(\mathsf{MCoIt}_k\vec{s}\big)r$$

- ○ Mendler-Style Corecursion. Equation (1.24) becomes

$$\mathsf{out}_{k,i}(\mathsf{MCoRec}_k\vec{s}\,r) = s_i(\lambda yy)\big(\mathsf{MCoRec}_k\vec{s}\big)r$$

In the next sections we add the previous concepts to system F getting extensions with monotone (co)inductive types.

2.2 The System MICT

This is our basic extension with traditional (i.e. not clasular) (co)inductive types and conventional (co)induction principles taken from section 2.1.1. The resulting term rewrite system is called MICT a system of Monotone Inductive and Coinductive Types.

2.2.1 Definition of the System

We add the following to system $\mathsf{F}^{+,\times}$:

- ○ If α is a type variable and ρ is a type then $\mu\alpha\rho$ and $\nu\alpha\rho$ are types.
- ○ If m, r, s, t are terms then

  $$\begin{array}{c}\mathsf{It}(m,s,t), \mathsf{Rec}(m,s,t), \mathsf{in}\, t, \mathsf{in}^{-1}(m,t) \\ \mathsf{CoIt}(m,s,t), \mathsf{CoRec}(m,s,t), \mathsf{out}\, t, \mathsf{out}^{-1}(m,t)\end{array}$$

  are terms.

We add eight typing rules for inductive and coinductive types:

$$\frac{\Sigma \rhd t : \rho[\alpha := \mu\alpha\rho]}{\Sigma \rhd \mathsf{in}\, t : \mu\alpha\rho}\ (\mu I)$$

$$\frac{\begin{array}{c}\Sigma \rhd t : \mu\alpha\rho \\ \Sigma \rhd m : \rho\, \mathsf{mon}\, \alpha\end{array}}{\Sigma \rhd \mathsf{in}^{-1}(m,t) : \rho[\alpha := \mu\alpha\rho]}\ (\mu E^i)$$

$$\frac{\begin{array}{l}\Sigma \rhd t : \mu\alpha\rho \\ \Sigma \rhd m : \rho\, \mathsf{mon}\, \alpha \\ \Sigma \rhd s : \rho[\alpha := \sigma] \to \sigma\end{array}}{\Sigma \rhd \mathsf{It}(m,s,t) : \sigma}\ (\mu E)$$

$$\frac{\begin{array}{l}\Sigma \rhd t : \mu\alpha\rho \\ \Sigma \rhd m : \rho\, \mathsf{mon}\, \alpha \\ \Sigma \rhd s : \rho[\alpha := \mu\alpha\rho \times \sigma] \to \sigma\end{array}}{\Sigma \rhd \mathsf{Rec}(m,s,t) : \sigma}\ (\mu E^+)$$

$$\frac{\begin{array}{l}\Sigma \rhd s : \sigma \to \rho[\alpha := \sigma] \\ \Sigma \rhd m : \rho\, \mathsf{mon}\, \alpha \\ \Sigma \rhd t : \sigma\end{array}}{\Sigma \rhd \mathsf{CoIt}(m,s,t) : \nu\alpha\rho}\ (\nu I)$$

$$\frac{\begin{array}{l}\Sigma \rhd s : \sigma \to \rho[\alpha := \nu\alpha\rho + \sigma] \\ \Sigma \rhd m : \rho\, \mathsf{mon}\, \alpha \\ \Sigma \rhd t : \sigma\end{array}}{\Sigma \rhd \mathsf{CoRec}(m,s,t) : \nu\alpha\rho}\ (\nu I^+)$$

$$\frac{\begin{array}{c}\Sigma \rhd t : \rho[\alpha := \nu\alpha\rho] \\ \Sigma \rhd m : \rho \,\mathsf{mon}\, \alpha\end{array}}{\Sigma \rhd \mathsf{out}^{-1}(m, t) : \nu\alpha\rho}\ (\nu I^i)$$

$$\frac{\Sigma \rhd r : \nu\alpha\rho}{\Sigma \rhd \mathsf{out}\, r : \rho[\alpha := \nu\alpha\rho]}\ (\nu E)$$

Finally the equalities given in section 2.1.1 are added to the system as β-reduction rules:

$$\begin{array}{rcl}
\mathsf{It}(m, s, \mathsf{in}\, t) & \mapsto_\beta & s\Big(m\big(\lambda x.\mathsf{It}(m, s, x)\big)t\Big) \\
\mathsf{Rec}(m, s, \mathsf{in}\, t) & \mapsto_\beta & s\Big(m\Big(\langle \mathsf{Id}, \lambda z.\mathsf{Rec}(m, s, z)\rangle\Big)t\Big) \\
\mathsf{in}^{-1}(m, \mathsf{in}\, t) & \mapsto_\beta & m(\lambda z.z)t \\
\mathsf{out}\,\mathsf{CoIt}(m, s, t) & \mapsto_\beta & m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)(st) \\
\mathsf{out}\,\mathsf{CoRec}(m, s, t) & \mapsto_\beta & m\big([\mathsf{Id}, \lambda x.\mathsf{CoRec}(m, s, x)]\big)(st) \\
\mathsf{out}\,\mathsf{out}^{-1}(m, t) & \mapsto_\beta & m(\lambda z.z)t
\end{array}$$

where for given $f : \rho \to \tau, g : \sigma \to \tau$ we define $[f, g] : \rho + \sigma \to \tau$ as

$$[f, g] := \lambda z.\mathsf{case}(z, x.fx, y.gy).$$

Analogously for $f : \tau \to \rho, g : \tau \to \sigma$, $\langle f, g\rangle : \tau \to \rho \times \sigma$ is defined as

$$\langle f, g\rangle := \lambda z.\langle fz, gz\rangle.$$

Proposition 2.1 (Subject Reduction) *If $\Sigma \rhd r : \rho$ and $r \to_\beta s$ then $\Sigma \rhd s : \sigma$*

Proof. This property can be proved with the same method of section 4.1.3. ⊣

The Natural Numbers in MICT

The natural numbers are represented in MICT as follows:

$$\mathsf{nat} := \mu\alpha.1 + \alpha$$

where 1 is the unit type defined as $1 := \forall\alpha.\alpha \to \alpha$ which has only one inhabitant, namely $\star := \lambda xx$. This type generates a constructor $\mathbb{C} := \lambda x.\,\mathsf{in}\, x$ such that

$$\rhd\ \mathbb{C} : 1 + \mathsf{nat} \to \mathsf{nat}$$

The usual constructors for the natural numbers are encoded in the constructor $\mathbb{C}$, and are defined as

$$0 := \mathbb{C}(\mathsf{inl}\,\star) \qquad s := \lambda x.\mathbb{C}(\mathsf{inr}\, x)$$

Observe that we have to work with injections.

Streams in MICT

Given a type ρ the type of streams (infinite lists) of elements of ρ is defined in MICT as follows:

$$\mathsf{stream}(\rho) := \nu\alpha.\rho \times \alpha$$

This type generates a destructor $\mathbb{D} := \lambda x.\,\mathsf{out}\,x$ such that

$$\rhd\ \mathbb{D} : \mathsf{stream}(\rho) \to \rho \times \mathsf{stream}(\rho)$$

The usual destructors are encoded in the destructor $\mathbb{D}$ and are defined as

$$\mathsf{head} := \lambda x.\pi_1(\mathbb{D}x) \qquad \mathsf{tail} := \lambda x.\pi_2(\mathbb{D}x)$$

As this example shows, the use of projections is essential to obtain the actual destructors.

More (Co)inductive Types in MICT

- Lists of objets of type ρ: $\mathsf{list}(\rho) := \mu\alpha.1 + \rho \times \alpha$
- Well-founded ρ-branching trees: $\mathsf{tree}(\rho) := \mu\alpha.1 + (\rho \to \alpha)$
- Infinite depth ρ-labelled trees: $\mathsf{inftree}(\rho) := \nu\alpha.\rho \times \mathsf{list}(\alpha)$

with $\alpha \notin FV(\rho)$ in all cases.

2.2.2 Strong Normalization of MICT

This will be the last system for which we give a direct proof of strong normalization. We proceed by extending the proof for $\mathsf{F}^{+,\times}$ given in section 1.2.1.

The concept of elimination is extended with the following expressions:

$$\mathsf{It}(m,s,\star),\ \mathsf{Rec}(m,s,\star),\ \mathsf{in}^{-1}(m,\star), \mathsf{out}\,\star$$

The definition of the set SN is extended with the following rules:

$$\frac{m,s,E[x] \in \mathsf{SN}}{\mathsf{It}(m,s,E[x]) \in \mathsf{SN}} \qquad \frac{m,s,E[x] \in \mathsf{SN}}{\mathsf{Rec}(m,s,E[x]) \in \mathsf{SN}} \qquad \frac{m,E[x] \in \mathsf{SN}}{\mathsf{in}^{-1}(m,E[x]) \in \mathsf{SN}}$$

$$\frac{E[x] \in \mathsf{SN}}{\mathsf{out}\,E[x] \in \mathsf{SN}}$$

$$\frac{t \in \mathsf{SN}}{\mathsf{in}\,t \in \mathsf{SN}} \qquad \frac{E\Big[s\Big(m(\lambda x.\mathsf{It}(m,s,x))t\Big)\Big] \in \mathsf{SN}}{E\big[\mathsf{It}(m,s,\mathsf{in}\,t)\big] \in \mathsf{SN}}$$

$$\frac{E\Big[s\Big(m(\langle \mathsf{Id}, \lambda x.\mathsf{Rec}(m,s,x)\rangle)t\Big)\Big] \in \mathsf{SN}}{E\big[\mathsf{Rec}(m,s,\mathsf{in}\,t)\big] \in \mathsf{SN}} \qquad \frac{E[m(\lambda zz)t] \in \mathsf{SN}}{E\big[\,\mathsf{in}^{-1}(m,\mathsf{in}\,t)\big] \in \mathsf{SN}}$$

$$\frac{m,s,t \in \mathsf{SN}}{\mathsf{CoIt}(m,s,t) \in \mathsf{SN}} \qquad \frac{m,s,t \in \mathsf{SN}}{\mathsf{CoRec}(m,s,t) \in \mathsf{SN}} \qquad \frac{m,t \in \mathsf{SN}}{\mathsf{out}^{-1}(m,t) \in \mathsf{SN}}$$

$$\frac{E\big[m\big(\lambda z.\mathsf{CoIt}(m,s,z)\big)(st)\big] \in \mathsf{SN}}{E\big[\,\mathsf{out}\,\mathsf{CoIt}(m,s,t)\big] \in \mathsf{SN}} \qquad \frac{E\big[m\big([\mathsf{Id},\lambda z.\mathsf{CoRec}(m,s,z)]\big)(st)\big] \in \mathsf{SN}}{E\big[\,\mathsf{out}\,\mathsf{CoRec}(m,s,t)\big] \in \mathsf{SN}}$$

$$\frac{E\big[m(\lambda zz)t\big] \in \mathsf{SN}}{E\big[\,\mathsf{out}\,\mathsf{out}^{-1}(m,t)\big] \in \mathsf{SN}}$$

The definition of SAT sets is extended with the following clauses:

$$\frac{E\Big[s\Big(m\big(\lambda x.\mathsf{It}(m,s,x)\big)t\Big)\Big] \in \mathcal{M}}{E\big[\mathsf{It}(m,s,\mathsf{in}\,t)\big] \in \mathcal{M}}$$

$$\frac{E\Big[s\Big(m\big(\langle \mathsf{Id},\lambda x.\mathsf{Rec}(m,s,x)\rangle\big)t\Big)\Big] \in \mathcal{M}}{E\big[\mathsf{Rec}(m,s,\mathsf{in}\,t)\big] \in \mathcal{M}} \qquad \frac{E\big[m(\lambda zz)t\big] \in \mathcal{M}}{E\big[\,\mathsf{in}^{-1}(m,\mathsf{in}\,t)\big] \in \mathcal{M}}$$

$$\frac{E\big[m\big(\lambda z.\mathsf{CoIt}(m,s,z)\big)(st)\big] \in \mathcal{M}}{E\big[\,\mathsf{out}\,\mathsf{CoIt}(m,s,t)\big] \in \mathcal{M}} \qquad \frac{E\big[m\big([\mathsf{Id},\lambda z.\mathsf{CoRec}(m,s,z)]\big)(st)\big] \in \mathcal{M}}{E\big[\,\mathsf{out}\,\mathsf{CoRec}(m,s,t)\big] \in \mathcal{M}}$$

$$\frac{E\big[m(\lambda zz)t\big] \in \mathcal{M}}{E\big[\,\mathsf{out}\,\mathsf{out}^{-1}(m,t)\big] \in \mathcal{M}}$$

Saturated Sets for Inductive Types

From now on, we fix $\Phi : \mathsf{SAT} \to \mathsf{SAT}$.

Definition 2.1 *Given $\mathcal{M} \in \mathsf{SAT}$ we define*

$$\mathcal{I}_\mu(\mathcal{M}) := \{\mathsf{in}\,r \mid r \in \Phi(\mathcal{M})\}$$

and $\Psi_I : \mathsf{SAT} \to \mathsf{SAT}$ as

$$\Psi_I(\mathcal{M}) := \mathsf{cl}(\mathcal{I}_\mu(\mathcal{M})).$$

As we do not know if Ψ_I is monotone we proceed as follows: set

$$\mathsf{mon}(\Phi) := \bigcap_{\mathcal{P},\mathcal{Q}\,\in\mathsf{SAT}} (\mathcal{P} \to \mathcal{Q}) \to (\Phi(\mathcal{P}) \to \Phi(\mathcal{Q}))$$

and define $\Phi^{\supseteq} : \mathsf{SAT} \to \mathcal{P}(\mathsf{SN})$ as:

$$\Phi^{\supseteq}(\mathcal{M}) := \{t \in \mathsf{SN} \mid \forall m \in \mathsf{mon}(\Phi), \forall \mathcal{N} \in \mathsf{SAT}, \forall s \in \mathcal{M} \to \mathcal{N}.mst \in \Phi(\mathcal{N})\}$$

Lemma 2.1 *For all $\mathcal{P}, \mathcal{Q}, \mathcal{N} \in \mathsf{SAT}$. If $\mathcal{P} \subseteq \mathcal{Q}$ then $\mathcal{Q} \to \mathcal{N} \subseteq \mathcal{P} \to \mathcal{N}$.*
Proof. Assume $\mathcal{P} \subseteq \mathcal{Q}$. It suffices to show $\mathcal{I}_{\to}(\mathcal{Q},\mathcal{N}) \cap \mathsf{SN} = \mathcal{I}_{\to}(\mathcal{Q},\mathcal{N}) \subseteq \mathcal{P} \to \mathcal{N}$. Take $\lambda xt \in \mathcal{I}_{\to}(\mathcal{Q},\mathcal{N})$, i.e., $t \in \mathsf{S}_x(\mathcal{Q},\mathcal{N})$. To show $\lambda xt \in \mathcal{P} \to \mathcal{N}$ it suffices to prove $t \in \mathsf{S}_x(\mathcal{P},\mathcal{N})$. Therefore we take $p \in \mathcal{P}$ and show $t[x := p] \in \mathcal{N}$, but this is clear from $t \in \mathsf{S}_x(\mathcal{Q},\mathcal{N})$ because by assumption we also have $p \in \mathcal{Q}$. ⊣

Corollary 2.1 *$\Phi^{\supseteq}$ is monotone, i.e., for all $\mathcal{P}, \mathcal{Q}, \in$ SAT, if $\mathcal{P} \subseteq \mathcal{Q}$ then $\Phi^{\supseteq}(\mathcal{P}) \subseteq \Phi^{\supseteq}(\mathcal{Q})$.*
Proof. Assume $\mathcal{P} \subseteq \mathcal{Q}$ and take $t \in \Phi^{\supseteq}(\mathcal{P})$. Take also $\mathcal{N} \in \mathsf{SAT}$, $m \in \mathsf{mon}(\Phi)$ and $s \in \mathcal{Q} \to \mathcal{N}$. We need to show $mst \in \Phi(\mathcal{N})$. By the previous lemma $s \in \mathcal{Q} \to \mathcal{N}$ implies $s \in \mathcal{P} \to \mathcal{N}$. The claim follows now from the assumption $t \in \Phi^{\supseteq}(\mathcal{P})$. ⊣

Next define

$$\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}) := \{\mathsf{in}\, r \mid r \in \Phi^{\supseteq}(\mathcal{M})\}$$

and $\Psi_I^{\supseteq} : \mathsf{SAT} \to \mathsf{SAT}$ as

$$\Psi_I^{\supseteq}(\mathcal{M}) := \mathsf{cl}(\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}))$$

Clearly $\Psi_I^{\supseteq}$ is monotone, because so is $\Phi^{\supseteq}$, therefore the following definition is correct

$$\mu(\Phi) := \mathsf{lfp}(\Psi_I^{\supseteq}).$$

i.e. $\mu(\Phi)$ is the least fixed point of $\Psi_I^{\supseteq}$.

Lemma 2.2 *$\mathcal{I}_{\mu}(\mathcal{M}) \subseteq \mathsf{SN}$ and $\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}) \subseteq \mathsf{SN}$.*
Proof. We show the second claim. Take $t \in \mathcal{I}_{\mu}^{\supseteq}(\mathcal{M})$, that is, $t \equiv \mathsf{in}\, r$ with $r \in \Phi^{\supseteq}(\mathcal{M})$. As $\Phi^{\supseteq}(\mathcal{M}) \subseteq \mathsf{SN}$ we have $r \in \mathsf{SN}$, which by definition of SN implies $\mathsf{in}\, r \in \mathsf{SN}$, i.e., $t \in \mathsf{SN}$. ⊣

Corollary 2.2 *$\mathcal{I}_{\mu}(\mathcal{M}) \subseteq \Psi_I(\mathcal{M})$ and $\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}) \subseteq \Psi_I^{\supseteq}(\mathcal{M})$.*
Proof. We proof the second claim. By definition of the closure we have $\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}) \cap \mathsf{SN} \subseteq \Psi_I^{\supseteq}(\mathcal{M})$. But the previous lemma yields $\mathcal{I}_{\mu}^{\supseteq}(\mathcal{M}) \cap \mathsf{SN} = \mathcal{I}_{\mu}^{\supseteq}(\mathcal{M})$. ⊣

Definition 2.2 *Given $\Phi : \mathsf{SAT} \to \mathsf{SAT}$ and $\mathcal{M} \in \mathsf{SAT}$ we define*

$$\mathcal{E}_{\mu}(\mathcal{M}) := \Big\{ r \in \mathsf{SN} \;\Big|\; \begin{array}{l} \forall m \in \mathsf{mon}(\Phi).\, \forall \mathcal{N} \in \mathsf{SAT}. \\ \big(\forall s \in \Phi(\mathcal{N}) \to \mathcal{N}.\ \mathsf{It}(m,s,r) \in \mathcal{N}\big) \wedge \\ \big(\forall s \in \Phi(\mathcal{M} \times \mathcal{N}) \to \mathcal{N}.\ \mathsf{Rec}(m,s,r) \in \mathcal{N}\big) \wedge \\ \mathsf{in}^{-1}(m,r) \in \Phi(\mathcal{M}) \end{array} \Big\}$$

and $\Psi_E : \mathsf{SAT} \to \mathsf{SAT}$ as

$$\Psi_E(\mathcal{M}) := \mathsf{cl}(\mathcal{E}_{\mu}(\mathcal{M})).$$

Lemma 2.3 $\mathcal{E}_\mu(\mathcal{M}) \in \mathsf{SAT}$.
Proof. Is clear that $\mathcal{E}_\mu(\mathcal{M}) \subseteq \mathsf{SN}$.
Take $E[x] \in \mathsf{SN}$. We have to show that $E[x] \in \mathcal{E}_\mu(\mathcal{M})$. Fix $m \in \mathsf{mon}(\Phi), \mathcal{N} \in \mathsf{SAT}$.

- ∘ Assume $s \in \Phi(\mathcal{N}) \to \mathcal{N}$.
  The goal is $\mathsf{It}(m, s, E[x]) \in \mathcal{N}$. Observe that this term is again a multiple elimination say $E'[x]$. As $\mathcal{N} \in \mathsf{SAT}$ it suffices to show that $E'[x] \in \mathsf{SN}$. We have $E[x] \in \mathsf{SN}$ and $s \in \Phi(\mathcal{N}) \to \mathcal{N} \subseteq \mathsf{SN}$ implies $s \in \mathsf{SN}$, similarly $m \in \mathsf{mon}(\Phi) \subseteq \mathsf{SN}$. Therefore all $m, s, E[x] \in \mathsf{SN}$ which by properties of SN implies $\mathsf{It}(m, s, E[x]) \in \mathsf{SN}$.

- ∘ Assume $s \in \Phi(\mathcal{M} \times \mathcal{N}) \to \mathcal{N}$. The goal is $\mathsf{Rec}(m, s, E[x]) \in \mathcal{N}$. As in the previous case we obtain $m, s \in \mathsf{SN}$, therefore by properties of SN we conclude $E'[x] := \mathsf{Rec}(m, s, E[x]) \in \mathsf{SN}$. Therefore, as $\mathcal{N} \in \mathsf{SAT}$ we get $E'[x] \in \mathcal{N}$.

- ∘ Goal is $\mathsf{in}^{-1}(m, E[x]) \in \Phi(\mathcal{M})$. Again we have $m \in \mathsf{SN}$ therefore, as $E[x] \in \mathsf{SN}$ by properties of SN we get $E'[x] \equiv \mathsf{in}^{-1}(m, E[x]) \in \mathsf{SN}$, which implies $E'[x] \in \Phi(\mathcal{M})$, because $\Phi(\mathcal{M}) \in \mathsf{SAT}$.

The other closure rules for SAT sets are proved in a similar way. ⊣

Corollary 2.3 $\mathcal{E}_\mu(\mathcal{M}) = \Psi_E(\mathcal{M})$.
Proof. $\subseteq$). we have $\mathcal{E}_\mu(\mathcal{M}) = \mathcal{E}_\mu(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathsf{cl}(\mathcal{E}_\mu(\mathcal{M})) \equiv \Psi_E(\mathcal{M})$.
$\supseteq$). By the previous lemma we have $\mathcal{E}_\mu(\mathcal{M}) \in \mathsf{SAT}$. Therefore by minimality of the closure we get $\Psi_E(\mathcal{M}) \equiv \mathsf{cl}(\mathcal{E}_\mu(\mathcal{M})) \subseteq \mathcal{E}_\mu(\mathcal{M})$.
⊣

Lemma 2.4 $\Psi_I(\mathcal{M}) \subseteq \mathcal{M} \Leftrightarrow \forall t \in \Phi(\mathcal{M}).\ \mathsf{in}\, t \in \mathcal{M}$.
Proof. $\Rightarrow$) Assume $\Psi_I(\mathcal{M}) \subseteq \mathcal{M}$, i.e., $\mathsf{cl}(\mathcal{I}_\mu(\mathcal{M})) \subseteq \mathcal{M}$. Take $t \in \Phi(\mathcal{M})$, this implies $\mathsf{in}\, t \in \mathcal{I}_\mu(\mathcal{M})$, which, by corollary 2.2, implies $\mathsf{in}\, t \in \Psi_I(\mathcal{M}) \subseteq \mathcal{M}$. Therefore $\mathsf{in}\, t \in \mathcal{M}$.
$\Leftarrow$) Assume $\forall t \in \Phi(\mathcal{M}).\ \mathsf{in}\, t \in \mathcal{M}$ and take $r \in \Psi_I(\mathcal{M}) \equiv \mathsf{cl}(\mathcal{I}_\mu(\mathcal{M}))$. Goal is $r \in \mathcal{M}$. As $\mathcal{M} \in \mathsf{SAT}$ it suffices to show $\mathcal{I}_\mu(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathcal{M}$, the goal follows by minimality of the closure. By lemma 2.2 we have $\mathcal{I}_\mu(\mathcal{M}) \subseteq \mathsf{SN}$, thus we only have to show $\mathcal{I}_\mu(\mathcal{M}) \subseteq \mathcal{M}$. Take $\mathsf{in}\, t \in \mathcal{I}_\mu(\mathcal{M})$, so $t \in \Phi(\mathcal{M})$ which by assumption implies $\mathsf{in}\, t \in \mathcal{M}$. Therefore $\mathcal{I}_\mu(\mathcal{M}) \subseteq \mathcal{M}$. ⊣

Lemma 2.5

$$\begin{array}{rl}\mathcal{M} \subseteq \Psi_E(\mathcal{M}) \Leftrightarrow & \forall r \in \mathcal{M}. \forall m \in \mathsf{mon}(\Phi). \forall \mathcal{N} \in \mathsf{SAT}. \\ & \big(\forall s \in \Phi(\mathcal{N}) \to \mathcal{N}.\ \mathsf{It}(m, s, r) \in \mathcal{N}\big) \wedge \\ & \big(\forall s \in \Phi(\mathcal{M} \times \mathcal{N}) \to \mathcal{N}.\ \mathsf{Rec}(m, s, r) \in \mathcal{N}\big) \wedge \\ & \mathsf{in}^{-\mathbf{1}}(m, r) \in \Phi(\mathcal{M})\end{array}$$

Proof. Call $\Box(r)$ to the condition on the right hand side for a given $r \in \mathcal{M}$.
$\Rightarrow$). Assume $\mathcal{M} \subseteq \Psi_E(\mathcal{M})$. We have to show $\Box(r)$ for all $r \in \mathcal{M}$. Take $r \in \mathcal{M}$,

by corollary 2.3 we have $\mathcal{M} \subseteq \mathcal{E}_\mu(\mathcal{M})$. Observing that $\mathcal{E}_\mu(\mathcal{M}) = \{r \in \mathsf{SN} |\ \Box(r)\}$ we are done.
$\Leftarrow$) Assume $\forall r \in \mathcal{M}.\Box(r)$ and take $r \in \mathcal{M}$, we have to show that $r \in \Psi_E(M)$. By corollary 2.3 suffices to show that $r \in \mathcal{E}_\mu(\mathcal{M})$. We have $r \in \mathsf{SN}$ because $\mathcal{M} \subseteq \mathsf{SN}$. Moreover $\Box(r)$ holds by assumption, which implies $r \in \mathcal{E}_\mu(\mathcal{M})$. ⊣

Lemma 2.6 $\mu(\Phi)$ *is a pre-fixed point of* Ψ_I*. i.e.,*

$$\Psi_I\big(\mu(\Phi)\big) \subseteq \mu(\Phi)$$

Proof. By definition of $\mu(\Phi)$ it suffices to show

$$\Psi_I\Big(\Psi_I^{\supseteq}\big(\mu(\Phi)\big)\Big) \subseteq \Psi_I^{\supseteq}\big(\mu(\Phi)\big),$$

to show this we will use the lemma 2.4. Take $t \in \Phi\Big(\Psi_I^{\supseteq}\big(\mu(\Phi)\big)\Big)$, this implies $\mathsf{in}\, t \in \mathcal{I}_\mu^{\supseteq}\Big(\Psi_I^{\supseteq}\big(\mu(\Phi)\big)\Big) \subseteq \Psi_I^{\supseteq}\Big(\Psi_I^{\supseteq}\big(\mu(\Phi)\big)\Big)$, the last inclusion given by corollary 2.2. Therefore by definition of $\mu(\Phi)$ we conclude $\mathsf{in}\, t \in \Psi_I^{\supseteq}\big(\mu(\Phi)\big)$. ⊣

Lemma 2.7 $\mu(\Phi)$ *is a post-fixed point of* Ψ_E*. i.e.,*

$$\mu(\Phi) \subseteq \Psi_E(\mu(\Phi))$$

Proof. Our goal is $\mu(\Phi) \subseteq \Psi_E\big(\mu(\Phi)\big)$. To prove this we will use extended induction on $\mu(\Phi)$. Therefore the goal becomes

$$\Psi_I^{\supseteq}\Big(\mu(\Phi) \cap \Psi_E\big(\mu(\Phi)\big)\Big) \subseteq \Psi_E\big(\mu(\Phi)\big)$$

Set $\mathcal{L} := \mu(\Phi)$, $\mathcal{L}' := \mathcal{L} \cap \Psi_E(\mathcal{L})$. The goal is $\Psi_I^{\supseteq}(\mathcal{L}') \subseteq \Psi_E(\mathcal{L})$. By monotonicity of the closure it suffices to show

$$\mathcal{I}_\mu^{\supseteq}(\mathcal{L}') \subseteq \mathcal{E}_\mu(\mathcal{L}).$$

Take $t \in \mathcal{I}_\mu^{\supseteq}(\mathcal{L}')$, i.e., $t \equiv \mathsf{in}\, r$ with $r \in \Phi^{\supseteq}(\mathcal{L}')$. We need to show $\mathsf{in}\, r \in \mathcal{E}_\mu(\mathcal{L})$. First observe that $\mathsf{in}\, r \in \mathsf{SN}$ because $r \in \Phi^{\supseteq}(\mathcal{L}') \subseteq \mathsf{SN}$ and by properties of SN. Next we have to prove that $\Box(\mathsf{in}\, r)$ (cf. proof of lemma 2.5), so fix $m \in \mathsf{mon}(\Phi)$ and $\mathcal{N} \in \mathsf{SAT}$.

- ○ Take $s \in \Phi(\mathcal{N}) \to \mathcal{N}$. We want to show that $\mathsf{It}(m, s, \mathsf{in}\, r) \in \mathcal{N}$. Using that $\mathcal{N} \in \mathsf{SAT}$, it suffices to show that $s\big(m\big(\lambda x.\mathsf{It}(m, s, x)\big)r\big) \in \mathcal{N}$. As $s \in \Phi(\mathcal{N}) \to \mathcal{N}$ we only have to show $m\big(\lambda x.\mathsf{It}(m, s, x)\big)r \in \Phi(\mathcal{N})$ but observing that $r \in \Phi^{\supseteq}(\mathcal{L}')$ we only have to show that $m \in \mathsf{mon}(\Phi), \mathcal{N} \in \mathsf{SAT}$ and $\lambda x.\mathsf{It}(m, s, x) \in \mathcal{L}' \to \mathcal{N}$. The first two claims are given and to prove the last one we will show that $\mathsf{It}(m, s, x) \in \mathsf{S}_x(\mathcal{L}', \mathcal{N})$. Take $q \in \mathcal{L}'$ we prove $\mathsf{It}(m, s, x)[x := q] \in \mathcal{N}$, w.l.o.g. $x \notin FV(m, s)$ therefore we show $\mathsf{It}(m, s, q) \in \mathcal{N}$. We have $\mathcal{L}' \subseteq \Psi_E(\mathcal{L}) = \mathcal{E}_\mu(\mathcal{L})$, the equality given by corollary 2.3. Therefore $q \in \mathcal{E}_\mu(\mathcal{L})$ which immediately yields $\mathsf{It}(m, s, q) \in \mathcal{N}$.

- ∘ Take $s \in \Phi(\mathcal{L} \times \mathcal{N}) \to \mathcal{N}$. We need to prove $\mathsf{Rec}(m, s, \mathsf{in}\, r) \in \mathcal{N}$. By a similar reason as the previous case we only have to show

  $$\lambda z.\langle(\lambda yy)z, (\lambda x.\mathsf{Rec}(m, s, x))z\rangle \in \mathcal{L}' \to \mathcal{L} \times \mathcal{N}.$$

  It suffices to prove $\langle(\lambda yy)z, (\lambda x.\mathsf{Rec}(m, s, x))z\rangle \in \mathsf{S}_z(\mathcal{L}', \mathcal{L}\times\mathcal{N})$, so we take $q \in \mathcal{L}'$ and show $\langle(\lambda yy)q, (\lambda x.\mathsf{Rec}(m, s, x))q\rangle \in \mathcal{L} \times \mathcal{N}$. For this we prove two things:

  – $(\lambda yy)q \in \mathcal{L}$. Clearly we have $\lambda yy \in \mathcal{L} \to \mathcal{L}$ and as $q \in \mathcal{L}' \subseteq \mathcal{L}$ we get $(\lambda yy)q \in \mathcal{L}$.

  – $(\lambda x.\mathsf{Rec}(m, s, x))q \in \mathcal{N}$. It suffices to show $\lambda x.\mathsf{Rec}(m, s, x) \in \mathcal{L}' \to \mathcal{N}$, that is $\mathsf{Rec}(m, s, x) \in \mathsf{S}_x(\mathcal{L}', \mathcal{N})$. Take $p \in \mathcal{L}'$, we will show $\mathsf{Rec}(m, s, x)[x := p] \in \mathcal{N}$, where w.l.o.g. $x \notin FV(m, s)$ so we prove $\mathsf{Rec}(m, s, p) \in \mathcal{N}$. We have $\mathcal{L}' \subseteq \Psi_E(\mathcal{L}) = \mathcal{E}_\mu(\mathcal{L})$, the equality given by corollary 2.3. Therefore $p \in \mathcal{E}_\mu(\mathcal{L})$ which immediately yields $\mathsf{Rec}(m, s, p) \in \mathcal{N}$.

- ∘ Goal is $\mathsf{in}^{-1}(m, \mathsf{in}\, r) \in \Phi(\mathcal{L})$. As $r \in \Phi^{\supseteq}(\mathcal{L}')$ and $m \in \mathsf{mon}(\Phi)$ it suffices to show $\lambda zz \in \mathcal{L}' \to \mathcal{L}$, i.e., $z \in \mathsf{S}_z(\mathcal{L}', \mathcal{L})$, so we take $s \in \mathcal{L}'$ and want to show $s \in \mathcal{L}$, but this is obvious because $\mathcal{L}' \subseteq \mathcal{L}$.

Therefore $\Box(\mathsf{in}\, r)$ and we are done. ⊣

Saturated Sets for Coinductive Types

Definition 2.3 *Given* $\Phi : \mathsf{SAT} \to \mathsf{SAT}, \mathcal{M} \in \mathsf{SAT}$, *define*

$$\begin{array}{rcl}\mathcal{I}_\nu(\mathcal{M}) & := & \{\mathsf{CoIt}(m, s, t) \mid m \in \mathsf{mon}(\Phi),\ s \in \mathcal{N} \to \Phi(\mathcal{N}),\ t \in \mathcal{N}, \mathcal{N} \in \mathsf{SAT}\} \\ & \cup & \{\mathsf{CoRec}(m, s, t) \mid m \in \mathsf{mon}(\Phi),\ s \in \mathcal{N} \to \Phi(\mathcal{M} + \mathcal{N}),\ t \in \mathcal{N} \in \mathsf{SAT}\} \\ & \cup & \{\mathsf{out}^{-1}(m, t) \mid m \in \mathsf{mon}(\Phi),\ t \in \Phi(\mathcal{M})\}\end{array}$$

and $\Psi_I : \mathsf{SAT} \to \mathsf{SAT}$ *with*

$$\Psi_I(\mathcal{M}) := \mathsf{cl}(\mathcal{I}_\nu(\mathcal{M})).$$

Lemma 2.8 $\mathcal{I}_\nu(\mathcal{M}) \subseteq \mathsf{SN}$.
Proof. Take $r \in \mathcal{I}_\nu(\mathcal{M})$. We have three cases:

- ∘ $r \equiv \mathsf{CoIt}(m, s, t)$. We have $m, s, t \in \mathsf{SN}$ because they belong to some saturated set. Therefore by properties of SN we also have $\mathsf{CoIt}(m, s, t) \in \mathsf{SN}$.

- ∘ $r \equiv \mathsf{CoRec}(m, s, t)$. Similarly $m, s, t \in \mathsf{SN}$ implies $\mathsf{CoRec}(m, s, t) \in \mathsf{SN}$.

- ∘ $r \equiv \mathsf{out}^{-1}(m, t)$. Again $m, t \in \mathsf{SN}$ implies $\mathsf{out}^{-1}(m, t) \in \mathsf{SN}$.

⊣

Corollary 2.4 $\mathcal{I}_\nu(\mathcal{M}) \subseteq \Psi_I(\mathcal{M})$.
Proof. By definition of closure we have $\mathcal{I}_\nu(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathsf{cl}(\mathcal{I}_\nu(\mathcal{M}))$ which, by the previous lemma is the same as $\mathcal{I}_\nu(\mathcal{M}) \subseteq \mathsf{cl}(\mathcal{I}_\nu(\mathcal{M})) \equiv \Psi_I(\mathcal{M})$. ⊣

Definition 2.4 *Given* $\Phi : \mathsf{SAT} \to \mathsf{SAT}, \mathcal{M} \in \mathsf{SAT}$*, define*

$$\mathcal{E}_\nu(\mathcal{M}) := \{r \in \mathsf{SN} \mid \mathsf{out}\, r \in \Phi(\mathcal{M})\}$$

and $\Psi_E : \mathsf{SAT} \to \mathsf{SAT}$*, with*

$$\Psi_E(\mathcal{M}) := \mathsf{cl}(\mathcal{E}_\nu(\mathcal{M})).$$

As we do not know if Ψ_E is monotone we proceed as follows:
Define $\Phi^{\subseteq} : \mathsf{SAT} \to \mathsf{SAT}$ as

$$\Phi^{\subseteq}(\mathcal{M}) := \mathsf{cl}(\mathsf{A}(\mathcal{M}))$$

with

$$\mathsf{A}(\mathcal{M}) := \{mqr \mid m \in \mathsf{mon}(\Phi),\ q \in \mathcal{N} \to \mathcal{M},\ r \in \Phi(\mathcal{N}) \text{ for some } \mathcal{N} \in \mathsf{SAT}\}$$

Lemma 2.9 *For all* $\mathcal{M} \in \mathsf{SAT}$*,* $\mathsf{A}(\mathcal{M}) \subseteq \Phi(\mathcal{M})$.
Proof. Take $t \in \mathsf{A}(\mathcal{M})$, i.e.,$t \equiv mqr$ with $m \in \mathsf{mon}(\Phi)$, $q \in \mathcal{N} \to \mathcal{M}$, $r \in \Phi(\mathcal{N})$ for some $\mathcal{N} \in \mathsf{SAT}$. $m \in \mathsf{mon}(\Phi) \;\Rightarrow\; m \in (\mathcal{N} \to \mathcal{M}) \to (\Phi(\mathcal{N}) \to \Phi(\mathcal{M})) \;\Rightarrow\; mq \in \Phi(\mathcal{N}) \to \Phi(\mathcal{M}) \;\Rightarrow mqr \in \Phi(\mathcal{M})$, i.e. $t \in \Phi(\mathcal{M})$. ⊣

Corollary 2.5 *For all* $\mathcal{M} \in \mathsf{SAT}$*,* $\mathsf{A}(\mathcal{M}) \subseteq \mathsf{SN}$.
Proof. $\mathsf{A}(\mathcal{M}) \subseteq \Phi(\mathcal{M}) \subseteq \mathsf{SN}$. ⊣

Corollary 2.6 *For all* $\mathcal{M} \in \mathsf{SAT}$*,* $\Phi^{\subseteq}(\mathcal{M}) \subseteq \Phi(\mathcal{M})$.
Proof. As $\Phi(\mathcal{M}) \in \mathsf{SAT}$, by minimality of the closure it suffices to show $\mathsf{A}(\mathcal{M}) \cap \mathsf{SN} \subseteq \Phi(\mathcal{M})$, but by the previous corollary we only need to show $\mathsf{A}(\mathcal{M}) \subseteq \Phi(\mathcal{M})$ but this is the statement of the lemma. ⊣

Corollary 2.7 *For all* $\mathcal{M} \in \mathsf{SAT}$*,* $\mathsf{A}(\mathcal{M}) \subseteq \Phi^{\subseteq}(\mathcal{M})$.
Proof. $\mathsf{A}(\mathcal{M}) = \mathsf{A}(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathsf{cl}(\mathsf{A}(\mathcal{M})) \equiv \Phi^{\subseteq}(\mathcal{M})$. ⊣

Lemma 2.10 *For all* $\mathcal{P}, \mathcal{Q}, \mathcal{N} \in \mathsf{SAT}$. *If* $\mathcal{P} \subseteq \mathcal{Q}$ *then* $\mathcal{N} \to \mathcal{P} \subseteq \mathcal{N} \to \mathcal{Q}$.
Proof. It suffices to show that $\mathcal{I}_{\to}(\mathcal{N},\mathcal{P}) \cap \mathsf{SN} = \mathcal{I}_{\to}(\mathcal{N},\mathcal{P}) \subseteq \mathcal{I}_{\to}(\mathcal{N},\mathcal{Q})$. Take $\lambda xt \in \mathcal{I}_{\to}(\mathcal{N},\mathcal{P})$, i.e., $t \in \mathsf{S}_x(\mathcal{N},\mathcal{P})$. Therefore we have $\forall s \in \mathcal{N}.t[x := s] \in \mathcal{P}$ which by assumption implies $\forall s \in \mathcal{N}.t[x := s] \in \mathcal{Q}$. That is $t \in \mathsf{S}_x(\mathcal{N},\mathcal{Q}) \Rightarrow \lambda xt \in \mathcal{I}_{\to}(\mathcal{N},\mathcal{Q})$. ⊣

Corollary 2.8 $\Phi^{\subseteq}$ *is monotone, i.e., for all* $\mathcal{P}, \mathcal{Q} \in \mathsf{SAT}$*, if* $\mathcal{P} \subseteq \mathcal{Q}$ *then* $\Phi^{\subseteq}(\mathcal{P}) \subseteq \Phi^{\subseteq}(\mathcal{Q})$.
Proof. Assume $\mathcal{P} \subseteq \mathcal{Q}$. Take $mqr \in \Phi^{\subseteq}(\mathcal{P})$, then $m \in \mathsf{mon}(\Phi)$, $q \in \mathcal{N} \to \mathcal{P}$, $r \in \Phi(\mathcal{N})$. $q \in \mathcal{N} \to \mathcal{P}$ implies by the previous lemma $q \in \mathcal{N} \to \mathcal{Q}$. Therefore we have $mqr \in \Phi^{\subseteq}(\mathcal{Q})$. ⊣

Next set

$$\mathcal{E}^{\subseteq}_{\nu}(\mathcal{M}) := \{r \in \mathsf{SN} \mid \mathsf{out}\, r \in \Phi^{\subseteq}(\mathcal{M})\}$$

and define $\Psi^{\subseteq}_{E} : \mathsf{SAT} \to \mathsf{SAT}$ as

$$\Psi^{\subseteq}_{E}(\mathcal{M}) := \mathsf{cl}(\mathcal{E}^{\subseteq}_{\nu}(\mathcal{M})).$$

Clearly $\Psi^{\subseteq}_{E}$ is monotone, because so is $\Phi^{\subseteq}$, therefore the following definition is valid:

$$\nu(\Phi) := \mathsf{gfp}(\Psi^{\subseteq}_{E}).$$

i.e., $\nu(\Phi)$ is the greatest fixed point of $\Psi^{\subseteq}_{E}$.

Lemma 2.11 $\mathcal{E}_{\nu}(\mathcal{M})$, $\mathcal{E}^{\subseteq}_{\nu}(\mathcal{M}) \in \mathsf{SAT}$.
Proof. We show the first part. Clearly we have $\mathcal{E}_{\nu}(\mathcal{M}) \subseteq \mathsf{SN}$.
Take $E[x] \in \mathsf{SN}$. Goal is $E[x] \in \mathcal{E}_{\nu}(\mathcal{M})$, i.e., $\mathsf{out}\,\mathcal{E}[x] \in \Phi(\mathcal{M})$. By properties of SN, $E[x] \in \mathsf{SN}$ implies $\mathsf{out}\,\mathcal{E}[x] \in \mathsf{SN}$, but $\mathsf{out}\,\mathcal{E}[x]$ is a multiple elimination say $E'[x] \in \mathsf{SN}$. Therefore, as $\Phi(\mathcal{M}) \in \mathsf{SAT}$, we get $E'[x] \in \Phi(\mathcal{M})$. The remaining rules are easily proved.
⊣

Corollary 2.9 $\mathcal{E}_{\nu}(\mathcal{M}) = \Psi_{E}(\mathcal{M})$, $\mathcal{E}^{\subseteq}_{\nu}(\mathcal{M}) = \Psi^{\subseteq}_{E}(\mathcal{M})$
Proof. We show the first part.
$\subseteq$) We have $\mathcal{E}_{\nu}(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathsf{cl}(\mathcal{E}_{\nu}(\mathcal{M}))$, which, as $\mathcal{E}_{\nu}(\mathcal{M}) \subseteq \mathsf{SN}$, is the same as $\mathcal{E}_{\nu}(\mathcal{M}) \subseteq \mathsf{cl}(\mathcal{E}_{\nu}(\mathcal{M})) \equiv \Psi_{E}(\mathcal{M})$.
$\supseteq$). By the previous lemma, using the minimality of the closure we have $\Psi_{E}(\mathcal{M}) = \mathsf{cl}(\mathcal{E}_{\nu}(\mathcal{M})) \subseteq \mathcal{E}_{\nu}(\mathcal{M})$. ⊣

Lemma 2.12 $\mathcal{M} \subseteq \Psi_{E}(\mathcal{M}) \Leftrightarrow \forall t \in \mathcal{M}.\, \mathsf{out}\, t \in \Phi(\mathcal{M})$
Proof. $\Rightarrow$) Take $t \in \mathcal{M}$, by assumption we get $t \in \Psi_{E}(\mathcal{M})$, and by the previous corollary $t \in \mathcal{E}_{\nu}(\mathcal{M})$, which by definition of $\mathcal{E}_{\nu}(\mathcal{M})$ yields $\mathsf{out}\, t \in \Phi(\mathcal{M})$.
$\Leftarrow$) Take $t \in \mathcal{M}$, by assumption we get $\mathsf{out}\, t \in \Phi(\mathcal{M})$. On the other hand, as $\mathcal{M} \subseteq \mathsf{SN}$, we get $t \in \mathsf{SN}$. Therefore $t \in \mathcal{E}_{\nu}(\mathcal{M})$, which by the previous corollary is the same as $t \in \Psi_{E}(\mathcal{M})$. ⊣

Lemma 2.13

$$\begin{array}{rl}
\Psi_{I}(\mathcal{M}) \subseteq \mathcal{M} \Leftrightarrow & \forall m \in \mathsf{mon}(\Phi).\, \forall \mathcal{N} \in \mathsf{SAT}. \\
& \big(\forall t \in \mathcal{N}\ \forall s \in \mathcal{N} \to \Phi(\mathcal{N}).\ \mathsf{CoIt}(m,s,t) \in \mathcal{M}\,\big) \wedge \\
& \big(\forall t \in \mathcal{N}\ \forall s \in \mathcal{N} \to \Phi(\mathcal{M}+\mathcal{N}).\ \mathsf{CoRec}(m,s,t) \in \mathcal{M}\,\big) \wedge \\
& \big(\forall t \in \Phi(\mathcal{M}).\, \mathsf{out}^{-1}(m,t) \in \mathcal{M}\big)
\end{array}$$

Proof. $\Rightarrow$). Assume $\Psi_{I}(\mathcal{M}) \subseteq \mathcal{M}$. By corollary 2.4 we get $\mathcal{I}_{\nu}(\mathcal{M}) \subseteq \mathcal{M}$.
Take $m \in \mathsf{mon}(\Phi), \mathcal{N} \in \mathsf{SAT}$. We prove every part of the conjunction:

- ◦ Take $t \in \mathcal{N}$, $s \in \mathcal{N} \to \Phi(\mathcal{N})$. From this we get $\mathsf{CoIt}(m,s,t) \in \mathcal{I}_{\nu}(\mathcal{M})$, therefore $\mathsf{CoIt}(m,s,t) \in \mathcal{M}$.

- ◦ Take $t \in \mathcal{N}, s \in \mathcal{N} \to \Phi(\mathcal{M} + \mathcal{N})$. Analogously to the previous case we get $\mathsf{CoRec}(m, s, t) \in \mathcal{I}_\nu(\mathcal{M}) \subseteq \mathcal{M}$.
- ◦ Take $t \in \Phi(\mathcal{M})$. This yields $\mathsf{out}^{-1}(m, t) \in \mathcal{I}_\nu(\mathcal{M}) \subseteq \mathcal{M}$.

$\Leftarrow$) Assume the condition on the right hand side. We have $\Psi_I(M) = \mathsf{cl}(\mathcal{I}_\nu(\mathcal{M}))$. By minimality of the closure it suffices to show $\mathcal{I}_\nu(\mathcal{M}) \cap \mathsf{SN} \subseteq \mathcal{M}$. But by lemma 2.8 this is the same as $\mathcal{I}_\nu(\mathcal{M}) \subseteq \mathcal{M}$. But this follows immediately from the assumption and the definition of $\mathcal{I}_\nu(\mathcal{M})$. ⊣

Lemma 2.14 *$\nu(\Phi)$ is a pre-fixed point of Ψ_I. i.e.,*

$$\Psi_I(\nu(\Phi)) \subseteq \nu(\Phi)$$

Proof. We will use extended coinduction. Therefore the goal becomes

$$\Psi_I\big(\nu(\Phi)\big) \subseteq \Psi_E^{\subseteq}\Big(\nu(\Phi) \cup \Psi_I\big(\nu(\Phi)\big)\Big)$$

Set $\mathcal{G} := \nu(\Phi)$, $\mathcal{G}' := \mathcal{G} \cup \Psi_I(\mathcal{G})$. The goal becomes $\Psi_I(\mathcal{G}) \subseteq \Psi_E^{\subseteq}(\mathcal{G}')$. By monotonicity of the closure it suffices to show

$$\mathcal{I}_\nu(\mathcal{G}) \subseteq \mathcal{E}_\nu^{\subseteq}(\mathcal{G}')$$

Assume $r \in \mathcal{I}_\nu(\mathcal{G})$. To show $r \in \mathcal{E}_\nu^{\subseteq}(\mathcal{G}')$ it suffices $\mathsf{out}\, r \in \Phi^{\subseteq}(\mathcal{G}')$ ($r \in \mathsf{SN}$ because $\mathcal{I}_\nu(\mathcal{G}) \subseteq \mathsf{SN}$). We have three cases:

- ◦ $r \equiv \mathsf{CoIt}(m, s, t)$ with $m \in \mathsf{mon}(\Phi), s \in \mathcal{N} \to \Phi(\mathcal{N}), t \in \mathcal{N}$. By properties of saturated sets it suffices to show $m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)(st) \in \Phi^{\subseteq}(\mathcal{G}')$ and using corollary 2.7 we will prove only $m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)(st) \in \mathsf{A}(\mathcal{G}')$. We have by assumption $m \in \mathsf{mon}(\Phi)$ and easily we get $st \in \Phi(\mathcal{N})$. To prove $\lambda z.\mathsf{CoIt}(m, s, z) \in \mathcal{N} \to \mathcal{G}'$, we show $\mathsf{CoIt}(m, s, z) \in \mathsf{S}_z(\mathcal{N}, \mathcal{G}')$. Taking $q \in \mathcal{N}$ we show $\mathsf{CoIt}(m, s, z)[z := q] \equiv \mathsf{CoIt}(m, s, q) \in \mathcal{G}'$. Clearly $\mathsf{CoIt}(m, s, q) \in \mathcal{I}_\nu(\mathcal{G})$, therefore by corollary 2.4 we have $\mathsf{CoIt}(m, s, q) \in \Psi_I(\mathcal{G}) \subseteq \mathcal{G}'$.
- ◦ $r \equiv \mathsf{CoRec}(m, s, t)$ with $m \in \mathsf{mon}(\Phi), s \in \mathcal{N} \to \Phi(\mathcal{G} + \mathcal{N}), t \in \mathcal{N}$. By similar reasoning as the previous case we only need to show

  $$m\big([\mathsf{Id}, \lambda z.\mathsf{CoRec}(m, s, z)]\big)(st) \in \mathsf{A}(\mathcal{G}').$$

  We have $m \in \mathsf{mon}(\Phi)$ and easily we get $st \in \Phi(\mathcal{G} + \mathcal{N})$. Remains to show that $[\mathsf{Id}, \lambda z.\mathsf{CoRec}(m, s, z)] \in \mathcal{G}{+}\mathcal{N} \to \mathcal{G}'$. We have $[\mathsf{Id}, \lambda z.\mathsf{CoRec}(m, s, z)] \equiv \lambda x.\mathsf{case}(x, y.y, z.\mathsf{CoRec}(m, s, z))$ therefore the goal reduces to show

  $$\mathsf{case}(x, y.y, z.\mathsf{CoRec}(m, s, z)) \in \mathsf{S}_x(\mathcal{G} + \mathcal{N}, \mathcal{G}').$$

  So we take $q \in \mathcal{G} + \mathcal{N}$ and prove $\mathsf{case}(x, y.y, z.\mathsf{CoRec}(m, s, z)) \in \mathcal{G}'$, which, by properties of saturated sets, reduces to the next two claims:

- $y \in \mathsf{S}_y(\mathcal{G}, \mathcal{G}')$. This holds trivially because $\mathcal{G} \subseteq \mathcal{G}'$.
- $\mathsf{CoRec}(m, s, z) \in \mathsf{S}_z(\mathcal{N}, \mathcal{G}')$. For this we take $p \in \mathcal{N}$ and show $\mathsf{CoRec}(m, s, z)[z := p] \equiv \mathsf{CoRec}(m, s, p) \in \mathcal{G}'$. Clearly we have $\mathsf{CoRec}(m, s, p) \in \mathcal{I}_\nu(\mathcal{G})$. Therefore by corollary 2.4 we have

$$\mathsf{CoRec}(m, s, p) \in \Psi_I(\mathcal{G}) \subseteq \mathcal{G}'.$$

○ $r \equiv \mathsf{out}^{-1}(m, t)$ with $m \in \mathsf{mon}(\Phi)$ and $t \in \Phi(\mathcal{G})$. By properties of saturated sets it suffices to show $m(\lambda zz)t \in \Phi^{\subseteq}(\mathcal{G}')$. Using corollary 2.7 we show $m(\lambda zz)t \in \mathsf{A}(\mathcal{G}')$. We have $m \in \mathsf{mon}(\Phi)$ and $t \in \Phi(\mathcal{G})$, only remains to show $\lambda zz \in \mathcal{G} \to \mathcal{G}'$, but this is consequence of $\mathcal{G} \subseteq \mathcal{G}'$.

⊣

Lemma 2.15 *$\nu(\Phi)$ is a post-fixed point of Ψ_E. i.e.,*

$$\nu(\Phi) \subseteq \Psi_E(\nu(\Phi))$$

Proof. By lemma 2.12 it suffices to show $\forall t \in \nu(\Phi). \mathsf{out}\, t \in \Phi\big(\nu(\Phi)\big)$. By definition we have $\nu(\Phi) = \Psi_E^{\subseteq}\big(\nu(\Phi)\big)$ and by corollary 2.9 $\Psi_E^{\subseteq}\big(\nu(\Phi)\big) = \mathcal{E}_\nu^{\subseteq}\big(\nu(\Phi)\big)$. So take $t \in \nu(\Phi) = \mathcal{E}_\nu^{\subseteq}\big(\nu(\Phi)\big) \Rightarrow \mathsf{out}\, t \in \Phi^{\subseteq}\big(\nu(\Phi)\big)$. Finally by corollary 2.6 we get $\mathsf{out}\, t \in \Phi\big(\nu(\Phi)\big)$. ⊣

Proposition 2.2 (Properties of Saturated Sets) *Given $\Phi : \mathsf{SAT} \to \mathsf{SAT}$ the following holds.*

1. *$\mu(\Phi) \in \mathsf{SAT}$.*
2. *If $t \in \Phi(\mu(\Phi))$ then $\mathsf{in}\, t \in \mu(\Phi)$.*
3. *If $r \in \mu(\Phi), m \in \mathsf{mon}(\Phi), \mathcal{N} \in \mathsf{SAT}$ and $s \in \Phi(\mathcal{N}) \to \mathcal{N}$ then $\mathsf{It}(m, s, r) \in \mathcal{N}$.*
4. *If $r \in \mu(\Phi), m \in \mathsf{mon}(\Phi), \mathcal{N} \in \mathsf{SAT}$ and $s \in \Phi(\mu(\Phi) \times \mathcal{N}) \to \mathcal{N}$ then $\mathsf{Rec}(m, s, r) \in \mathcal{N}$.*
5. *If $m \in \mathsf{mon}(\Phi)$ and $r \in \mu(\Phi)$ then $\mathsf{in}^{-1}(m, r) \in \Phi(\mu(\Phi))$.*
6. *$\nu(\Phi) \in \mathsf{SAT}$.*
7. *If $t \in \nu(\Phi)$ then $\mathsf{out}\, t \in \Phi(\nu(\Phi))$.*
8. *If $\mathcal{N} \in \mathsf{SAT}, r \in \mathcal{N}, m \in \mathsf{mon}(\Phi)$ and $s \in \mathcal{N} \to \Phi(\mathcal{N})$ then $\mathsf{CoIt}(m, s, r) \in \nu(\Phi)$.*
9. *If $\mathcal{N} \in \mathsf{SAT}, r \in \mathcal{N}, m \in \mathsf{mon}(\Phi)$ and $s \in \mathcal{N} \to \Phi(\nu(\Phi) + \mathcal{N})$ then $\mathsf{CoRec}(m, s, r) \in \nu(\Phi)$.*
10. *If $m \in \mathsf{mon}(\Phi)$ and $r \in \Phi(\nu(\Phi))$ then $\mathsf{out}^{-1}(m, r) \in \nu(\Phi)$.*

Proof.

1. Is clear.
2. By lemma 2.6 we have $\Psi_I\big(\mu(\Phi)\big) \subseteq \mu(\Phi)$. The claim follows from lemma 2.4.
3. Analogous to 4.
4. By lemma 2.7 we have $\mu(\Phi) \subseteq \Psi_E\big(\mu(\Phi)\big)$. The claim follows from lemma 2.5.
5. Analogous to 4.
6. Is clear.
7. By lemma 2.15 we have $\nu(\Phi) \subseteq \Psi_E\big(\nu(\Phi)\big)$. The claim follows from lemma 2.12.
8. Analogous to 9.
9. By lemma 2.14 we have $\Psi_I\big(\nu(\Phi)\big) \subseteq \nu(\Phi)$. The claim follows from lemma 2.13.
10. Analogous to 9.

$\dashv$

Definition 2.5 (Strong Computability Predicates) *We add the following to the definition of* $\mathsf{SC}^\rho[\Gamma]$*:*

$$\mathsf{SC}^{\mu\alpha\rho}[\Gamma] := \mu(\Phi_\Gamma^{\lambda\alpha\rho})$$

$$\mathsf{SC}^{\nu\alpha\rho}[\Gamma] := \nu(\Phi_\Gamma^{\lambda\alpha\rho})$$

where $\Phi_\Gamma^{\lambda\alpha\rho} : \mathsf{SAT} \to \mathsf{SAT}$ *is defined as:*

$$\Phi_\Gamma^{\lambda\alpha\rho}(\mathcal{M}) := \mathsf{SC}^\rho[\Gamma, \alpha : \mathcal{M}]$$

Lemma 2.16 (Coincidence) *If* $\alpha \notin FV(\rho)$ *then* $\mathsf{SC}^\rho[\Gamma, \alpha : \mathcal{M}] = \mathsf{SC}^\rho[\Gamma]$.
Proof. Induction on ρ.
Case $\rho \equiv \nu\beta\tau$, with $\alpha \notin FV(\nu\beta\tau)$ and $\alpha \neq \beta$. We have $\mathsf{SC}^{\nu\beta\tau}[\Gamma, \alpha : \mathcal{M}] = \nu\big(\Phi^{\lambda\beta\tau}_{\Gamma,\alpha:\mathcal{M}}\big)$ with

$$\Phi^{\lambda\beta\tau}_{\Gamma,\alpha:\mathcal{M}}(\mathcal{N}) = \mathsf{SC}^\tau[\Gamma, \alpha : \mathcal{M}, \beta : \mathcal{N}] \underset{IH\ (\alpha \notin FV(\tau))}{=} \mathsf{SC}^\tau[\Gamma, \beta : \mathcal{N}]$$

On the other hand we have

$$\mathsf{SC}^{\nu\beta\tau}[\Gamma] = \nu\big(\Phi_\Gamma^{\lambda\beta\tau}\big)$$

with $\Phi_\Gamma^{\lambda\beta\tau}(\mathcal{N}) = \mathsf{SC}^\tau[\Gamma, \beta : \mathcal{N}]$. Therefore $\Phi_\Gamma^{\lambda\beta\tau}(\mathcal{N}) = \Phi^{\lambda\beta\tau}_{\Gamma,\alpha:\mathcal{M}}(\mathcal{N})$ for all $\mathcal{N} \in \mathsf{SAT}$ and the claim follows. $\dashv$

Lemma 2.17 (Substitution) $\mathsf{SC}^{\rho[\alpha:=\sigma]}[\Gamma] = \mathsf{SC}^{\rho}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$.
Proof. Induction on ρ. If $\rho = \mu\beta\tau$ then assuming w.l.o.g. $\beta \neq \alpha$ and $\beta \notin FV(\sigma)$ we have $\mathsf{SC}^{(\mu\beta\tau)[\alpha:=\sigma]}[\Gamma] = \mathsf{SC}^{\mu\beta.\tau[\alpha:=\sigma]}[\Gamma] = \mu(\Phi_{\Gamma}^{\lambda\beta.\tau[\alpha:=\sigma]})$, with

$$\Phi_{\Gamma}^{\lambda\beta.\tau[\alpha:=\sigma]}(\mathcal{M}) = \mathsf{SC}^{\tau[\alpha:=\sigma]}[\Gamma, \beta : \mathcal{M}] \underset{IH}{=} \mathsf{SC}^{\tau}[\Gamma, \beta : \mathcal{M}, \alpha : \mathsf{SC}^{\sigma}[\Gamma, \beta : \mathcal{M}]].$$

But observe that as $\beta \notin FV(\sigma)$ by the coincidence lemma we have $\mathsf{SC}^{\sigma}[\Gamma, \beta : \mathcal{M}] = \mathsf{SC}^{\sigma}[\Gamma]$, therefore:

$$\Phi_{\Gamma}^{\lambda\beta.\tau[\alpha:=\sigma]}(\mathcal{M}) = \mathsf{SC}^{\tau}[\Gamma, \beta : \mathcal{M}, \alpha : \mathsf{SC}^{\sigma}[\Gamma]]$$

On the other hand we have $\mathsf{SC}^{\mu\beta\tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma]] = \mu\big(\Phi_{\Gamma,\alpha:\mathsf{SC}^{\sigma}[\Gamma]}^{\lambda\beta\tau}\big)$ where

$$\Phi_{\Gamma,\alpha:\mathsf{SC}^{\sigma}[\Gamma]}^{\lambda\beta\tau}(\mathcal{M}) = \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\sigma}[\Gamma], \beta : \mathcal{M}].$$

Therefore $\Phi_{\Gamma}^{\lambda\beta.\tau[\alpha:=\sigma]}(\mathcal{M}) = \Phi_{\Gamma,\alpha:\mathsf{SC}^{\sigma}[\Gamma]}^{\lambda\beta\tau}(\mathcal{M})$ and we are done. ⊣

Lemma 2.18 (Main Lemma) *If* $\Sigma \rhd r : \rho$ *with* $\Sigma = \{x_1 : \rho_1, \ldots, x_k : \rho_k\}$ *and* $s_i \in \mathsf{SC}^{\rho_i}[\Gamma]$*, for* $1 \leq i \leq k$*, then* $r[\vec{x} := \vec{s}\,] \in \mathsf{SC}^{\rho}[\Gamma]$.
Proof. Induction on $\rhd$. Case (μI). Assume $\Sigma \rhd \mathsf{in}\, t : \mu\alpha\rho$ from $\Sigma \rhd t : \rho[\alpha := \rho]$. Our goal is $(\mathsf{in}\, t)[\vec{x} := \vec{s}] \in \mathsf{SC}^{\mu\alpha\rho}[\Gamma]$, i.e., $\mathsf{in}\, t[\vec{x} := \vec{s}] \in \mu(\Phi_{\Gamma}^{\lambda\alpha\rho})$. Using the proposition 2.2, part 2, it suffices to show $t[\vec{x} := \vec{s}] \in \Phi_{\Gamma}^{\lambda\alpha\rho}\big(\mu(\Phi_{\Gamma}^{\lambda\alpha\rho})\big)$.
Observe that

$$\mathsf{SC}^{\rho[\alpha:=\mu\alpha\rho]}[\Gamma] = \mathsf{SC}^{\rho}[\Gamma, \alpha : \mathsf{SC}^{\mu\alpha\rho}[\Gamma]] = \Phi_{\Gamma}^{\lambda\alpha\rho}\big(\mathsf{SC}^{\mu\alpha\rho}[\Gamma]\big) = \Phi_{\Gamma}^{\lambda\alpha\rho}\big(\mu(\Phi_{\Gamma}^{\lambda\alpha\rho})\big)$$

and by IH we have $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\rho[\alpha:=\mu\alpha\rho]}[\Gamma]$. The claim follows.
Case (νI^{+}). Assume $\Sigma \rhd \mathsf{CoRec}(m, s, t) : \nu\alpha\tau$ from $\Sigma \rhd m : \tau\, \mathsf{mon}\, \alpha$, $\Sigma \rhd s : \sigma \to \tau[\alpha := \nu\alpha\tau + \sigma]$, $\Sigma \rhd t : \sigma$. By IH we have $m[\vec{x} := \vec{s}] \in \mathsf{SC}^{\tau\, \mathsf{mon}\, \alpha}[\Gamma]$, $s[\vec{x} := \vec{s}] \in \mathsf{SC}^{\sigma\to\tau[\alpha:=\nu\alpha\tau+\sigma]}[\Gamma]$, $t[\vec{x} := \vec{s}] \in \mathsf{SC}^{\sigma}[\Gamma]$.
Our goal is $\mathsf{CoRec}\big(m[\vec{x} := \vec{s}], s[\vec{x} := \vec{s}], t[\vec{x} := \vec{s}]\big) \in \mathsf{SC}^{\nu\alpha\tau}[\Gamma] = \nu\big(\Phi_{\Gamma}^{\lambda\alpha\tau}\big)$. By proposition 2.2 it suffices to show

1. $m[\vec{x} := \vec{s}] \in \mathsf{mon}(\Phi_{\Gamma}^{\lambda\alpha\tau})$
2. $s[\vec{x} := \vec{x}] \in \mathsf{SC}^{\sigma}[\Gamma] \to \Phi_{\Gamma}^{\lambda\alpha\tau}\Big(\nu\big(\Phi_{\Gamma}^{\lambda\alpha\tau}\big) + \mathsf{SC}^{\sigma}[\Gamma]\Big)$

For the first part is not difficult to show that

$$\mathsf{SC}^{\tau\, \mathsf{mon}\, \alpha}[\Gamma] = \bigcap_{\mathcal{P},\mathcal{Q}\in\mathsf{SAT}} (\mathcal{P} \to \mathcal{Q}) \to \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathcal{P}] \to \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathcal{Q}] =$$

$$= \bigcap_{\mathcal{P},\mathcal{Q}\in\mathsf{SAT}} (\mathcal{P} \to \mathcal{Q}) \to \Phi_{\Gamma}^{\lambda\alpha\tau}\big(\mathcal{P}\big) \to \Phi_{\Gamma}^{\lambda\alpha\tau}\big(\mathcal{Q}\big) = \mathsf{mon}(\Phi_{\Gamma}^{\lambda\alpha\tau})$$

Therefore the claim follows from the IH.

The second part follows from the IH by observing that

$$\mathsf{SC}^{\tau[\alpha:=\nu\alpha\tau+\sigma]}[\Gamma] = \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\nu\alpha\tau+\sigma}[\Gamma]] = \mathsf{SC}^{\tau}[\Gamma, \alpha : \mathsf{SC}^{\nu\alpha\tau}[\Gamma] + \mathsf{SC}^{\sigma}[\Gamma]] =$$

$$= \mathsf{SC}^{\tau}[\Gamma, \alpha : \nu\big(\Phi_{\Gamma}^{\lambda\alpha\tau}\big) + \mathsf{SC}^{\sigma}[\Gamma]] = \Phi_{\Gamma}^{\lambda\alpha\tau}\Big(\nu\big(\Phi_{\Gamma}^{\lambda\alpha\tau}\big) + \mathsf{SC}^{\sigma}[\Gamma]\Big)$$

$\dashv$

Proposition 2.3 *If* $\Sigma \rhd r : \rho$ *then* $r \in \mathsf{SN}$.
Proof. The same as for proposition 1.8. $\dashv$

Terms in SN are Strongly Normalizing

Lemma 2.19 *If* $m, s, E[x] \in \mathsf{sn}$ *then* $\mathsf{It}(m, s, E[x]) \in \mathsf{sn}$ *and* $\mathsf{Rec}(m, s, E[x]) \in \mathsf{sn}$.
Proof. Induction on $m, s, E[x] \in \mathsf{sn}$. $\dashv$

Lemma 2.20 *If* $m, E[x] \in \mathsf{sn}$ *then* $\mathsf{in}^{-1}(m, E[x]) \in \mathsf{sn}$
Proof. Induction on $m, E[x] \in \mathsf{sn}$ $\dashv$

Lemma 2.21 *If* $E[x] \in \mathsf{sn}$ *then* $\mathsf{out}\, E[x] \in \mathsf{sn}$
Proof. Induction on $E[x] \in \mathsf{sn}$ $\dashv$

Lemma 2.22 *If* $t \in \mathsf{sn}$ *then* $\mathsf{in}\, t \in \mathsf{sn}$
Proof. Induction on $t \in \mathsf{sn}$. $\dashv$

Lemma 2.23 $IfE\big[s\Big(m\big(\lambda x.\mathsf{It}(m, s, x)\big)t\Big)\big] \in \mathsf{sn}$ *then* $E\big[\mathsf{It}(m, s, \mathsf{in}\, t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[s\Big(m\big(\lambda x.\mathsf{It}(m, s, x)\big)t\Big)\big] \in$[1] sn^{+}. $\dashv$

Lemma 2.24 *If* $E\big[s\Big(m\big(\langle\mathsf{Id}, \lambda x.\mathsf{Rec}(m, s, x)\rangle\big)t\Big)\big] \in \mathsf{sn}$ *then* $E\big[\mathsf{Rec}(m, s, \mathsf{in}\, t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[s\Big(m\big(\langle\mathsf{Id}, \lambda x.\mathsf{Rec}(m, s, x)\rangle\big)t\Big)\big] \in \mathsf{sn}^{+}$. $\dashv$

Lemma 2.25 *If* $E\big[m(\lambda zz)t\big] \in \mathsf{sn}$ *then* $E\big[\,\mathsf{in}^{-1}(m, \mathsf{in}\, t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[m(\lambda zz)t\big] \in \mathsf{sn}$ $\dashv$

Lemma 2.26 *If* $m, s, t \in \mathsf{sn}$ *then* $\mathsf{CoIt}(m, s, t) \in \mathsf{sn}$ *and* $\mathsf{CoRec}(m, s, t) \in \mathsf{sn}$
Proof. Induction on $m, s, t \in \mathsf{sn}$. $\dashv$

Lemma 2.27 *If* $m, t \in \mathsf{sn}$ *then* $\mathsf{out}^{-1}(m, t) \in \mathsf{sn}$
Proof. Induction on $m, t \in \mathsf{sn}$ $\dashv$

Lemma 2.28 *If* $E\big[m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)(st)\big] \in \mathsf{sn}$ *then* $E\big[\,\mathsf{out}\,\mathsf{CoIt}(m, s, t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[m\big(\lambda z.\mathsf{CoIt}(m, s, z)\big)(st)\big] \in \mathsf{sn}^{+}$ $\dashv$

[1] The set sn^{+} is defined as sn but with the relation $\to_{\beta}^{+}$. The proof needs it as there are two occurrences of s in the canonical reduct of $E[\mathsf{It}(m, s, \mathsf{in}\, t)]$. On the other hand it is easy to prove that $\mathsf{sn} = \mathsf{sn}^{+}$.

Lemma 2.29 *If* $E\big[m\big([\mathsf{Id}, \lambda z.\mathsf{CoRec}(m,s,z)]\big)(st)\big] \in \mathsf{sn}$ *then* $E\big[\,\mathsf{out}\,\mathsf{CoRec}(m,s,t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[m\big([\mathsf{Id}, \lambda z.\mathsf{CoRec}(m,s,z)]\big)(st)\big] \in \mathsf{sn}^{+}$ ⊣

Lemma 2.30 *If* $E\big[m(\lambda zz)t\big] \in \mathsf{sn}$ *then* $E\big[\,\mathsf{out}\,\mathsf{out}^{-1}(m,t)\big] \in \mathsf{sn}$
Proof. Induction on $E\big[m(\lambda zz)t\big] \in \mathsf{sn}$ ⊣

Proposition 2.4 $\mathsf{SN} \subseteq \mathsf{sn}$
Proof. Proposition 1.9 and the above lemmas show that sn is closed under the defining rules of SN, therefore the claim follows by minimality of SN. ⊣

Proposition 2.5 MICT *is strongly normalizing.*
Proof. Immediate from propositions 2.3 and 2.4. ⊣

2.3 The System MCICT

This is an extension of F with initial/final dialgebras, represented by clausular (co)inductive types, and only conventional (co)induction principles taken from section 2.1.2.

2.3.1 Definition of the System

We add the following to system $\mathsf{F}^{+,\times}$:

- If α is a type variable and $\rho_1, \ldots, \rho_k$ are types then

$$\mu\alpha(\rho_1, \ldots, \rho_k),\ \nu\alpha(\rho_1, \ldots, \rho_k)$$

  are types. Where each ρ_i is called a clause.

- If $\vec{m}, r, \vec{s}, t$ are terms and $k, i \in \mathbb{N}$ with $i \le k$ then

$$\mathsf{in}_{k,i}\, t,\ \mathsf{in}_k^{-1}(\vec{m}, t),\ \mathsf{It}_k(\vec{m}, \vec{s}, t),\ \mathsf{Rec}_k(\vec{m}, \vec{s}, t)$$

$$\mathsf{CoIt}_k(\vec{m}, \vec{s}, t), \mathsf{CoRec}_k(\vec{m}, \vec{s}, t), \mathsf{out}_k^{-1}(\vec{m}, \vec{t}), \mathsf{out}_{k,i}\, t$$

  are terms.

We extend the typing relation with eight rules:

$$\frac{\Sigma \rhd t : \rho_i[\alpha := \mu\alpha(\rho_1, \ldots, \rho_k)]}{\Sigma \rhd \mathsf{in}_{k,i}\, t : \mu\alpha(\rho_1, \ldots, \rho_k)}\ (\mu I)$$

$$\frac{\begin{array}{l}\Sigma \rhd t : \mu\alpha(\rho_1, \ldots, \rho_k)\\ \Sigma \rhd m_i : \rho_i\,\mathsf{mon}\,\alpha\ \ 1 \le i \le k\\ \Sigma \rhd s_i : \rho_i[\alpha := \sigma] \to \sigma\ \ 1 \le i \le k\end{array}}{\Sigma \rhd \mathsf{It}_k(\vec{m}, \vec{s}, t) : \sigma}\ (\mu E)$$

$$\dfrac{\begin{array}{l}\Sigma \rhd t : \mu\alpha(\rho_1,\ldots,\rho_k)\\ \Sigma \rhd m_i : \rho_i\ \mathsf{mon}\ \alpha \ \ 1 \leq i \leq k\\ \Sigma \rhd s_i : \rho_i[\alpha := \mu\alpha(\rho_1,\ldots,\rho_k)\times\sigma] \to \sigma \ \ 1 \leq i \leq k\end{array}}{\Sigma \rhd \mathsf{Rec}_k(\vec{m},\vec{s},t) : \sigma}\ (\mu E^+)$$

$$\dfrac{\begin{array}{l}\Sigma \rhd s_i : \sigma \to \rho_i[\alpha := \sigma] \ \ 1 \leq i \leq k\\ \Sigma \rhd m_i : \rho_i\ \mathsf{mon}\ \alpha \ \ 1 \leq i \leq k\\ \Sigma \rhd t : \sigma\end{array}}{\Sigma \rhd \mathsf{CoIt}_k(\vec{m},\vec{s},t) : \nu\alpha(\rho_1,\ldots,\rho_k)}\ (\nu I)$$

$$\dfrac{\begin{array}{l}\Sigma \rhd s_i : \sigma \to \rho_i[\alpha := \nu\alpha(\rho_1,\ldots,\rho_k)+\sigma] \ \ 1 \leq i \leq k\\ \Sigma \rhd m_i : \rho_i\ \mathsf{mon}\ \alpha \ \ 1 \leq i \leq k\\ \Sigma \rhd t : \sigma\end{array}}{\Sigma \rhd \mathsf{CoRec}_k(\vec{m},\vec{s},t) : \nu\alpha(\rho_1,\ldots,\rho_k)}\ (\nu I^+)$$

$$\dfrac{\Sigma \rhd r : \nu\alpha(\rho_1,\ldots,\rho_k)}{\Sigma \rhd \mathsf{out}_{k,i}\, r : \rho_i[\alpha := \nu\alpha(\rho_1,\ldots,\rho_k)]}\ (\nu E)$$

The last two rules deserve a detailed discussion

The Principles of (Co)inductive Inversion

Inductive Inversion

Equation (1.20) is not suitable to be represented directly in our framework. The reason is that there is no satisfactory way to represent the tuples of objects $\langle F_1\mu,\ldots,F_k\mu\rangle$ Observe that the inverse in section 2.1.2 is a function $\mathsf{in}_k^{-1} : \langle\mu,\ldots,\mu\rangle \to \langle F_1\mu,\ldots,F_k\mu\rangle$ such that

$$\mathsf{in}_k^{-1}\circ\langle\mathsf{in}_{k,1},\ldots,\mathsf{in}_{k,k}\rangle = \mathsf{Id}_{\langle F_1\mu,\ldots,F_k\mu\rangle}$$

So that we would need a rule like this:

$$\dfrac{\Sigma \rhd t : \langle\mu\alpha(\rho_1,\ldots,\rho_k),\ldots,\mu\alpha(\rho_1,\ldots,\rho_k)\rangle \quad \Sigma \rhd m_i : \rho_i\ \mathsf{mon}\ \alpha\ 1 \leq i \leq k}{\Sigma \rhd \mathsf{in}_k^{-1}(\vec{m},t) : \langle\rho_1[\alpha := \mu\alpha(\rho_1,\ldots,\rho_k)],\ldots,\rho_k[\alpha := \mu\alpha(\rho_1,\ldots,\rho_k)]\rangle}$$

Of course we would need to give sense to a tuple of objects as a type, but this would complicate the system only to be able to model this principle.

On the other hand the main application of such rule is to define inductive destructors following the reasoning:

> "If we have an inductive object $t : \mu\alpha(\rho_1,\ldots,\rho_k)$ then it was generated by a clause $\mathsf{in}_k^{-1}\, t : \rho_i[\alpha := \mu\alpha(\rho_1,\ldots,\rho_k)]$ for some $1 \leq i \leq k$".

which implies, for instance, the fact that if t is a natural number then t is either 0 or a succesor sn.

This reasoning corresponds to an inverse $\mathsf{in}_k^{-1} : \mu \to F_1\mu + \ldots + F_k\mu$ such that

$$\mathsf{in}_k^{-1}(\vec{m}, \mathsf{in}_{k,i}\, t) \;=\; \mathsf{inj}_i^k \left(m_i(\lambda z.z)t\right)$$

where inj_i^k is the canonical ith-injection.

We model this kind of inverse instead of the one given by equation (1.20).

$$\frac{\Sigma \rhd t : \mu\alpha(\rho_1,\dots,\rho_k) \quad \Sigma \rhd m_i : \rho_i \,\mathsf{mon}\, \alpha \;\; 1 \le i \le k}{\Sigma \rhd \mathsf{in}_k^{-1}(\vec{m}, t) : \rho_1[\alpha := \mu\alpha(\rho_1,\dots,\rho_k)] + \ldots + \rho_k[\alpha := \mu\alpha(\rho_1,\dots,\rho_k)]} \;(\mu E^i)$$

The main application of this rule can be easily achieved using primitive recursion, so that we will omit the rule in later systems as it would cause more problems than profits. One of the main disadvantages of this rule is that generates a term inhabiting a sum type in an unusual way. So that inhabitants of sum types are not only generated by the typing rules for sums.

Coinductive Inversion

Analogously the rule corresponding to equation (1.17) would be:

$$\frac{\begin{array}{l}\Sigma \rhd t : \langle \rho_1[\alpha := \nu\alpha(\rho_1,\dots,\rho_k)],\dots,\rho_k[\alpha := \nu\alpha(\rho_1,\dots,\rho_k)]\rangle \\ \Sigma \rhd m_i : \rho_i \,\mathsf{mon}\, \alpha \;\; 1 \le i \le k\end{array}}{\Sigma \rhd \mathsf{out}_k^{-1}(\vec{m}, t) : \langle \nu\alpha(\rho_1,\dots,\rho_k),\dots,\nu\alpha(\rho_1,\dots,\rho_k)\rangle}$$

Instead we use a rule able to construct coinductive objects following the reasoning:

"If we have all pieces $t_i : \rho_i[\alpha := \nu\alpha(\rho_1,\dots,\rho_k)]$ for $1 \le i \le k$ then we can construct a coinductive object $\mathsf{out}_k^{-1}(\vec{m}, \vec{t}) : \nu\alpha(\rho_1,\dots,\rho_k)$."

For instance using this principle we can construct a stream given its head and tail.

This principle corresponds to an "inverse" $\mathsf{out}_k^{-1} : \langle F_1\nu,\dots,F_k\nu\rangle \to \nu$ such that

$$\mathsf{out}_{k,i}\, \mathsf{out}_k^{-1}(\vec{m}, \vec{t}) \;=\; m_i(\lambda z.z)t_i$$

Therefore we arrive to this rule:

$$\frac{\begin{array}{l}\Sigma \rhd t_i : \rho_i[\alpha := \nu\alpha(\rho_1,\dots,\rho_k)] \;\; 1 \le i \le k \\ \Sigma \rhd m_i : \rho_i \,\mathsf{mon}\, \alpha \;\; 1 \le i \le k\end{array}}{\Sigma \rhd \mathsf{out}_k^{-1}(\vec{m}, \vec{t}) : \nu\alpha(\rho_1,\dots,\rho_k)} \;(\nu I^i)$$

To finish the definition of this system we add six rules to the β-reduction relation, which are generated by the equalities in sections 2.1.2 and 2.3.1.

$$\begin{array}{rcl}
\mathsf{It}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t) & \mapsto_\beta & s_i\Big(m_i\big(\lambda x.\mathsf{It}_k(\vec{m},\vec{s},x)\big)t\Big) \\
\mathsf{Rec}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t) & \mapsto_\beta & s_i\Big(m_i\Big(\langle\mathsf{Id},\lambda z.\mathsf{Rec}_k(\vec{m},\vec{s},z)\rangle\Big)t\Big) \\
\mathsf{in}_k^{-1}(\vec{m},\mathsf{in}_{k,i}\,t) & \mapsto_\beta & \mathsf{inj}_i^k\,\big(m_i(\lambda z.z)t\big) \\
\mathsf{out}_{k,i}\,\mathsf{CoIt}_k(\vec{m},\vec{s},t) & \mapsto_\beta & m_i\Big(\lambda z.\mathsf{CoIt}_k(\vec{m},\vec{s},z)\Big)(s_i t) \\
\mathsf{out}_{k,i}\,\mathsf{CoRec}_k(\vec{m},\vec{s},t) & \mapsto_\beta & m_i\Big([\mathsf{Id},\lambda z.\mathsf{CoRec}_k(\vec{m},\vec{s},z)]\Big)(s_i t) \\
\mathsf{out}_{k,i}\,\mathsf{out}_k^{-1}(\vec{m},\vec{t}) & \mapsto_\beta & m_i(\lambda z.z)t_i
\end{array}$$

This finish the definition of the system MCICT. A system of Monotone and Clausular Inductive and Coinductive Types.

Subject Reduction for MCICT

To prove this property suffices to simplify the proof for the logic MCICD presented in section 4.1.3.

The subsystem without inductive inversion will play an important role later and will be denoted MCICT^-

The Natural Numbers in MCICT

The natural numbers are defined as follows:

$$\mathsf{nat} := \mu\alpha(1,\alpha)$$

This time the constructors are defined directly:

$$0 := \mathsf{in}_{2,1}\star \qquad s := \lambda x.\mathsf{in}_{2,2}x$$

Streams in MCICT

$$\mathsf{stream}(\rho) := \nu\alpha(\rho,\alpha)$$

The destructors are defined directly as

$$\mathsf{head} := \lambda x.\mathsf{out}_{2,1}x \qquad \mathsf{tail} := \lambda x.\mathsf{out}_{2,2}x$$

Degenerated Types

The degenerated types $\mu\alpha(), \nu\alpha()$ having no clauses can be considered as the empty and the unit type respectively. Setting $\mathsf{0} := \mu\alpha()$ we have no constructors but the degenerated iteration principle gives a derived rule

$$\frac{t : \mathsf{0}}{\mathsf{It}_0(t) : \sigma}\ (0E)$$

for every type σ. Of course $\mathsf{0}$ cannot be inhabited.

Analogusly setting $\mathsf{1} := \nu\alpha()$ there are no destructors but the coiteration principle degenerates to the following

$$\frac{t : \sigma}{\mathsf{CoIt}_0(t) : \mathsf{1}}$$

for every type σ. In some sense there is only one inhabitant of $\mathsf{1}$ as the term t does not play an important role, just to fix the definition we set $\star := \mathsf{CoIt}_0(\lambda x x)$, so that we have the usual rule:

$$\frac{}{\star : \mathsf{1}}$$

More (Co)inductive Types in MCICT

- Lists of objects of type ρ: $\mathsf{list}(\rho) := \mu\alpha(1, \rho \times \alpha)$
- Well-founded ρ-branching trees: $\mathsf{tree}(\rho) := \mu\alpha(1, \rho \to \alpha)$
- Infinite depth ρ-labelled trees: $\mathsf{inftree}(\rho) := \nu\alpha(\rho, \mathsf{list}(\alpha))$

with $\alpha \notin FV(\rho)$ in all cases.

On Sum and Product Types

Our type system MCICT has a strong expressive power, so that we could even get rid of sums and products as basic type constructors, in the following way:

$$\rho + \sigma := \mu\alpha(\rho, \sigma) \qquad \rho \times \sigma := \nu\alpha(\rho, \sigma)$$

with $\alpha \notin FV(\rho, \sigma)$

The constructors for the sum are $\mathsf{inl} := \lambda x.\, \mathsf{in}_{2,1}\, x, \mathsf{inr} := \lambda x.\, \mathsf{in}_{2,2}\, x$, analogously the destructors for the product are $\pi_1 := \lambda x.\, \mathsf{out}_{2,1}\, x, \pi_2 := \lambda x.\, \mathsf{out}_{2,2}$. The pair and case analysis are defined as:

$$\begin{array}{rcl} \langle r, s\rangle & := & \mathsf{out}_2^{-1}(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, r, s) \\ \mathsf{case}(r, x.s, y.t) & := & \mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, \lambda x.s, \lambda y.t, r) \end{array}$$

The reader can verify that the β-reduction rules from page 15 still hold. The only reason to consider $\times$ and $+$ as basic type constructors is to avoid problems (ad-hoc definitions) in the embedding from MCICT into MICT presented in next section.

2.3.2 Strong Normalization of MCICT

As usual, this is achieved by means of an embedding, this time into the already strongly normalizing system MICT.

From now on we agree to associate sum and product to the right, that is,

$$\begin{array}{l} \rho_1 + \ldots + \rho_k := \rho_1 + (\rho_2 + (\ldots + \rho_k)\ldots) \\ \rho_1 \times \ldots \times \rho_k := \rho_1 \times (\rho_2 \times (\ldots \times \rho_k)\ldots) \end{array}$$

Definition 2.6 *The following syntactic sugar will be useful, where $k \geq 2$:*

$$\begin{array}{rcl} \mathsf{inj}^k_j & := & \lambda z.\,\mathsf{inr}^{j-1}(\mathsf{inl}\, z), \;\; 1 \leq j < k \\ \mathsf{inj}^k_k & := & \lambda z.\,\mathsf{inr}^{k-1}\, z \\ \pi_{k,j} & := & \lambda z.\pi_1({\pi_2}^{j-1} z), \;\; 1 \leq j < k \\ \pi_{k,k} & := & \lambda z.{\pi_2}^{k-1} z \end{array}$$

These are, of course, the canonical injections and projections for a k-sum and k-product.

Definition 2.7 (MICT) *Given variables $x_1, \ldots, x_k, y_1, \ldots, y_k, f, u, v, w, z$ we define, for $k \geq 2$ and $1 \leq i \leq k$, the following families of terms t_i, r_i, q_i, p_i:*

$$\begin{array}{rcl} t_j[u] & := & \mathsf{inj}^k_j(x_j f u) \;\;\; 1 \leq j \leq k \\ & & \\ r_0[v] & := & t_k[v] \\ r_{j+1}[v] & := & \mathsf{case}(v, x.t_{k-(j+1)}[x], y.r_j[y]) \;\;\; 0 \leq j < k-1 \\ & & \\ q_0[w] & := & y_k w \\ q_{j+1}[w] & := & \mathsf{case}(w, x.y_{k-(j+1)}x, y.q_j[y]) \;\;\; 0 \leq j < k-1 \\ & & \\ p_j[z] & := & x_j f(\pi_{k,j} z) \;\;\; 1 \leq j \leq k \end{array}$$

Observe that

$$\begin{array}{rcl} FV(t_i[u]) & = & \{x_i, f, u\} \\ FV(r_i[v]) & = & \{x_{k-i}, \ldots, x_k, f, v\} \\ FV(q_i[w]) & = & \{y_{k-i}, \ldots, y_k, w\} \\ FV(p_i[z]) & = & \{x_i, f, z\} \end{array}$$

Definition 2.8 *Given variables $\vec{x}, \vec{y}$ with $|\vec{x}| = |\vec{y}| = k$ define the following terms:*

$$\begin{array}{rcl} \mathbb{M}^+[\vec{x}\,] & := & \lambda f \lambda z.r_{k-1}[z] \\ \mathbb{S}^+[\vec{y}\,] & := & \lambda w.q_{k-1}[w] \\ \mathbb{M}^\times[\vec{x}\,] & := & \lambda f.\lambda z.\langle p_1[z], \ldots, p_k[z]\rangle \\ \mathbb{S}^\times[\vec{y}\,] & := & \lambda w.\langle y_1 w, \ldots, y_k w\rangle \end{array}$$

Observe that

$$\begin{array}{c} FV(\mathbb{M}^+[\vec{x}\,]) = FV(\mathbb{M}^\times[\vec{x}\,]) = \vec{x} \\ FV(\mathbb{S}^+[\vec{y}\,]) = FV(\mathbb{S}^\times[\vec{x}\,]) = \vec{y} \end{array}$$

These terms will be needed for the embedding of (co)iterators, (co)recursors and in / out functions, the next proposition give us its needed typings.

Proposition 2.6 *Given types $\mu := \mu\alpha.\rho_1 + \ldots + \rho_k, \nu := \nu\alpha.\rho_1 \times \ldots \times \rho_k$ and contexts*

$$\begin{array}{rcl}
\mathbb{\Gamma} & := & \{f : \alpha \to \beta, z : \rho_1 + \ldots + \rho_k\} \\
\mathbb{\Gamma}' & := & \{f : \alpha \to \beta, z : \rho_1 \times \ldots \times \rho_k\} \\
\mathbb{\Pi} & := & \{x_i : \rho_i \,\mathsf{mon}\, \alpha\} \;\; 1 \leq i \leq k \\
\mathbb{\Sigma} & := & \{y_i : \rho_i[\alpha := \gamma] \to \gamma\} \;\; 1 \leq i \leq k \\
\mathbb{\Sigma}' & := & \{y_i : \gamma \to \rho_i[\alpha := \gamma]\} \;\; 1 \leq i \leq k \\
\mathbb{\Delta} & := & \{z_i : \rho_i[\alpha := \mu \times \gamma] \to \gamma\} \;\; 1 \leq i \leq k \\
\mathbb{\Delta}' & := & \{z_i : \gamma \to \rho_i[\alpha := \nu + \gamma]\} \;\; 1 \leq i \leq k
\end{array}$$

we have the following typings

$$\begin{array}{l}
\mathbb{\Gamma}, x_j : \rho_j \,\mathsf{mon}\, \alpha, u : \rho_j \rhd t_j[u] : \rho_1[\alpha := \beta] + \ldots + \rho_k[\alpha := \beta] \\
\mathbb{\Gamma}, \mathbb{\Pi}, v : \rho_{k-j} + \ldots + \rho_k \rhd r_j[v] : \rho_1[\alpha := \beta] + \ldots + \rho_k[\alpha := \beta] \\
\mathbb{\Sigma}, w : (\rho_{k-j} + \ldots + \rho_k)[\alpha := \gamma] \rhd q_j[w] : \gamma \\
\mathbb{\Delta}, w : (\rho_{k-j} + \ldots + \rho_k)[\alpha := \mu \times \gamma] \rhd q_j[w] : \gamma \\
\mathbb{\Gamma}', x_j : \rho_j \,\mathsf{mon}\, \alpha \rhd p_j[z] : \rho_j[\alpha := \beta] \quad \textit{for } 1 \leq j \leq k \\
\mathbb{\Pi} \rhd \mathbb{M}^+[\vec{x}] : (\rho_1 + \ldots + \rho_k) \,\mathsf{mon}\, \alpha \\
\mathbb{\Pi} \rhd \mathbb{M}^\times[\vec{x}] : (\rho_1 \times \ldots \times \rho_k) \,\mathsf{mon}\, \alpha \\
\mathbb{\Sigma} \rhd \mathbb{S}^+[\vec{y}] : (\rho_1 + \ldots + \rho_k)[\alpha := \gamma] \to \gamma \\
\mathbb{\Sigma}' \rhd \mathbb{S}^\times[\vec{y}] : \gamma \to (\rho_1 \times \ldots \times \rho_k)[\alpha := \gamma] \\
\mathbb{\Delta} \rhd \mathbb{S}^+[\vec{z}] : (\rho_1 + \ldots + \rho_k)[\alpha := \mu \times \gamma] \to \gamma \\
\mathbb{\Delta}' \rhd \mathbb{S}^\times[\vec{z}] : \gamma \to (\rho_1 \times \ldots \times \rho_k)[\alpha := \nu + \gamma]
\end{array}$$

Proof. Straightforward. ⊣

Proposition 2.7 *For every term t, s and for $0 \leq i < k-1$ we have the following reductions:*

$$\begin{array}{c}
r_{i+1}[\mathsf{inr}^{j+1}t] \to^+ r_i[\mathsf{inr}^j] \\
q_{i+1}[\mathsf{inr}^{j+1}s] \to^+ q_i[\mathsf{inr}^j s]
\end{array}$$

and therefore for $2 \leq j \leq k$:

$$\begin{array}{c}
r_{k-2}[\mathsf{inr}^{j-2}t] \to^\star r_{k-j}[t] \\
q_{k-2}[\mathsf{inr}^{j-2}s] \to^\star q_{k-j}[s]
\end{array}$$

Proof.

$$\begin{array}{ll}
r_{i+1}[\mathsf{inr}^{j+1}t] & \to \mathsf{case}(\mathsf{inr}^{\,j+1}t, x.t_{k-(i+1)}[x], y.r_i[y]) \\
 & \equiv \mathsf{case}(\mathsf{inr}(\mathsf{inr}^{\,j}t).x.t_{k-(i+1)}[x], y.r_i[y]) \to r_i[\mathsf{inr}^j t] \\
q_{i+1}[\mathsf{inr}^{j+1}s] & \to \mathsf{case}(\mathsf{inr}^{\,j+1}s, x.y_{k-(i+1)}x, y.q_i[y]) \\
 & \equiv \mathsf{case}(\mathsf{inr}(\mathsf{inr}^{\,j}s), x.y_{k-(i+1)}x, y.q_i[y]) \to q_i[\mathsf{inr}^j s]
\end{array}$$

⊣

Proposition 2.8 *For $k \geq 2$ and every $1 \leq i \leq k$ we have*

$$r_{k-1}[\mathsf{inj}_i^k\, s] \to_\beta^+ t_i[s]$$

Proof. By case analysis on i and proposition 2.7. ⊣

Definition 2.9 *The embedding* $(\cdot)'$: $\mathsf{MCICT} \to \mathsf{MICT}$ *is defined in two parts, first we define it for the special cases of empty and unit types which are special encoded types. Then we give the general definition which excludes the previous cases.*

$$\begin{array}{rcl}
(\mu\alpha())' & := & \forall\alpha\alpha \\
\mathsf{It}_0(t)' & := & t' \\
\mathsf{Rec}_0(t)' & := & t' \\
(\nu\alpha())' & := & \forall\alpha.\alpha \to \alpha \\
\mathsf{CoIt}_0(t)' & := & \lambda zz \\
\mathsf{CoRec}_0(t)' & := & \lambda zz
\end{array}$$

Next the general definition where $k \geq 1$

$$\begin{array}{rcl}
\alpha' & := & \alpha \\
(\sigma \to \rho)' & := & \sigma' \to \rho' \\
(\forall\alpha\rho)' & := & \forall\alpha.\rho' \\
(\rho \times \sigma)' & := & \rho' \times \sigma' \\
(\rho + \sigma)' & := & \rho' + \sigma' \\
\big(\mu\alpha(\rho_1, \ldots, \rho_k)\big)' & := & \mu\alpha.\rho'_1 + \ldots + \rho'_k \\
\big(\nu\alpha(\rho_1, \ldots, \rho_k)\big)' & := & \nu\alpha.\rho'_1 \times \ldots \times \rho'_k \\
x' & := & x \\
(\lambda xr)' & := & \lambda x.r' \\
(rs)' & := & r's' \\
\langle r, s\rangle' & := & \langle r', s'\rangle \\
(\pi_1 r)' & := & \pi_1 r' \\
(\pi_2 r)' & := & \pi_2 r' \\
(\mathsf{inl}\, r)' & := & \mathsf{inl}\, r' \\
(\mathsf{inr}\, r)' & := & \mathsf{inr}\, r' \\
\big(\mathsf{case}(r, x.s, y.t)\big)' & := & \mathsf{case}(r', x.s', y.t') \\
\mathsf{in}_{1,1}\, t' & := & \mathsf{in}\, t' \\
\mathsf{in}_{k,i}\, t' & := & \mathsf{in}(\mathsf{inj}_i^k\, t')\ k \geq 2 \\
\mathsf{in}_k^{-1}(\vec{m}, t)' & := & \mathsf{in}^{-1}(\mathbb{M}^+[\vec{m}'], t') \\
\mathsf{It}_1(m, s, t)' & := & \mathsf{It}(m', s', t') \\
\mathsf{It}_k(\vec{m}, \vec{s}, t)' & := & \mathsf{It}(\mathbb{M}^+[\vec{m}'], \mathbb{S}^+[\vec{s}\,'], t') \quad k \geq 2 \\
\mathsf{Rec}_1(m, s, t)' & := & \mathsf{Rec}(m', s', t') \\
\mathsf{Rec}_k(\vec{m}, \vec{s}, t)' & := & \mathsf{Rec}(\mathbb{M}^+[\vec{m}'], \mathbb{S}^+[\vec{s}\,'], t') \quad k \geq 2 \\
(\mathsf{out}_{1,1}\, t)' & := & \mathsf{out}\, t' \\
(\mathsf{out}_{k,i}\, t)' & := & \pi_{k,i}(\mathsf{out}\, t')\ k \geq 2 \\
\mathsf{out}_k^{-1}(\vec{m}, \vec{t}\,)' & := & \mathsf{out}^{-1}(\mathbb{M}^\times[\vec{m}'], \langle t'_1, \ldots, t'_k\rangle) \\
\mathsf{CoIt}_1(m, s, t)' & := & \mathsf{CoIt}(m', s', t') \\
\mathsf{CoIt}_k(\vec{m}, \vec{s}, t)' & := & \mathsf{CoIt}(\mathbb{M}^\times[\vec{m}'], \mathbb{S}^\times[\vec{s}\,'], t') \quad k \geq 2 \\
\mathsf{CoRec}_1(m, s, t)' & := & \mathsf{CoRec}(m', s', t') \\
\mathsf{CoRec}_k(\vec{m}, \vec{s}, t)' & := & \mathsf{CoRec}(\mathbb{M}^\times[\vec{m}'], \mathbb{S}^\times[\vec{s}\,'], t') \quad k \geq 2
\end{array}$$

where the terms $\mathbb{M}^+, \mathbb{M}^\times, \mathbb{S}^+, \mathbb{S}^\times$ *are taken from definition 2.8.*

Proposition 2.9 (Substitution Properties) *The following holds:*

- $(\rho[\alpha := \sigma])' = \rho'[\alpha := \sigma']$
- $r[x := s]' = r'[x := s']$
- $(\rho \operatorname{\mathsf{mon}} \alpha)' = \rho' \operatorname{\mathsf{mon}} \alpha$

Proof. The first part by induction on ρ, the second part by induction on r, the third part is immediate from the first part. ⊣

Proposition 2.10 *If* $\mathbb{\Sigma} \rhd_{\mathsf{MCICT}} r : \sigma$ *then* $\mathbb{\Sigma}' \rhd_{\mathsf{MICT}} r' : \sigma'$.

Proof. Induction on $\rhd_{\mathsf{MCICT}}$. ⊣

Proposition 2.11 *If* $r \to_\beta s$ *in* MCICT *then* $r' \to_\beta^+ s'$ *in* MICT.

Proof. Induction on $\to_\beta$ in MCICT.
Case $\mathsf{It}_k(\vec{m}, \vec{s}, \mathsf{in}_{k,i}\, t) \mapsto_\beta s_i\Big(m_i\big(\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x)\big)t\Big)$
The cases for $k = 0, 1$ are trivial. Assume $k \geq 2$. Set $\mathbb{M} := \mathbb{M}^+[m'_1, \ldots, m'_k], \mathbb{S} := \mathbb{S}^+[s'_1, \ldots, s'_k]$ taken from definition 2.8. We have three subcases:

- Subcase $i = 1$.

$$\Big(\mathsf{It}_k(\vec{m}, \vec{s}, \mathsf{in}_{k,1}\, t)\Big)' \equiv \mathsf{It}(\mathbb{M}, \mathbb{S}, (\mathsf{in}_{k,1}\, t)') \equiv \mathsf{It}\big(\mathbb{M}, \mathbb{S}, \mathsf{in}(\mathsf{inj}^k_1\, t')\big) \to$$

$$\mathbb{S}\big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))(\mathsf{inl}\, t')\big) \to \mathsf{case}\Big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))(\mathsf{inl}\, t'), x.s'_1 x, y.q_{k-2}[y]\Big)$$

$$\mathsf{case}\Big(\mathsf{case}(\mathsf{inl}\, t', x.t_1[x], y.r_{k-2}[y]), x.s'_1 x, y.q_{k-2}[y]\Big) \to$$

$$\mathsf{case}\Big(t_1[t'], x.s'_1 x, y.q_{k-2}[y]\Big) \to \mathsf{case}\Big(\, \mathsf{inl}(m'_1(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'), x.s'_1 x, y.q_{k-2}[y]\Big)$$

$$s'_1\Big(m'_1(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'\Big) \equiv \Big(s_1\Big(m_1(\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x))t\Big)\Big)'.$$

- ○ Subcase $1 < i < k$.

$$
\begin{aligned}
\Big(\mathsf{It}_k(\vec{m}, \vec{s}, \mathsf{in}_{k,i}\, t)\Big)' \equiv \mathsf{It}(\mathbb{M}, \mathbb{S}, (\mathsf{in}_{k,i}\, t)') &\quad \equiv \\
\mathsf{It}(\mathbb{M}, \mathbb{S}, \mathsf{in}(\mathsf{inj}_i^k\, t')) &\quad \to \\
\mathbb{S}\Big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))(\mathsf{inj}_i^k\, t')\Big) &\quad \to \\
\mathsf{case}\Big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))(\mathsf{inj}_i^k\, t'), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \to \\
\mathsf{case}\Big(\mathsf{case}(\mathsf{inr}^{i-1}(\mathsf{inl}\, t'), x.t_1[x], y.r_{k-2}[y]), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \equiv \\
\mathsf{case}\Big(\mathsf{case}(\mathsf{inr}(\mathsf{inr}^{i-2}(\mathsf{inl}\, t')), x.t_1[x], y.r_{k-2}[y]), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \to \\
\mathsf{case}\Big(r_{k-2}[\mathsf{inr}^{i-2}(\mathsf{inl}\, t')], x.s_1' x, y.q_{k-2}[y]\Big) &\quad \underset{\mathsf{prop}\ 2.7}{\to^{\star}} \\
\mathsf{case}\Big(r_{k-i}[\mathsf{inl}\, t'], x.s_1' x, y.q_{k-2}[y]\Big) &\quad \to \\
\mathsf{case}\Big(\mathsf{case}(\mathsf{inl}\, t', x.t_i[x], y.r_{k-i-1}[y]), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \to
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{case}\Big(t_i[t'], x.s_1' x, y.q_{k-2}[y]\Big) &\quad \to \\
\mathsf{case}\Big(\mathsf{inr}^{i-1}(\mathsf{inl}(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t')), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \equiv \\
\mathsf{case}\Big(\mathsf{inr}(\mathsf{inr}^{i-2}(\mathsf{inl}(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'))), x.s_1' x, y.q_{k-2}[y]\Big) &\quad \equiv \\
q_{k-2}[\mathsf{inr}^{i-2}(\mathsf{inl}(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'))] &\quad \underset{\mathsf{prop}\ 2.7}{\to^{\star}} \\
q_{k-i}[\mathsf{inl}(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t')] &\quad \underset{k-i>0}{\to} \\
\mathsf{case}\Big(\mathsf{inl}(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'), x.s_i' x, y.q_{k-i-1}[y]\Big) &\quad \to \\
s_i'\Big(m_i'(\lambda x.\mathsf{It}(\mathbb{M}, \mathbb{S}, x))t'\Big) \equiv \Big(s_i\Big(m_i(\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x))t\Big)\Big)'. &
\end{aligned}
$$

- ○ Subcase $i = k$.

$$
\begin{aligned}
\Big(\mathsf{It}_k(\vec{m},\vec{s},\mathsf{in}_{k,k}\,t)\Big)' \equiv \mathsf{It}(\mathbb{M},\mathbb{S},(\mathsf{in}_{k,k}\,t)') &\quad \equiv \\
\mathsf{It}(\mathbb{M},\mathbb{S},\mathsf{in}(\mathsf{inj}^k_k\,t')) &\quad \longrightarrow \\
\mathbb{S}\Big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))(\mathsf{inj}^k_k\,t')\Big) &\quad \longrightarrow \\
\mathsf{case}\Big(\mathbb{M}(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))(\mathsf{inj}^k_k\,t'), x.s'_1x, y.q_{k-2}[y]\Big) &\quad \longrightarrow \\
\mathsf{case}\Big(\mathsf{case}(\mathsf{inr}^{k-1}\,t', x.t_1[x], y.r_{k-2}[y]), x.s'_1x, y.q_{k-2}[y]\Big) &\quad \equiv \\
\mathsf{case}\Big(\mathsf{case}(\mathsf{inr}(\mathsf{inr}^{k-2}\,t'), x.t_1[x], y.r_{k-2}[y]), x.s'_1x, y.q_{k-2}[y]\Big) &\quad \longrightarrow \\
\mathsf{case}\Big(r_{k-2}[\mathsf{inr}^{k-2}\,t'], x.s'_1x, y.q_{k-2}[y]\Big) &\quad \underset{\text{prop 2.7}}{\longrightarrow^{\star}} \\
\mathsf{case}\Big(r_0[t'], x.s'_1x, y.q_{k-2}[y]\Big) \equiv \mathsf{case}\Big(t_k[t'], x.s'_1x, y.q_{k-2}[y]\Big) &\quad \longrightarrow \\
\mathsf{case}(\mathsf{inr}^{k-1}(m'_k(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))t'), x.s'_1x, y.q_{k-2}[y]) &\quad \equiv \\
\mathsf{case}(\mathsf{inr}(\mathsf{inr}^{k-2}(m'_k(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))t')), x.s'_1x, y.q_{k-2}[y]) &\quad \longrightarrow \\
q_{k-2}[\mathsf{inr}^{k-2}(m'_k(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))t')] &\quad \underset{\text{prop 2.7}}{\longrightarrow^{\star}} \\
q_0[m'_k(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))t'] &\quad \equiv \\
s'_k\big(m'_k(\lambda x.\mathsf{It}(\mathbb{M},\mathbb{S},x))t'\big) \equiv \Big(s_k\Big(m_k(\lambda x.\mathsf{It}_k(\vec{m},\vec{s},x))t\Big)\Big)'. &
\end{aligned}
$$

The remaining cases are solved analogously.

⊣

Proposition 2.12 MCICT *is strongly normalising.*

Proof. Immediate from propositions 2.5 and 2.11. ⊣

2.3.3 On η-rules

In this section we introduce some η-rules for the system MCICT.

The so-called η-rules of reduction are added to a type system to represent extensionality principles, which, from the categorical point of view means that some morphisms are unique. They identify certain functions (terms) which have the same behaviour, yet which are represented in different ways.

The first such rule is the one for λ-abstractions:

$$\lambda x.rx \mapsto_\eta r \qquad \text{with } x \notin FV(r) \quad (\eta_\to)$$

and guarantees extensionality for functions, i.e. allows to conclude that two functions f and g are equal if they coincide in all arguments, that is, if $fx = gx$ for all suitable arguments x.

Similarly we have η-rules for sums and products:

$$\begin{array}{rcll} \langle \pi_1 r, \pi_2 r\rangle & \mapsto_\eta & r & (\eta_\times) \\ \mathsf{case}(r, y.\,\mathsf{inl}\,y, z.\,\mathsf{inr}\,z) & \mapsto_\eta & r & (\eta_+) \end{array}$$

The rule for pairing is usually called surjective pairing.

To finish the system of η-rules for MCICT, we define η-rules for iteration and coinductive inversion.

The η-rule for iteration is:

$$\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1 \ldots \mathbb{C}^k_k, r) \mapsto_\eta r \quad (\eta_\mu)$$

where $\mathbb{C}^k_i := \lambda z.\,\mathsf{in}_{k,i}\,z$.

This rule is justified as follows: Doing iteration over the inductive type $\mu\alpha(\rho_1, \ldots, \rho_k)$ with step-functions $\mathbb{C}^k_i$ yields the diagram

$$\begin{tikzcd}
\rho_i[\alpha := \mu\alpha(\rho_1, \ldots, \rho_k)] \arrow[r, "\mathbb{C}^k_i"] \arrow[d, "m_i\left(\lambda z.\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, z)\right)"'] & \mu\alpha(\rho_1, \ldots, \rho_k) \arrow[d, "\lambda z.\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, z)"] \\
\rho_i[\alpha := \mu\alpha(\rho_1, \ldots, \rho_k)] \arrow[r, "\mathbb{C}^k_i"] & \mu\alpha(\rho_1, \ldots, \rho_k)
\end{tikzcd}$$

The iteration principle guarantees that this diagram commute, however, assuming the first functor law ($m_i(\mathsf{Id}) = \mathsf{Id}$) the identity function also makes the diagram commutative:

$$\begin{tikzcd}
\rho_i[\alpha := \mu\alpha(\rho_1,\dots,\rho_k)] \arrow[r, "\mathbb{C}_i^k"] \arrow[d, "m_i(\mathsf{Id})"'] & \mu\alpha(\rho_1,\dots,\rho_k) \arrow[d, "\mathsf{Id}"] \\
\rho_i[\alpha := \mu\alpha(\rho_1,\dots,\rho_k)] \arrow[r, "\mathbb{C}_i^k"] & \mu\alpha(\rho_1,\dots,\rho_k)
\end{tikzcd}$$

Therefore if we want the iterative morphism to be unique we have to settle $\lambda z.\mathsf{It}_k(\vec{m},\mathbb{C}_1^k\dots\mathbb{C}_k^k,z) = \lambda zz$, which implies that for every r we have,

$$\mathsf{It}_k(\vec{m},\mathbb{C}_1^k\dots\mathbb{C}_k^k,r) = r.$$

The η-rule for iteration follows from this last equality.

By dualizing we can get an η-rule for coiteration $\mathsf{CoIt}_k(\vec{m},\mathbb{D}_1^k\dots\mathbb{D}_k^k,r) \mapsto_\eta r$, where $\mathbb{D}_i^k := \lambda z.\,\mathsf{out}_{k,i}\,z$. However for our purposes, this rule is not neccesary, instead we need to consider the following η-rule for coinductive inversion:

$$\mathsf{out}_k^{-1}(\vec{m},\mathbb{D}_1^k r,\dots,\mathbb{D}_k^k r) \quad \mapsto_\eta \quad r \quad (\eta_\nu)$$

The justification of this rule is similar to the last one: In this case the composition

$$\big(\lambda\vec{x}.\,\mathsf{out}_k^{-1}(\vec{m},\vec{x})\big)\circ\langle\mathbb{D}_1^k,\dots,\mathbb{D}_k^k\rangle : \nu\alpha(\rho_1,\dots,\rho_k)\to\nu\alpha(\rho_1,\dots,\rho_k)$$

should be equal to the identity to guarantee that out_k^{-1} is an inverse of $\langle\mathbb{D}_1^k,\dots,\mathbb{D}_k^k\rangle$, so we settle $\big(\lambda\vec{x}.\,\mathsf{out}_k^{-1}(\vec{m},\vec{x})\big)\circ\langle\mathbb{D}_1^k,\dots,\mathbb{D}_k^k\rangle = \lambda zz$, which implies that for every r we have

$$\mathsf{out}_k^{-1}(\vec{m},\mathbb{D}_1^k r,\dots,\mathbb{D}_k^k r) = r$$

The η-rule follows by directing this equation. A more intuitive justification of this rule is the following: If we take a coinductive object r and destruct it getting all its pieces $\mathbb{D}_1^k r,\dots,\mathbb{D}_k^k r$ and then with these pieces we construct a coinductive object $\mathsf{out}_k^{-1}(\vec{m},\mathbb{D}_1^k r,\dots,\mathbb{D}_k^k r)$ this should be exactly the original object r.

The system MCICT with η-rules will be denoted MCICTη. We do not know anything about the strong normalization of the $\beta\eta$-reduction. Moreover the subject-reduction fails already for system F, i.e., with the first η-rule, as the following example shows:

$$\rhd\lambda x\lambda y.xy : (\forall\alpha.\sigma\to\tau)\to\forall\alpha\sigma\to\forall\alpha\tau$$

$$\lambda x\lambda y.xy \to_\eta \lambda xx$$

but

$$\not\rhd \lambda xx : (\forall\alpha.\sigma \to \tau) \to \forall\alpha\sigma \to \forall\alpha\tau$$

In general η-rules are evil for Curry-style systems as they destroy the type-preservation property.
However the use of η-rules is still of interest to us. We will see in the following section that in $\mathsf{MCICT}\eta$ we can prove the first functor law for the so-called canonical monotonicity witnesses, fact that will be useful in chapter 4 to formulate a nice soundness theorem for realizability.

It is not clear if η-rules are rules of computation. Moreover, to our knowledge, it is a piece of folklore to say that the β-rules are computationally sufficient to ensure the same results from the application of η-equivalent terms (functions). For the cases of the rules $(\eta_\to), (\eta_\times), (\eta_+)$ this is more or less clear as an η-redex can be avoided with β-reduction. For instance for surjective pairing the η-reduction $\pi_2\langle\pi_1 r, \pi_2 r\rangle \to_\eta \pi_2 r$ is clearly also a β-reduction $\pi_2\langle\pi_1 r, \pi_2 r\rangle \to_\beta \pi_2 r$. This folkloric view is not so clear for the new rules $(\eta_\mu), (\eta_\nu)$ and needs further study. In [Ho92] (theorem 3.3.5, page 35) B. Howard claims to justify the redundancy of a system of η-rules, including some rules for (co)inductive types, with respect to essentially the same β-rules of MCICT. However we were not able to understand the proof-sketch he gives.

2.3.4 Canonical Monotonicity Witnesses

In this section we present a canonical selection for monotonicity witnesses which essentially corresponds to the usual definitions for the positive cases, we do not restrict ourselves to strict positivity and define also antimonotonicity. Moreover we define witnesses for interleaving types.

Definition 2.10 (Antimonotonicity) *Given a type ρ and a type variable α, we define the type $\rho\,\mathsf{mon}^-\,\alpha$ as:*

$$\rho\,\mathsf{mon}^-\,\alpha := \forall\alpha.\forall\beta.(\alpha \to \beta) \to \rho[\alpha := \beta] \to \rho$$

If a term m inhabits the type $\rho\,\mathsf{mon}^-\,\alpha$ in a given context, then the functor $\langle\lambda\alpha\rho, m\rangle$ will be antimonotone (contravariant) in the same context.

Definition 2.11 (Generic (Anti)monotonicity Witnesses) *We define the following* MCICT*-terms:*

- $\mathbb{M}_{\mathsf{id}} := \lambda xx$
- $\mathbb{M}_{\mathsf{triv}} := \lambda f\lambda xx$

- $\mathbb{M}_{\to} := \lambda m_1 \lambda m_2 \lambda f \lambda x \lambda y. m_2 f(x(m_1 f y))$
- $\mathbb{M}_{\forall} := \lambda m \lambda f \lambda x. m f x$
- $\mathbb{M}_{\times} := \lambda m_1 \lambda m_2 \lambda f \lambda x \langle m_1 f(\pi_1 x), m_2 f(\pi_2 x)\rangle$
- $\mathbb{M}_{+} := \lambda m_1 \lambda m_2 \lambda f \lambda x. \mathsf{case}(x, y.\, \mathsf{inl}\, m_1 f y, z.\, \mathsf{inr}\, m_2 f z)$
- $\mathbb{M}^k_{\mu} := \lambda \vec{m} \lambda \vec{n} \lambda f \lambda x. \mathsf{It}_k(\vec{m}, \vec{s}, x)$, *where* $s_i := \lambda z.\, \mathsf{in}_{k,i}(n_i f z)$.
- $\mathbb{M}^k_{\nu} := \lambda \vec{m} \lambda \vec{n} \lambda f \lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, \vec{s})$ *where* $s_i := n_i f(\mathsf{out}_{k,i}\, x)$.

Proposition 2.13 *We have the following derivations:*

- $\rhd \mathbb{M}_{\mathsf{id}} : \alpha\, \mathsf{mon}\, \alpha$
- *If* $\alpha \notin FV(\rho)$ *then* $\rhd \mathbb{M}_{\mathsf{triv}} : \rho\, \mathsf{mon}\, \alpha$ *and* $\rhd \mathbb{M}_{\mathsf{triv}} : \rho\, \mathsf{mon}^-\, \alpha$
- $\rhd \mathbb{M}_{\to} : \sigma\, \mathsf{mon}^-\, \alpha \to \tau\, \mathsf{mon}\, \alpha \to (\sigma \to \tau)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}_{\to} : \sigma\, \mathsf{mon}\, \alpha \to \tau\, \mathsf{mon}^-\, \alpha \to (\sigma \to \tau)\, \mathsf{mon}^-\, \alpha$
- $\rhd \mathbb{M}_{\forall} : (\forall \gamma. \sigma\, \mathsf{mon}\, \alpha) \to (\forall \gamma. \sigma)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}_{\forall} : (\forall \gamma. \sigma\, \mathsf{mon}^-\, \alpha) \to (\forall \gamma. \sigma)\, \mathsf{mon}^-\, \alpha$
- $\rhd \mathbb{M}_{\times} : \sigma\, \mathsf{mon}\, \alpha \to \tau\, \mathsf{mon}\, \alpha \to (\sigma \times \tau)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}_{\times} : \sigma\, \mathsf{mon}^-\, \alpha \to \tau\, \mathsf{mon}^-\, \alpha \to (\sigma \times \tau)\, \mathsf{mon}^-\, \alpha.$
- $\rhd \mathbb{M}_{+} : \sigma\, \mathsf{mon}\, \alpha \to \tau\, \mathsf{mon}\, \alpha \to (\sigma + \tau)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}_{+} : \sigma\, \mathsf{mon}^-\, \alpha \to \tau\, \mathsf{mon}^-\, \alpha \to (\sigma + \tau)\, \mathsf{mon}^-\, \alpha.$
- $\rhd \mathbb{M}_{\mu} : (\forall \alpha. \tau_i\, \mathsf{mon}\, \gamma) \to (\forall \gamma. \tau_i\, \mathsf{mon}\, \alpha) \to \mu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}_{\mu} : (\forall \alpha. \tau_i\, \mathsf{mon}\, \gamma) \to (\forall \gamma. \tau_i\, \mathsf{mon}^-\, \alpha) \to \mu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}^-\, \alpha$
- $\rhd \mathbb{M}^k_{\nu} : (\forall \alpha. \tau_i\, \mathsf{mon}\, \gamma) \to (\forall \gamma. \tau_i\, \mathsf{mon}\, \alpha) \to \nu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}\, \alpha$
  $\rhd \mathbb{M}^k_{\nu} : (\forall \alpha. \tau_i\, \mathsf{mon}\, \gamma) \to (\forall \gamma. \tau_i\, \mathsf{mon}^-\, \alpha) \to \nu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}^-\, \alpha$

Proof. Straightforward ⊣

Corollary 2.10 (Derived Typing Rules for (Anti)monotonicity) *The following rules are derivable:*

- $\Sigma \rhd \mathbb{M}_{\mathsf{id}} : \alpha\, \mathsf{mon}\, \alpha$
- *If* $\alpha \notin FV(\rho)$ *then* $\Sigma \rhd \mathbb{M}_{\mathsf{triv}} : \rho\, \mathsf{mon}\, \alpha$ *and* $\Sigma \rhd \mathbb{M}_{\mathsf{triv}} : \rho\, \mathsf{mon}^-\, \alpha$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}^-\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}\, \alpha$ *then*
  $$\Sigma \rhd \mathbb{M}_{\to} m_1 m_2 : (\sigma \to \tau)\, \mathsf{mon}\, \alpha$$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}^-\, \alpha$ *then*
  $$\Sigma \rhd \mathbb{M}_{\to} m_1 m_2 : (\sigma \to \tau)\, \mathsf{mon}^-\, \alpha$$

- *If* $\Sigma \rhd t : \forall\gamma.\sigma\, \mathsf{mon}\, \alpha$ *then* $\Sigma \rhd \mathbb{M}_{\forall} t : (\forall\gamma.\sigma)\, \mathsf{mon}\, \alpha$
- *If* $\Sigma \rhd t : \forall\gamma.\sigma\, \mathsf{mon}^-\, \alpha$ *then* $\Sigma \rhd \mathbb{M}_{\forall} t : (\forall\gamma.\sigma)\, \mathsf{mon}^-\, \alpha$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}\, \alpha$ *then*
$$\Sigma \rhd \mathbb{M}_{\times} m_1 m_2 : (\sigma \times \tau)\, \mathsf{mon}\, \alpha$$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}^-\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}^-\, \alpha$ *then*
$$\Sigma \rhd \mathbb{M}_{\times} m_1 m_2 : (\sigma \times \tau)\, \mathsf{mon}^-\, \alpha$$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}\, \alpha$ *then*
$$\Sigma \rhd \mathbb{M}_{+} m_1 m_2 : (\sigma + \tau)\, \mathsf{mon}\, \alpha$$
- *If* $\Sigma \rhd m_1 : \sigma\, \mathsf{mon}^-\, \alpha$ *and* $\Sigma \rhd m_2 : \tau\, \mathsf{mon}^-\, \alpha$ *then*
$$\Sigma \rhd \mathbb{M}_{+} m_1 m_2 : (\sigma + \tau)\, \mathsf{mon}^-\, \alpha$$
- *If* $\Sigma \rhd m_i : (\forall\alpha.\tau_i\, \mathsf{mon}\, \gamma)$ *and* $\Sigma \rhd n_i : (\forall\gamma.\tau_i\, \mathsf{mon}\, \alpha)$ *then*
$$\Sigma \rhd \mathbb{M}^k_{\mu} \vec{m}\vec{n} : \mu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}\, \alpha$$
- *If* $\Sigma \rhd m_i : (\forall\alpha.\tau_i\, \mathsf{mon}\, \gamma)$ *and* $\Sigma \rhd n_i : (\forall\gamma.\tau_i\, \mathsf{mon}^-\, \alpha)$ *then*
$$\Sigma \rhd \mathbb{M}^k_{\mu} \vec{m}\vec{n} : \mu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}^-\, \alpha$$
- *If* $\Sigma \rhd m_i : (\forall\alpha.\tau_i\, \mathsf{mon}\, \gamma)$ *and* $\Sigma \rhd n_i : (\forall\gamma.\tau_i\, \mathsf{mon}\, \alpha)$ *then*
$$\Sigma \rhd \mathbb{M}^k_{\nu} \vec{m}\vec{n} : \nu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}\, \alpha$$
- *If* $\Sigma \rhd m_i : (\forall\alpha.\tau_i\, \mathsf{mon}\, \gamma)$ *and* $\Sigma \rhd n_i : (\forall\gamma.\tau_i\, \mathsf{mon}^-\, \alpha)$ *then*
$$\Sigma \rhd \mathbb{M}^k_{\nu} \vec{m}\vec{n} : \nu\gamma(\tau_1, \ldots, \tau_k)\, \mathsf{mon}^-\, \alpha$$

Proof. Trivial ⊣

Definition 2.12 *We will write* $\Sigma \rhd^{\mathsf{can}} m : \forall\vec{\gamma}.\rho\, \mathsf{mon}\, \alpha$ *if* m *was obtained by one of the rules of the previous corollary (possibly using also the rules for universal quantifiers). We say that a monotonicity witness* m *is canonical if* $\rhd^{\mathsf{can}} m : \rho\, \mathsf{mon}\, \alpha$*.*

The importance of the η-rules is made explicit in the following proposition which provides the first functor law for canonical witnesses by means of $\beta\eta$-reductions.

Proposition 2.14 (First Functor Law) *If* $\Sigma \rhd^{\mathsf{can}} m : \forall\vec{\gamma}.\rho\, \mathsf{mon}\, \alpha$ *and* $\Sigma \rhd^{\mathsf{can}} m^- : \forall\vec{\gamma}.\sigma\, \mathsf{mon}^-\, \alpha$ *then* m *and* m^- *satisfy the first functor law in* MCICTη*, that is:*
$$m(\lambda z.z) \to^{\star}_{\beta\eta} \lambda y.y \quad m^-(\lambda z.z) \to^{\star}_{\beta\eta} \lambda y.y$$

Proof. Induction on $\rhd^{\mathsf{can}}$ (see page 89 for essentially the needed proof). ⊣

2.3.5 (Co)recursive Programming in MCICT

In this section we give some examples of how to program with (co)induction principles in the type system MCICT.

Given an inductive type $\mu\alpha(\rho_1,\dots,\rho_k)$ the goal is to program functions

$$g : \mu\alpha(\rho_1,\dots,\rho_k) \to \sigma$$

To do this we have two tools available: iteration and primitive recursion. In this kind of definitions the consstructors of the type play an important role: to define a function g by iteration or recursion, one defines the value of g on all constructors.

Recall that the constructors of $\mu\alpha(\rho_1,\dots,\rho_k)$,

$$\mathbb{C}_i^k : \rho_i[\alpha := \mu\alpha(\rho_1,\dots,\rho_k)] \to \mu\alpha(\rho_1,\dots,\rho_k)$$

are defined as $\mathbb{C}_i^k := \lambda z.\, \mathsf{in}_{k,i}\, z$

The easiest way to program functions is by iteration, this scheme provides a mean to define functions $g : \mu\alpha(\rho_1,\dots,\rho_k) \to \sigma$ which satisfy the following recurrence equations:

$$\begin{array}{rcl} g(\mathbb{C}_1^k x) & = & s_1(m_1 g x) \\ & \vdots & \\ g(\mathbb{C}_k^k x) & = & s_k(m_k g x) \end{array}$$

where $s_i : \rho_i[\alpha := \sigma] \to \sigma$ and $m_i : \rho_i \,\mathsf{mon}\, \alpha, 1 \leq i \leq k$ are the fixed monotonicity witnesses used to eliminate the type $\mu\alpha(\rho_1,\dots,\rho_k)$.

If these conditions hold, then the categorical machinery says that we can define $g := \lambda z.\mathsf{It}_k(\vec{m},\vec{s},z)$ and we will obtain the desired reduction behaviour:

$$g(\mathbb{C}_i^k x) \to_\beta^+ s_i(m_i g x)$$

Analogously primitive recursion provides a mean to program functions $g : \mu\alpha(\rho_1,\dots,\rho_k) \to \sigma$ which satisfy the following recurrence equations:

$$\begin{array}{rcl} g(\mathbb{C}_1^k x) & = & s_1(m_1\langle \mathsf{Id}, g\rangle x) \\ & \vdots & \\ g(\mathbb{C}_k^k x) & = & s_k((m_k\langle \mathsf{Id}, g\rangle x) \end{array}$$

with $s_i : \rho_i[\alpha := \mu\alpha(\rho_1,\dots,\rho_k) \times \sigma] \to \sigma$. In this case g can be defined as $g := \lambda z.\mathsf{Rec}_k(\vec{m},\vec{s},z)$, and we get:

$$g(\mathbb{C}_i^k x) \to_\beta^+ s_i(m_i\langle \mathsf{Id}, g\rangle x).$$

In a dual way given a coinductive type $\nu\alpha(\rho_1,\dots,\rho_k)$ the goal is to program functions

$$g : \sigma \to \nu\alpha(\rho_1,\dots,\rho_k)$$

To do this we have two tools available: coiteration and corecursion. In this kind of definitions the destructors of the type play an important role: to define a function g by coiteration or corecursion, one defines the values of all destructors on each outcome gx.

Recall that the destructors of $\nu\alpha(\rho_1, \ldots, \rho_k)$,

$$\mathbb{D}_i^k : \nu\alpha(\rho_1, \ldots, \rho_k) \to \rho_i[\alpha := \nu\alpha(\rho_1, \ldots, \rho_k)]$$

are defined as $\mathbb{D}_i^k := \lambda z.\, \mathsf{out}_{k,i}\, z$

The first way to program functions is by coiteration, this scheme provides a mean to define functions $g : \sigma \to \nu\alpha(\rho_1, \ldots, \rho_k)$ which satisfy the following recurrence equations:

$$\begin{array}{rcl} \mathbb{D}_1^k(gx) & = & (m_1 g)(s_1 x) \\ & \vdots & \\ \mathbb{D}_k^k(gx) & = & (m_k g)(s_k x) \end{array}$$

where $s_i : \sigma \to \rho_i[\alpha := \sigma]$ and $m_i : \rho_i \,\mathsf{mon}\, \alpha, 1 \leq i \leq k$ are the fixed monotonicity witnesses used to introduce the type $\nu\alpha(\rho_1, \ldots, \rho_k)$.

If these conditions hold, then the categorical machinery says that we can define $g := \lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z)$ and we will obtain the desired reduction behaviour:

$$\mathbb{D}_i^k(gx) \to_\beta^+ (m_i g)(s_i x)$$

Analogously corecursion provides a mean to program functions $g : \sigma \to \nu\alpha(\rho_1, \ldots, \rho_k)$ which satisfy the following recurrence equations:

$$\begin{array}{rcl} \mathbb{D}_1^k(gx) & = & (m_1[\mathsf{Id}, g])(s_1 x) \\ & \vdots & \\ \mathbb{D}_k^k(gx) & = & (m_k[\mathsf{Id}, g])(s_k x) \end{array}$$

with $s_i : \sigma \to \rho_i[\alpha := \nu\alpha(\rho_1, \ldots, \rho_k) + \sigma]$. In this case g can be defined as $g := \lambda z.\mathsf{CoRec}_k(\vec{m}, \vec{s}, z)$, and we get:

$$\mathbb{D}_i^k(gx) \to_\beta^+ (m_i[\mathsf{Id}, g])(s_i x)$$

Let us see some examples

Examples with Inductive Types

Consider the inductive types $\mathsf{bool}, \mathsf{nat}, \mathsf{list}(\rho)$ defined together with its constructors and monotonicity witnesses as follows:

$\mathsf{bool} := \mu\alpha(1, 1)$	true	false	$\mathbb{M}_{\mathsf{triv}}$	$\mathbb{M}_{\mathsf{triv}}$
$\mathsf{nat} := \mu\alpha(1, \alpha)$	0	s	$\mathbb{M}_{\mathsf{triv}}$	$\mathbb{M}_{\mathsf{id}}$
$\mathsf{list}(\rho) := \mu\alpha(1, \rho \times \alpha)$	nil	cons	$\mathbb{M}_{\mathsf{triv}}$	$\mathbb{M}_{\rho\times\alpha} := \mathbb{M}_{\times}^2 \mathbb{M}_{\mathsf{triv}} \mathbb{M}_{\mathsf{id}}$

where the witnesses are the canonical ones obtained with the rules in section 2.3.4.

Negation

The negation function $\mathsf{not} : \mathsf{bool} \to \mathsf{bool}$ is defined as

$$\mathsf{not}(\mathsf{true}) = \mathsf{false} \quad \mathsf{not}(\mathsf{false}) = \mathsf{true}$$

This is an instance of iteration and is programmed as:

$$\mathsf{not} := \lambda y.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, \mathbb{C}_2^2, \mathbb{C}_1^2, y)$$

Boolean Conditional

Given a type σ we want to define the conditional function

$$\mathsf{if\_then\_else\_} : \mathsf{bool} \to \sigma \to \sigma \to \sigma$$

with the following behaviour, for $r, s : \sigma$:

$$\begin{array}{c}\mathsf{if\ true\ then}\ r\ \mathsf{else}\ s = r \\ \mathsf{if\ false\ then}\ r\ \mathsf{else}\ s = s\end{array}$$

This is easily defined by iteration as:

$$\mathsf{if\ \_\ then\ \_\ else\ \_} := \lambda z \lambda x \lambda y.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, \lambda u.x, \lambda v.y, z)$$

where $u \neq x, v \neq y$.

The Predecessor function

This is the inductive destructor for naturals $\mathsf{pred} : \mathsf{nat} \to \mathsf{nat}$ defined as:

$$\mathsf{pred}(0) = 0 \quad \mathsf{pred}(sn) = n$$

The usual way to program this function is with recursion as

$$\mathsf{pred} := \lambda x.\mathsf{Rec}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda z.0, \lambda z.\pi_1 z, x)$$

Another way to program the predecessor is by using inductive inversion getting:

$$\mathsf{pred} := \lambda z.\mathsf{case}(\mathsf{in}_2^{-1}(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, z), x.0, y.y)$$

Zero-check

The function $\mathsf{zero?} : \mathsf{nat} \to \mathsf{bool}$ is defined as

$$\mathsf{zero?}(0) = \mathsf{true} \quad \mathsf{zero?}(sn) = \mathsf{false}$$

$\mathsf{zero?}$ is programmed via inductive inversion as:

$$\mathsf{zero?} := \lambda x.\mathsf{case}(\mathsf{in}_2^{-1}(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, x), y.\mathsf{true}, z.\mathsf{false})$$

Equality of Natural Numbers

We would like to define a function $\mathsf{eq?} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$ such that $\mathsf{eq?}\langle n, m\rangle = \mathsf{true}$ if and only if $n = m$. We do not have a direct way to program this function, we only know how to define functions from inductive types and to coinductive types but $\mathsf{eq?}$ is a function from a product to an inductive type. The solution is to program the curried version $\mathsf{eq?} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{bool}$, which is a function from an inductive type. This is defined as:

$$\begin{aligned}\mathsf{eq?}(0) &= \mathsf{zero?}\\ \mathsf{eq?}(sn) &= \lambda m.\mathsf{if}\ \mathsf{zero?}m\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{eq?}(n)(\mathsf{pred}\, m)\end{aligned}$$

This is an instance of iteration with step functions

$$\begin{aligned}&s_1 := \lambda z.\mathsf{zero?}\\ &s_2 := \lambda g\lambda m.\mathsf{if}\ \mathsf{zero?}m\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ g(\mathsf{pred}\, m)\end{aligned}$$

With this definition we get $\mathsf{eq?}(n) : \mathsf{nat} \to \mathsf{bool}$ such that

$$\begin{gathered}\mathsf{eq?}(0)(0) = \mathsf{true}\ \ \mathsf{eq?}(0)(sn) = \mathsf{false}\\ \mathsf{eq?}(sm)(0) = \mathsf{false}\ \ \mathsf{eq?}(sm)(sn) = \mathsf{eq?}(m)(n)\end{gathered}$$

Testing for $\leq$

The function $\mathsf{leq?} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{bool}$ such that

$$\mathsf{leq?}mn = \mathsf{true} \text{ if and only if } m \leq n$$

is defined as:

$$\begin{gathered}\mathsf{leq?}(0)(0) = \mathsf{true}\ \ \mathsf{leq?}(0)(sn) = \mathsf{true}\\ \mathsf{leq?}(sm)(0) = \mathsf{false}\ \ \mathsf{leq?}(sm)(sn) = \mathsf{leq?}(m)(n)\end{gathered}$$

This function is defined by iteration analogous to $\mathsf{eq?}$ as

$$\mathsf{leq?} := \lambda z.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, z)$$

where the steps functions are

$$\begin{aligned}&s_1 := \lambda z.\mathsf{true}\\ &s_2 := \lambda g\lambda m.\mathsf{if}\ \mathsf{zero?}m\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ g(\mathsf{pred}\, m)\end{aligned}$$

Minimum Function

The minimum function $\mathrm{min} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ can be directly defined with help from $\mathsf{leq?}$ as:

$$\mathrm{min} := \lambda y\lambda z.\mathsf{if}\ \mathsf{leq?}\, y\, z\ \mathsf{then}\ y\ \mathsf{else}\ z$$

We can also define a non-curried version $\mathrm{min} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$ as

$$\mathrm{min} := \lambda z.\mathsf{if}\ \mathsf{leq?}(\pi_1 z)(\pi_2 z)\ \mathsf{then}\ \pi_1 z\ \mathsf{else}\ \pi_2 z$$

Append of Lists

The function append is usually defined as $\mathsf{append} : \mathsf{list}(\rho) \times \mathsf{list}(\rho) \to \mathsf{list}(\rho)$ with:

$$\mathsf{append}\langle \mathsf{nil}, l\rangle = l \qquad \mathsf{append}\langle \mathsf{cons}\langle a, l_1\rangle, l_2\rangle = \mathsf{cons}\langle a, \mathsf{append}\langle l_1, l_2\rangle\rangle$$

This function cannot be defined directly as neither its domain is an inductive type nor its codomain is a coinductive type. The solution is to program the curried version $\mathsf{append} : \mathsf{list}(\rho) \to \mathsf{list}(\rho) \to \mathsf{list}(\rho)$ defined as:

$$\mathsf{append}\,\mathsf{nil}\,l = l \qquad \mathsf{append}\,\mathsf{cons}\langle a, l_1\rangle\, l_2 = \mathsf{cons}\langle a, \mathsf{append}\langle l_1\, l_2\rangle\rangle$$

Now we have a function with an inductive type as domain, which can be iteratively defined with the step functions

$$s_1 : 1 \to \mathsf{list}(\rho) \to \mathsf{list}(\rho) \quad s_1 := \lambda z.z$$

$$s_2 : \rho \times (\mathsf{list}(\rho) \to \mathsf{list}(\rho)) \to \mathsf{list}(\rho) \to \mathsf{list}(\rho) \quad s_2 := \lambda x \lambda y.\, \mathsf{cons}\langle \pi_1 x, (\pi_2 x) y\rangle$$

Examples with Streams

Given a type ρ the type of streams (infinite lists) of elements of ρ is defined as:

$$\mathsf{stream}(\rho) := \nu\alpha(\rho, \alpha)$$

where $\alpha \notin FV(\rho)$. The monotonicity witnesses needed to introduce this type are canonical, we have: $\rhd_{\mathsf{C}} \mathbb{M}_{\mathsf{triv}} : \rho\,\mathsf{mon}\,\alpha$, $\rhd_{\mathsf{C}} \mathbb{M}_{\mathsf{id}} : \alpha\,\mathsf{mon}\,\alpha$.

The associated destructors are $\mathsf{head}, \mathsf{tail}$ defined as $\mathsf{head} := \mathbb{D}^2_1, \mathsf{tail} := \mathbb{D}^2_2$ such that

$$\rhd\, \mathsf{head} : \mathsf{stream}(\rho) \to \rho$$
$$\rhd\, \mathsf{tail} : \mathsf{stream}(\rho) \to \mathsf{stream}(\rho)$$

To define a function $g : \sigma \to \mathsf{stream}(\rho)$, the equations for coiteration are simplified to:

$$\begin{array}{rcl} \mathsf{head}(gx) & = & s_1 x \\ \mathsf{tail}(gx) & = & g(s_2 x) \end{array}$$

with $s_1 : \sigma \to \rho,\ s_2 : \sigma \to \sigma$,

whereas the equations for corecursion are:

$$\begin{array}{rcl} \mathsf{head}(gx) & = & s_1 x \\ \mathsf{tail}(gx) & = & [\mathsf{Id}, g](s_2 x) \end{array}$$

with $s_1 : \sigma \to \rho,\ s_2 : \sigma \to \mathsf{stream}(\rho) + \sigma$.
Let us program some functions involving streams.

A Stream of Constants

Given a constant $c : \rho$ we want to define the stream $\mathsf{cst}(c) := \langle c, c, c, c, \ldots \rangle$. That is we want to define a function $\mathsf{cst} : \rho \to \mathsf{stream}(\rho)$ such that:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{cst}\, x) & = & x \\ \mathsf{tail}(\mathsf{cst}\, x) & = & cst\, x \end{array}$$

The step functions are therefore $s_1, s_2 : \rho \to \rho$ with $s_1, s_2 := \lambda x.x$ and we define $\mathsf{cst} := \lambda z.\mathsf{CoIt}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, z)$.

The Stream of Naturals from a given one

The function $\mathsf{from} : \mathsf{nat} \to \mathsf{stream}(\mathsf{nat})$ with $\mathsf{from}\, n = \langle n, n+1, n+2, \ldots \rangle$ is destructed as follows:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{from}\, n) & = & n \\ \mathsf{tail}(\mathsf{from}\, n) & = & \mathsf{from}\, sn \end{array}$$

From these equations we identify the step functions $s_1 : \mathsf{nat} \to \mathsf{nat},\ s_2 : \mathsf{nat} \to \mathsf{nat}$ with $s_1 := \lambda zz,\ s_2 := s$ (the succesor function on nat). from can then be defined coiteratively.
The stream of natural numbers is

$$\omega := \mathsf{from}\ 0 \equiv \mathsf{CoIt}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda zz, s, 0).$$

The Append Function

The function $\mathsf{app} : \mathsf{stream}(\rho) \times \mathsf{stream}(\rho) \to \mathsf{stream}(\rho)$ is destructed as follows:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{app}\ x) & = & \mathsf{head}(\pi_1 x) \\ \mathsf{tail}(\mathsf{app}\ x) & = & \mathsf{app}\, \langle \mathsf{tail}(\pi_1 x), \pi_2 x \rangle \end{array}$$

Therefore its coiterative definition is $\mathsf{app} := \lambda z.\mathsf{CoIt}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, z)$, where $s_1 := \lambda z.\, \mathsf{head}\ \pi_1 x,\ s_2 := \lambda z.\langle \mathsf{tail}\ \pi_1 z, \pi_2 z \rangle$.

The Map Head Function

Given a function $h : \rho \to \rho$, the map head function $\mathsf{maphd}_h : \mathsf{stream}(\rho) \to \mathsf{stream}(\rho)$ maps a stream $\langle a_1, a_2, a_3, \ldots \rangle$ into the stream $\langle h(a_1), a_2, a_3, \ldots \rangle$. This function is destructed as follows:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{maphd}_h\, x) & = & h(\mathsf{head}\ x) \\ \mathsf{tail}(\mathsf{maphd}_h\, x) & = & \mathsf{tail}\ x \end{array}$$

This function can be defined by corecursion as

$$\mathsf{maphd}_h := \lambda z.\mathsf{CoRec}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, z)$$

taking $s_1 := h \circ \mathsf{head} : \mathsf{stream}(\rho) \to \rho$ and $s_2 := \mathsf{inl} \circ \mathsf{tail} : \mathsf{stream}(\rho) \to \mathsf{stream}(\rho) + \mathsf{stream}(\rho)$. We have

$$\mathsf{head}(\mathsf{maphd}_h x) \to_\beta (h \circ \mathsf{head})x \to_\beta h(\mathsf{head}\, x)$$

$$\mathsf{tail}(\mathsf{maphd}_h x) \to_\beta [\mathsf{Id}, \mathsf{maphd}_h]((\mathsf{inl} \circ \mathsf{tail})x) \to_\beta [\mathsf{Id}, \mathsf{maphd}_h]((\mathsf{inl}(\mathsf{tail}\ x))$$

$$\to_\beta \mathsf{Id}(\mathsf{tail}\ x) \to_\beta \mathsf{tail}\ x.$$

The cons Function

The cons function $\mathsf{cons} : \rho \times \mathsf{stream}(\rho) \to \mathsf{stream}(\rho)$ is destructed as:

$$\begin{aligned} \mathsf{head}(\mathsf{cons}\ x) &= \pi_1 x \\ \mathsf{tail}(\mathsf{cons}\ x) &= \pi_2 x \end{aligned}$$

Then cons can be corecursively defined from $s_1 := \lambda z.\pi_1 z, s_2 := \lambda z.\,\mathsf{inl}\,\pi_2 z$ as $\mathsf{cons} := \lambda x.\mathsf{CoRec}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, x)$.

However a more efficient cons function can be programmed using the inversion rule as follows: If $z : \rho \times \mathsf{stream}(\rho)$ then obviously $\pi_1 z : \rho$ and $\pi_2 z : \mathsf{stream}(\rho)$, therefore we can define $\mathsf{cons} := \lambda z.\,\mathsf{out}_2^{-1}(\pi_1 z)(\pi_2 z) : \rho \times \mathsf{stream}(\rho) \to \mathsf{stream}(\rho)$.

Sorted Insertion

Given a natural number n and a stream of naturals s we want to insert the number n exactly before the first element of s greater or equal than n. We define a function $\mathsf{si} : \mathsf{stream}(\mathsf{nat}) \times \mathsf{nat} \to \mathsf{stream}(\mathsf{nat})$ such that:

$$\mathsf{head}(\mathsf{si}\langle s, n\rangle) = \min\langle n, \mathsf{head}\, s\rangle$$

$$\mathsf{tail}(\mathsf{si}\langle s, n\rangle) = \begin{cases} s & \text{if } n \leq \mathsf{head}\ s \\ \mathsf{si}\langle \mathsf{tail}\, s, n\rangle & \text{if } n > \mathsf{head}\, s \end{cases}$$

We assume some given programs for the functions $\mathsf{leq?} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}, \min : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$. The condition to define the tail of $\mathsf{si}\langle s, n\rangle$ is controlled by the function $h := \lambda w.\mathsf{leq?}\langle \pi_2 w, \mathsf{head}\,\pi_1 w\rangle$. Now set

$$s_1 := \lambda z.\min\langle \pi_2 z, \mathsf{head}\,\pi_1 z\rangle$$

$$s_2 := \lambda z.[\lambda u.\,\mathsf{inl}\,\pi_1 z, \lambda v.\,\mathsf{inr}\langle \mathsf{tail}\,\pi_1 z, \pi_2 z\rangle](h\, z)$$

Finally si is defined as $\lambda z.\mathsf{CoRec}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, s_1, s_2, z)$.

2.4 The System MCICT_M

We present in this section another extension of system F, this time with (co)induction principles in Mendler-style. The section is mainly informative, we only give the definition of the system and sketch its normalization proof which is of theoretical interest, for it uses a non-homomorphical embedding on (co)inductive types by means of syntactical Kan extensions.

2.4.1 Definition of the System

We define a system, denoted MCICT_M, of monotone and clausular (co)inductive types with Mendler-style (co)iteration and (co)recursion as explained in section 2.1.2 extending F with clausular inductive types keeping the rules $(\mu I), (\nu E)$ and (νI^i) and substituting the rules for (co)iteration and (co)recursion with the following ones:

$$\frac{\Sigma \rhd s_i : \forall\alpha.\big(\alpha \to \sigma\big) \to \big(\rho_i \to \sigma\big),\;\; 1 \le i \le k}{\Sigma \rhd \mathsf{MIt}_k\vec{s} : \mu\alpha(\rho_1,\ldots,\rho_k) \to \sigma}\;(M\mu E)$$

$$\frac{\begin{array}{ll}\Sigma \rhd s_i : & \forall\alpha.\big(\alpha \to \mu\alpha(\rho_1,\ldots,\rho_k)\big) \to \\ & \big(\alpha \to \sigma\big) \to \big(\rho_i \to \sigma\big),\;\; 1 \le i \le k\end{array}}{\Sigma \rhd \mathsf{MRec}_k\vec{s} : \mu\alpha(\rho_1,\ldots,\rho_k) \to \sigma}\;(M\mu E^+)$$

$$\frac{\Sigma \rhd s_i : \forall\alpha.\big(\sigma \to \alpha\big) \to \big(\sigma \to \rho_i\big),\;\; 1 \le i \le k}{\Sigma \rhd \mathsf{MCoIt}_k\vec{s} : \sigma \to \nu\alpha(\rho_1,\ldots,\rho_k)}\;(M\nu I)$$

$$\frac{\begin{array}{ll}\Sigma \rhd s_i : & \forall\alpha.\big(\nu\alpha(\rho_1,\ldots,\rho_k) \to \alpha\big) \to \\ & \big(\sigma \to \alpha\big) \to \big(\sigma \to \rho_i\big),\;\; 1 \le i \le k\end{array}}{\Sigma \rhd \mathsf{MCoRec}_k\vec{s} : \sigma \to \nu\alpha(\rho_1,\ldots,\rho_k)}\;(M\nu I^+)$$

All rules with the proviso $\alpha \notin FV(\Sigma,\sigma)$.

These rules express Mendler-style (co)iteration and (co)recursion respectively.

The reduction behaviour is given by:

$$\begin{array}{rcl}\mathsf{MIt}_k\vec{s}(\mathsf{in}_{k,i}\,r) & \mapsto_\beta & s_i\big(\mathsf{MIt}_k\vec{s}\big)r \\ \mathsf{MRec}_k\vec{s}(\mathsf{in}_{k,i}\,r) & \mapsto_\beta & s_i(\lambda yy)\big(\mathsf{MRec}_k\vec{s}\big)r \\ \mathsf{out}_{k,i}\,\big(\mathsf{MCoIt}_k\vec{s}\,r\big) & \mapsto_\beta & s_i\big(\mathsf{MCoIt}_k\vec{s}\big)r \\ \mathsf{out}_{k,i}\,\big(\mathsf{MCoRec}_k\vec{s}\,r\big) & \mapsto_\beta & s_i(\lambda yy)\big(\mathsf{MCoRec}_k\vec{s}\big)r\end{array}$$

Observe that in MCICT_M we do not have neither sums nor products as they were only needed to define conventional (co) recursion. On the other hand we have neither inductive inversion as this rule cannot be faithfully embedded into MCICT.

2.4.2 Strong Normalization of $\mathsf{MCICT}_{\boldsymbol{M}}$

We give an embedding from $\mathsf{MCICT}_{\boldsymbol{M}}$ to $\mathsf{MCICT}^{\exists}$ which uses syntactical Kan extensions along the identity, for a discussion about them see [AMU04]. Here we only state the cases involving (co)inductive types/terms.

Types:

$$\begin{aligned}
\mu\alpha(\rho_1,\dots,\rho_k)' &:= \mu\alpha(\mathsf{Lan}\,\rho_1',\dots,\mathsf{Lan}\,\rho_k')\\
\mathsf{Lan}\,\rho &:= \exists\beta.(\beta\to\alpha)\times\rho[\alpha:=\beta]\\
\nu\alpha(\rho_1,\dots,\rho_k)' &:= \nu\alpha(\mathsf{Ran}\,\rho_1',\dots,\mathsf{Ran}\,\rho_k')\\
\mathsf{Ran}\,\rho &:= \forall\beta.(\alpha\to\beta)\to\rho[\alpha:=\beta]
\end{aligned}$$

Terms:

$$\begin{aligned}
(\mathsf{in}_{k,i}\,r)' &:= \mathsf{pack}\langle\lambda x x, r'\rangle\\
(\mathsf{MIt}_k\vec{s})' &:= \lambda x.\mathsf{It}_k(\vec{\mathbb{M}_{\mathsf{Lan}}},\vec{s^{\#}},x)\\
\mathbb{M}_{\mathsf{Lan}} &:= \lambda y\lambda z.\mathsf{open}\big(z,w.\,\mathsf{pack}\langle\lambda x.y\big((\pi_1 w)x\big),\pi_2 w\rangle\big)\\
s_i^{\#} &:= \lambda y.\mathsf{open}\big(y,z.s_i'(\pi_1 z)(\pi_2 z)\big)\\
(\mathsf{MRec}_k\vec{s}\,)' &:= \lambda x.\mathsf{Rec}_k(\vec{\mathbb{M}_{\mathsf{Lan}}},\vec{s^{\diamond}},x)\\
s_i^{\diamond} &:= \lambda y.\mathsf{open}\Big(y,z.s_i'\big(\lambda u.\pi_1\big((\pi_1 z)u\big)\big)\big(\lambda v.\pi_2\big((\pi_1 z)v\big)\big)(\pi_2 z)\Big)\\
(\mathsf{out}_{k,i}\,r)' &:= (\mathsf{out}_{k,i}\,r')(\lambda z.z)\\
(\mathsf{MCoIt}_k\vec{s}\,)' &:= \lambda x.\mathsf{CoIt}_k(\vec{\mathbb{M}_{\mathsf{Ran}}},\vec{\widehat{s}},x)\\
\mathbb{M}_{\mathsf{Ran}} &:= \lambda g\lambda y\lambda f.y(\lambda z.f(gz))\\
\widehat{s_i} &:= \lambda x\lambda f.s_i' f x\\
(\mathsf{MCoRec}_k\vec{s}\,)' &:= \lambda x.\mathsf{CoRec}_k(\vec{\mathbb{M}_{\mathsf{Ran}}},\vec{\widetilde{s}},x)\\
\widetilde{s_i} &:= \lambda x\lambda f.s_i'\Big(\lambda y.f(\mathsf{inl}\,y)\Big)\Big(\lambda z.f(\mathsf{inr}\,z)\Big)x\\
\Big(\mathsf{out}_k^{-1}(\vec{m},\vec{t}\,)\Big)' &:= \mathsf{out}_k^{-1}(\vec{\mathbb{M}_{\mathsf{Ran}}},\vec{\widehat{t}}\,)\\
\widehat{t_i} &:= \lambda g.m_i' g t_i'
\end{aligned}$$

We leave the details of proving that we have indeed an embedding to the reader.

2.5 The Hybrid System $\mathsf{MCICT}_{\mu M\nu}$

This system combines conventional iteration/recursion with Mendler-style coiteration/corecursion. On a first look it seems strange to combine a system in this way, the reason to do it will be clear when first order objects appear in section 6.2.

The system $\mathsf{MCICT}_{\mu M\nu}$ is obtained by extending $\mathsf{F}^{\times}$ with clausular (co)-inductive types through the rules of conventional iteration/recursion $(\mu E), (\mu E^{+})$ and the rules for Mendler-style coiteration/corecursion $(M\nu I), (M\nu I^{+})$ as well as with the rules $(\mu I), (\nu E), (\nu I^{i})$. Observe that again there is no inductive inversion in this system.

It is obvious that this system still enjoys of strong normalization as can be embedded into MCICT for example.

Zehn Ziegen zogen zehn Zentner Zucker zum Zoo.

Deutscher Zungenbrecher

Y cada fin de semana queda el negrito con la ucraniana, y bailan polka y pasito, y soplan vodka y mojito y vuelven trompas por la mañana.

La Casa por la ventana
Joaquin Sabina.

3

Monotone and Clausular (Co)inductive Definitions

We come to the main contribution of this work, an extension of AF2 with a special kind of (co)inductive definitions, namely full-monotone (co)inductive definitions given by clauses, feature which simplifies the mechanism of definition as well as the syntactical shape of the monotonicity witnesses.
Although the extension was designed having in mind the Curry-Howard correspondence starting from the type system MCICT, instead of the terminology of category theory, we use here that of fixed-point theory, which is more usual in logical systems with first-order objects.

3.1 Fixed-Point Theory

Let us recall the basic definitions of fixed-point theory.

Definition 3.1 *Let $\mathcal{P}(A)$ be the power of the set A. A function $\Gamma : \mathcal{P}(A) \to \mathcal{P}(A)$ is called an operator over A. Such operator is monotone if $X \subseteq Y \subseteq A$ implies $\Gamma(X) \subseteq \Gamma(Y)$.*

Definition 3.2 *Let $\Gamma : \mathcal{P}(A) \to \mathcal{P}(A)$ be an operator. A subset $\mathcal{K} \subseteq A$ is called*

- *Γ-closed or pre-fixed point of Γ if $\Gamma(\mathcal{K}) \subseteq \mathcal{K}$.*
- *Γ-supported or post-fixed point of Γ if $\mathcal{K} \subseteq \Gamma(\mathcal{K})$*
- *fixed point of Γ if $\Gamma(\mathcal{K}) = \mathcal{K}$*

- ○ Γ-*inductive if it is included in every pre-fixed point of* Γ*, i.e. if* $\Gamma(X) \subseteq X$ *implies* $\mathcal{K} \subseteq X$.
- ○ Γ-*coinductive if it contains every post-fixed point of* Γ*, i.e. if* $X \subseteq \Gamma(X)$ *implies* $X \subseteq \mathcal{K}$

Lemma 3.1 *The following holds:*

- ○ *There is at most one inductive pre-fixed point and at most one coinductive-post-fixed point.*
- ○ *Inductive pre-fixed points and coinductive post-fixed points of monotone operators are fixed points.*

Proof. Clear ⊣

The previous lemma implies that an inductive (coinductive) fixed point is a least (greatest) fixed point of an operator.

Theorem 3.1 (Knaster-Tarski) *Every monotone operator has an inductive and a coinductive fixed point.*
Proof. Straightforward. ⊣

Extensions of AF2 with least fixed-point primitive constructors have been developed in [Par92, Mir02]. An extension of AF2 with both least and greatest fixed-point primitive constructors can be found in [Raf94].

3.2 The Logic MCICD

In this section we present an extension of $\mathsf{AF2}^{\wedge,\vee}$ with monotone and clausular inductive and coinductive predicates.

Definition 3.3 *A clause is a tuple*

$$\langle \mathcal{F}, \mathbb{c}_1, \ldots, \mathbb{c}_m \rangle$$

such that $\mathcal{F}$ *is a predicate of arity* m *and* $\mathbb{c}_i$ *are given unary function symbols associated to* $\mathcal{F}$ *called tags. The arity of a clause is the arity of its defining predicate* $\mathcal{F}$*, which is also the number of tags in that clause. Clauses will be denoted with the letters* $\mathcal{C}_i, \mathcal{D}_j$.

The following notation will be useful:

If $\mathcal{C}_i = \langle \mathcal{F}_i, \mathbb{c}^i_1, \ldots, \mathbb{c}^i_m \rangle$ we set

$$\vec{\mathbb{c}}_i := \mathbb{c}^i_1, \ldots, \mathbb{c}^i_m,$$

and if $\vec{t} := t_1, \ldots, t_m$, we define

$$\vec{\mathbb{c}_i}\vec{t} := \mathbb{c}^i_1 t_1, \ldots, \mathbb{c}^i_m t_m.$$

Definition 3.4 *An expression of the form*

$$\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k),$$

where $\mathcal{C}_i := \langle \mathcal{F}_i, \vec{\mathbb{c}_i} \rangle$ and X and all the k clauses have the same arity m, is called an inductive predicate. The arity of an inductive predicate is the arity of the variable X. In this case the tags of a clause are called constructors. Analogously a coinductive predicate is an expression of the form

$$\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$$

and we speak of destructors instead of tags.

The predicate $\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ represents the least fixed point of the operator generated by $\mathcal{F}_1 \vee \ldots \vee F_k$ via the constructors $\vec{\mathbb{c}_1}, \ldots, \vec{\mathbb{c}_k}$. Analogously the predicate $\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ represents the greatest fixed point of the operator generated by $\mathcal{F}_1 \wedge \ldots \wedge F_k$ via the destructors $\vec{\mathbb{c}_1}, \ldots, \vec{\mathbb{c}_k}$. The inference rules below will make this intuition clear.

We fix some notation:

$$\begin{array}{rcl}
\mathcal{F} \wedge \mathcal{G} & := & \lambda \vec{z}.\mathcal{F}\vec{z} \wedge \mathcal{G}\vec{z} \\
\mathcal{F} \vee \mathcal{G} & := & \lambda \vec{z}.\mathcal{F}\vec{z} \vee \mathcal{G}\vec{z} \\
\mathcal{K}^{\vec{\mathbb{c}_i}} & := & \lambda \vec{y}.\mathcal{K}(\vec{\mathbb{c}_i}\vec{y}) \\
\mathcal{F} \subseteq \mathcal{G} & := & \forall \vec{y}.\mathcal{F}\vec{y} \to \mathcal{G}\vec{y} \\
\mathcal{F}\,\mathsf{mon}\,X & := & \forall X \forall Y. X \subseteq Y \to \mathcal{F} \subseteq \mathcal{F}[X := Y]
\end{array}$$

The (co)inductive definitions $\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k), \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ where $\mathcal{C}_i := \langle \mathcal{F}_i, \vec{\mathbb{c}_i} \rangle$ and $\mathcal{D}_i := \langle \mathcal{G}_i, \vec{\mathbb{d}_i} \rangle$ are ruled by:

- Folding of the Least Fixed Point: for $1 \leq j \leq k$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)]\vec{t}}{\Gamma \vdash_{\mathbb{E}} \mathsf{in}_{k,j}\, r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{\mathbb{c}_j}\vec{t}} \ (\mu I)$$

- Iteration:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{r} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathbb{c}_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{It}_k(\vec{m}, \vec{s}, r) : \mathcal{K}\vec{r}} \ (\mu E)$$

- Primitive Recursion:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mu X.(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{r} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathbb{c}_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{Rec}_k(\vec{m}, \vec{s}, r) : \mathcal{K}\vec{r}} \ (\mu E^{+})$$

- Coiteration:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mathcal{K}\vec{t} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{G}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{K} \subseteq \mathcal{G}_i[X := \mathcal{K}]^{\vec{\mathbb{d}_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{CoIt}_k(\vec{m}, \vec{s}, r) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}}\ (\nu I)$$

- Primitive Corecursion:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mathcal{K}\vec{t} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{G}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{K} \subseteq \mathcal{G}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\vec{\mathbb{d}_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{CoRec}_k(\vec{m}, \vec{s}, r) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}}\ (\nu I^+)$$

- Folding of the Greatest Fixed Point (Inversion):

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r_i : \mathcal{G}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{d}_i}\vec{t},\ \ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{G}_i\ \mathsf{mon}\ X,\ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{out}_k^{-1}(\vec{m}, \vec{r}) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}}\ (\nu I^i)$$

- Unfolding of the Greatest Fixed Point: for $1 \leq j \leq k$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}}{\Gamma \vdash_{\mathbb{E}} \mathsf{out}_{k,j}\, r : \mathcal{G}_j[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{d}_j}\vec{t},}\ (\nu E)$$

The reader may have noticed that the symmetry between the inductive and coinductive parts is lost because we did not give a rule for unfolding of the least fixed point (inductive inversion) like we did for the corresponding type system MCICT. This rule, having a bad reduction behaviour, would produce more problems than benefits, its main application — to define inductive destructors (like the predeccesor in naturals), can be achieved in a satisfactory way with the rule for primitive recursion. In contrast the rule for coinductive inversion has a good reduction behaviour and it is neccesary to obtain coinductive constructors (like the cons function on streams) in an optimal way.

The proof-reduction is given by the following β-reduction rules between proof-terms:

$$
\begin{array}{rcl}
\mathsf{It}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t) & \mapsto_\beta & s_i\Big(m_i\big(\lambda x.\mathsf{It}_k(\vec{m},\vec{s},x)\big)t\Big) \\[2ex]
\mathsf{Rec}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t) & \mapsto_\beta & s_i\Big(m_i\Big(\langle \mathsf{Id}, \lambda z.\mathsf{Rec}_k(\vec{m},\vec{s},z)\rangle\Big)t\Big) \\[2ex]
\mathsf{out}_{k,i}\,\mathsf{CoIt}_k(\vec{m},\vec{s},t) & \mapsto_\beta & m_i\Big(\lambda z.\mathsf{CoIt}_k(\vec{m},\vec{s},z)\Big)(s_i t) \\[2ex]
\mathsf{out}_{k,i}\,\mathsf{CoRec}_k(\vec{m},\vec{s},t) & \mapsto_\beta & m_i\Big([\mathsf{Id}, \lambda z.\mathsf{CoRec}_k(\vec{m},\vec{s},z)]\Big)(s_i t) \\[2ex]
\mathsf{out}_{k,i}\,\mathsf{out}_k^{-1}(\vec{m},\vec{t}) & \mapsto_\beta & m_i(\lambda z.z)t_i
\end{array}
$$

These rules are obtained, as usual, by normalizing proofs which contain consecutive ocurrences of an introduction and elimination rule for the same formula constructor. They also have a categorical interpretation which was discussed in section 2.1.2.

The described logical system will be called MCICD, a system of Monotone and Clausular Inductive and Coinductive Definitions.

Definition 3.5 *Given an inductive predicate $\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)$ with $\mathcal{C}_i := \langle \mathcal{F}_i, \vec{\mathbb{c}_i}\rangle$, we define the closure, induction and strong induction axioms for $\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)$ as follows*[1]*:*

$$
\begin{array}{lll}
\mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),i} & := & \mathcal{F}_i[X := \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)] \subseteq (\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k))^{\vec{\mathbb{c}_i}} \\[2ex]
\mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)} & := \forall Z. & \mathcal{F}_1\,\mathsf{mon}\,X,\ldots,\mathcal{F}_k\,\mathsf{mon}\,X, \\
 & & \mathcal{F}_1[X := Z] \subseteq Z^{\vec{\mathbb{c}_1}},\ldots,\mathcal{F}_k[X := Z] \subseteq Z^{\vec{\mathbb{c}_k}} \\
 & & \to \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k) \subseteq Z \\[2ex]
\mathsf{Ind}^{+}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)} & := \forall Z. & \mathcal{F}_1\,\mathsf{mon}\,X,\ldots,\mathcal{F}_k\,\mathsf{mon}\,X, \\
 & & \mathcal{F}_1[X := \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k) \wedge Z] \subseteq Z^{\vec{\mathbb{c}_1}}, \\
 & & \qquad\qquad\vdots \\
 & & \mathcal{F}_k[X := \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k) \wedge Z] \subseteq Z^{\vec{\mathbb{c}_k}} \\
 & & \to \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k) \subseteq Z
\end{array}
$$

Analogously, given a coinductive predicate $\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)$ with $\mathcal{D}_i := \langle \mathcal{G}_i, \vec{\mathbb{d}_i}\rangle$, we define the coclosure, coinduction and inversion axioms for $\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)$

[1] Recall that $A_1,\ldots,A_k \to B$ means $A_1 \to \ldots \to A_k \to B$.

as follows:

$$
\begin{array}{rll}
\mathsf{CoCl}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k),i} & := & \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \subseteq (\mathcal{G}_i[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)])^{\vec{\mathbb{d}_i}} \\[1em]
\mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} & := \forall Z. & \mathcal{G}_1 \,\mathsf{mon}\, X,\ldots,\mathcal{G}_k \,\mathsf{mon}\, X, \\
 & & Z \subseteq \mathcal{G}_1[X := Z]^{\vec{\mathbb{d}_1}},\ldots,Z \subseteq \mathcal{G}_k[X := Z]^{\vec{\mathbb{d}_k}} \\
 & & \to Z \subseteq \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \\[1em]
\mathsf{CoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} & := \forall Z. & \mathcal{G}_1 \,\mathsf{mon}\, X,\ldots,\mathcal{G}_k \,\mathsf{mon}\, X, \\
 & & Z \subseteq \mathcal{G}_1[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \vee Z]^{\vec{\mathbb{d}_1}}, \\
 & & \qquad\qquad \vdots \\
 & & Z \subseteq \mathcal{G}_k[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \vee Z]^{\vec{\mathbb{d}_k}} \\
 & & \to Z \subseteq \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \\[1em]
\mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} & := \forall \vec{z}. & \mathcal{G}_1 \,\mathsf{mon}\, X,\ldots,\mathcal{G}_k \,\mathsf{mon}\, X, \\
 & & \mathcal{G}_1[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)]\vec{\mathbb{d}_1}\vec{z}, \\
 & & \qquad\qquad \vdots \\
 & & \mathcal{G}_k[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)]\vec{\mathbb{d}_k}\vec{z} \\
 & & \to \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{z}
\end{array}
$$

Proposition 3.1 *The following holds:*

$$\vdash \lambda x.\, \mathsf{in}_{k,j}\, x : \mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),j}$$

$$\vdash \lambda \vec{m} \lambda \vec{x}. \lambda z. \mathsf{It}_k(\vec{m},\vec{x},z) : \mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$$

$$\vdash \lambda \vec{m} \lambda \vec{x}. \lambda z. \mathsf{Rec}_k(\vec{m},\vec{x},z) : \mathsf{Ind}^{+}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$$

$$\vdash \lambda x.\, \mathsf{out}_{k,j}\, x : \mathsf{CoCl}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k),j}$$

$$\vdash \lambda \vec{m} \lambda \vec{x}. \lambda z. \mathsf{CoIt}_k(\vec{m},\vec{x},z) : \mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$$

$$\vdash \lambda \vec{m} \lambda \vec{x}. \lambda z. \mathsf{CoRec}_k(\vec{m},\vec{x},z) : \mathsf{CoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$$

$$\vdash \lambda \vec{m} \lambda \vec{z}.\, \mathsf{out}_k^{-1}(\vec{m},\vec{z}) : \mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$$

Proof. Straightforward. ⊣

The pet examples of (co)inductive predicates are the natural numbers and the streams of elements of a given set $\mathcal{A}$:

Natural Numbers in MCICD

Given the unit predicate $\mathbb{1}$ with $X \notin FV(\mathbb{1})$ whose unique inhabitant is an object $\star$ (see page 145) and two unary function symbols $0_{\mathsf{g}}, s$ we define the predicate of natural numbers as

$$\mathbb{N} := \mu X\Big(\langle \mathbb{1}, 0_{\mathsf{g}}\rangle, \langle X, s\rangle\Big)$$

The closure axioms are:

- $\mathsf{Cl}_{\mathbb{N},1} := \forall x. \mathbb{1}x \to \mathbb{N}0_{\mathsf{g}}x$
- $\mathsf{Cl}_{\mathbb{N},2} := \forall x. \mathbb{N}x \to \mathbb{N}sx$

The first axiom is reminiscent of the use of global elements in category theory, we do not have a 0-ary constructor 0 but a unary constructor 0_{g} representing a global zero. Observe that as $\mathbb{1}$ only has one inhabitant the axiom $\mathsf{Cl}_{\mathbb{N},1}$ implies that $\mathbb{N}(0_{\mathsf{g}}\star)$, Now we define $0 := 0_{\mathsf{g}}\star$ so that $\mathbb{N}0$ holds.

The induction axiom is:

$$\mathsf{Ind}_{\mathbb{N}} := \forall Z. \mathbb{1} \,\mathsf{mon}\, X, X \,\mathsf{mon}\, X, \mathbb{1} \subseteq Z^{0_{\mathsf{g}}}, Z \subseteq Z^{s} \to \mathbb{N} \subseteq Z$$

It is easy to see that the monotonicity hypothesis are trivially derivable (see section 3.4), therefore the axiom $\mathsf{Ind}_{\mathbb{N}}$ implies the following formula:

$$\forall Z. Z0, (\forall x. Zx \to Zsx) \to \mathbb{N} \subseteq Z$$

which is the usual induction axiom for natural numbers.

Analogously the strong induction axiom

$$\mathsf{Ind}^{+}_{\mathbb{N}} := \forall Z. \mathbb{1} \,\mathsf{mon}\, Z, Z \,\mathsf{mon}\, Z, \mathbb{1} \subseteq Z^{0_{\mathsf{g}}}, \mathbb{N} \land Z \subseteq Z^{s} \to \mathbb{N} \subseteq Z$$

implies the usual strong induction axiom for the natural numbers:

$$\forall Z. Z0, (\forall x. \mathbb{N}x \land Zx \to Zsx) \to \mathbb{N} \subseteq Z$$

Streams in MCICD

Given a predicate $\mathcal{A}$ such that $X \notin FV(\mathcal{A})$ and unary function symbols $\mathsf{head}, \mathsf{tail}$ we define the predicate of streams of elements of $\mathcal{A}$ as

$$\mathcal{S}_{\mathcal{A}} := \nu X\Big(\langle \mathcal{A}, \mathsf{head}\rangle, \langle X, \mathsf{tail}\rangle\Big)$$

The coclosure axioms are:

- $\mathsf{Cocl}_{\mathcal{S}_{\mathcal{A}},1} := \forall x. \mathcal{S}_{\mathcal{A}}x \to \mathcal{A}\,\mathsf{head}\,x$
- $\mathsf{Cocl}_{\mathcal{S}_{\mathcal{A}},2} := \forall x. \mathcal{S}_{\mathcal{A}}x \to \mathcal{S}_{\mathcal{A}}\,\mathsf{tail}\,x$

These axioms show how a stream can be destructed.
The coinduction axiom is

$$\mathsf{CoInd}_{\mathcal{S}_{\mathcal{A}}} := \forall Z. \mathcal{A}\,\mathsf{mon}\,X, X\,\mathsf{mon}\,X, Z \subseteq \mathcal{A}^{\mathsf{head}}, Z \subseteq Z^{\mathsf{tail}} \to Z \subseteq \mathcal{S}_{\mathcal{A}}$$

which implies the usual one:

$$\forall Z. Z \subseteq \mathcal{A}^{\mathsf{head}}, Z \subseteq \mathcal{S}_{\mathcal{A}}^{\mathsf{tail}} \to Z \subseteq \mathcal{S}_{\mathcal{A}}$$

i.e.,

$$\forall Z.(\forall x. Zx \to \mathcal{A}\,\mathsf{head}\,x), (\forall x. Zx \to \mathcal{S}_{\mathcal{A}}\,\mathsf{tail}\,x) \to \forall x. Zx \to \mathcal{S}_{\mathcal{A}}x$$

Analogously the strong coinduction axiom

$$\mathsf{CoCI}^{+}_{\mathcal{S}_{\mathcal{A}}} := \forall Z. \mathcal{A}\,\mathsf{mon}\,X, X\,\mathsf{mon}\,X, Z \subseteq \mathcal{A}^{\mathsf{head}}, Z \subseteq (\mathcal{S}_{\mathcal{A}} \vee Z)^{\mathsf{tail}} \to Z \subseteq \mathcal{S}_{\mathcal{A}}$$

implies the usual one

$$\forall Z. Z \subseteq \mathcal{A}^{\mathsf{head}}, Z \subseteq (\mathcal{S}_{\mathcal{A}} \vee Z)^{\mathsf{tail}} \to Z \subseteq \mathcal{S}_{\mathcal{A}}$$

The usual axioms are easily obtained because the monotonicity assumptions are derivable in an automatic way as we will see in section 3.4.

Subject Reduction

To get this property we just had to simplify the proof for $\mathsf{MCICD}^{\star}$ given in section 4.1.3.

3.3 Strong Normalization of MCICD

We use again a first-order forgetful map on formulas as embedding, obtained by extending the embedding for $\mathsf{AF2}^{\wedge,\vee}$ (see pages 29,30) as follows:

$$\begin{array}{rcl} \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t}\,' & := & \mu X(\mathcal{F}_1', \ldots, \mathcal{F}_k) \\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}\,' & := & \nu X(\mathcal{F}_1', \ldots, \mathcal{F}_k') \end{array}$$

where if $\mathcal{C}_i := \langle \mathcal{F}_i, \vec{\mathfrak{c}_i} \rangle$ and $\mathcal{F}_i := \lambda \vec{y}.G$ then $\mathcal{F}_i' := G'$.

The details of the proof are left to the reader.

3.4 Canonical Monotonicity Witnesses

This section is esentially the same as section 2.3.4. Nevertheless as we have now first-order objects we repeat here some details.

Definition 3.6 (Antimonotonicity) *Given an inductive predicate $\mathcal{F}$ and a variable X, we define the formula $\mathcal{F}\,\mathsf{mon}^-\,X$ as:*

$$\mathcal{F}\,\mathsf{mon}^-\,X := \forall X.\forall Y.(X \subseteq Y) \to \mathcal{F}[X := Y] \subseteq \mathcal{F}$$

Definition 3.7 (Generic (Anti)monotonicity Witnesses) *We define the following* MITC*-terms:*

- $\mathbb{M}_{\mathsf{id}} := \lambda x x$
- $\mathbb{M}_{\mathsf{triv}} := \lambda f \lambda x.x$
- $\mathbb{M}_{\to} := \lambda m_1 \lambda m_2 \lambda f \lambda x \lambda y.m_2 f(x(m_1 f y))$
- $\mathbb{M}_{\forall} := \lambda m \lambda f \lambda x.m f x$
- $\mathbb{M}_{\wedge} := \lambda m_1 \lambda m_2 \lambda f \lambda x.\langle m_1 f(x\pi_1), m_2 f(x\pi_2)\rangle$
- $\mathbb{M}_{\vee} := \lambda m_1 \lambda m_2 \lambda f \lambda x.\mathsf{case}(x, y.\,\mathsf{inl}\, m_1 f y, z.\,\mathsf{inr}\, m_2 f z)$
- $\mathbb{M}_{\mu}^{k} := \lambda \vec{m} \lambda \vec{n} \lambda f \lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x)$*, where* $s_i := \lambda z.\,\mathsf{in}_{k,i}(n_i f z)$.
- $\mathbb{M}_{\nu}^{k} := \lambda \vec{m} \lambda \vec{n} \lambda f \lambda x.\,\mathsf{out}_k^{-1}(\vec{m}, \vec{s})$ *where* $s_i := n_i f\,\mathsf{out}_{k,i}\,x$.

Proposition 3.2 *We have the following derivations:*

- $\vdash \mathbb{M}_{\mathsf{id}} : (\lambda \vec{y}.X\vec{t}\,)\,\mathsf{mon}\,X$
- *If* $X \notin FV(F)$ *then* $\vdash \mathbb{M}_{\mathsf{triv}} : (\lambda \vec{y} F)\,\mathsf{mon}\,X$ *and* $\vdash \mathbb{M}_{\mathsf{triv}} : (\lambda \vec{y} F)\,\mathsf{mon}^-\,X$
- $\vdash \mathbb{M}_{\to} : (\lambda \vec{y} A)\,\mathsf{mon}^-\,X \to (\lambda \vec{y} B)\,\mathsf{mon}\,X \to (\lambda \vec{y}.A \to B)\,\mathsf{mon}\,X$
  $\vdash \mathbb{M}_{\to} : (\lambda \vec{y} A)\,\mathsf{mon}\,X \to (\lambda \vec{y} B)\,\mathsf{mon}^-\,X \to (\lambda \vec{y}.A \to B)\,\mathsf{mon}^-\,X$
- *If ξ is a first or second-order variable, but the same on every formula then:*
  $\vdash \mathbb{M}_{\forall} : (\forall \xi.(\lambda \vec{y} A)\,\mathsf{mon}\,X) \to (\lambda \vec{y}.\forall \xi A)\,\mathsf{mon}\,X$
  $\vdash \mathbb{M}_{\forall} : (\forall \xi.(\lambda \vec{y} A)\,\mathsf{mon}^-\,X) \to (\lambda \vec{y}.\forall \xi A)\,\mathsf{mon}^-\,X$
- $\vdash \mathbb{M}_{\wedge} : (\lambda \vec{y} A)\,\mathsf{mon}\,X \to (\lambda \vec{y} B)\,\mathsf{mon}\,X \to (\lambda \vec{y}.A \wedge B)\,\mathsf{mon}\,X$
  $\vdash \mathbb{M}_{\wedge} : (\lambda \vec{y} A)\,\mathsf{mon}^-\,X \to (\lambda \vec{y} B)\,\mathsf{mon}^-\,X \to (\lambda \vec{y}.A \wedge B)\,\mathsf{mon}^-\,X.$
- $\vdash \mathbb{M}_{\vee} : (\lambda \vec{y} A)\,\mathsf{mon}\,X \to (\lambda \vec{y} B)\,\mathsf{mon}\,X \to (\lambda \vec{y}.A \vee B)\,\mathsf{mon}\,X$
  $\vdash \mathbb{M}_{\vee} : (\lambda \vec{y} A)\,\mathsf{mon}^-\,X \to (\lambda \vec{y} B)\,\mathsf{mon}^-\,X \to (\lambda \vec{y}.A \vee B)\,\mathsf{mon}^-\,X.$

- *If* $\mathcal{C}_i := \langle \lambda\vec{y}B_i, \vec{\mathbb{c}_i}\rangle$ *then*

$$\begin{aligned} \vdash \mathbb{M}^k_\mu : \quad & (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z) \to (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}\,X) \\ & \to (\lambda\vec{y}.\mu Z(\mathcal{C}_1,\dots,\mathcal{C}_k)\vec{t}\,)\,\mathsf{mon}\,X \\ \vdash \mathbb{M}^k_\mu : \quad & (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z) \to (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}^-\,X) \\ & \to (\lambda\vec{y}.\mu Z(\mathcal{C}_1,\dots,\mathcal{C}_k)\vec{t}\,)\,\mathsf{mon}^-\,X \end{aligned}$$

- *If* $\mathcal{D}_i := \langle \lambda\vec{y}B_i, \vec{\mathbb{d}_i}\rangle$ *then*

$$\begin{aligned} \vdash \mathbb{M}^k_\nu : \quad & (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z) \to (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}\,X) \\ & \to (\lambda\vec{y}.\nu Z(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{t}\,)\,\mathsf{mon}\,X \\ \vdash \mathbb{M}^k_\nu : \quad & (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z) \to (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}^-\,X) \\ & \to (\lambda\vec{y}.\nu Z(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{t}\,)\,\mathsf{mon}^-\,X \end{aligned}$$

Proof. Straightforward ⊣

Corollary 3.1 (Derived Rules for (Anti)monotonicity) *The following rules are derivable:*

- $\Gamma \vdash \mathbb{M}_{\mathsf{id}} : (\lambda\vec{y}.X\vec{t}\,)\,\mathsf{mon}\,X$
- *If* $X \notin FV(F)$ *then* $\Gamma \vdash \mathbb{M}_{\mathsf{triv}} : (\lambda\vec{y}F)\,\mathsf{mon}\,X$ *and* $\Gamma \vdash \mathbb{M}_{\mathsf{triv}} : (\lambda\vec{y}F)\,\mathsf{mon}^-\,X$
- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}^-\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\to m_1 m_2 : (\lambda\vec{y}.A \to B)\,\mathsf{mon}\,X$$

- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}^-\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\to m_1 m_2 : (\lambda\vec{y}.A \to B)\,\mathsf{mon}^-\,X$$

- *If* ξ *is a first or second-order variable, but the same in every formula then:*

$$\Gamma \vdash t : \forall\xi.(\lambda\vec{y}A)\,\mathsf{mon}\,X \textit{ implies } \Gamma \vdash \mathbb{M}_\forall t : (\lambda\vec{y}.\forall\xi A)\,\mathsf{mon}\,X$$

$$\Gamma \vdash t : \forall\xi.(\lambda\vec{y}A)\,\mathsf{mon}^-\,X \textit{ implies } \Gamma \vdash \mathbb{M}_\forall t : (\lambda\vec{y}.\forall\xi A))\,\mathsf{mon}^-\,X$$

- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\wedge m_1 m_2 : (\lambda\vec{y}.A \wedge B)\,\mathsf{mon}\,X$$

- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}^-\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}^-\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\wedge m_1 m_2 : (\lambda\vec{y}.A \wedge B)\,\mathsf{mon}^-\,X$$

- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\vee m_1 m_2 : (\lambda\vec{y}.A \vee B)\,\mathsf{mon}\,X$$

- *If* $\Gamma \vdash m_1 : (\lambda\vec{y}A)\,\mathsf{mon}^-\,X$ *and* $\Gamma \vdash m_2 : (\lambda\vec{y}B)\,\mathsf{mon}^-\,X$ *then*

$$\Gamma \vdash \mathbb{M}_\vee m_1 m_2 : (\lambda\vec{y}.A \vee B)\,\mathsf{mon}^-\,X$$

- *If* $\Gamma \vdash m_i : (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z),\ \Gamma \vdash n_i : (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}\,X)$ *and* $\mathcal{C}_i := \langle\lambda\vec{y}B_i, \vec{\mathbb{c}_i}\rangle$, *then*

$$\Gamma \vdash \mathbb{M}^k_\mu \vec{m}\vec{n} : (\lambda\vec{y}.\mu Z(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t})\,\mathsf{mon}\,X$$

- *If* $\Gamma \vdash m_i : (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z),\ \Gamma \vdash n_i : (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}^-\,X)$ *and* $\mathcal{C}_i := \langle\lambda\vec{y}B_i, \vec{\mathbb{c}_i}\rangle$, *then*

$$\Gamma \vdash \mathbb{M}^k_\mu \vec{m}\vec{n} : (\lambda\vec{y}.\mu Z(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t})\,\mathsf{mon}^-\,X$$

- *If* $\Gamma \vdash m_i : (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z),\ \Gamma \vdash n_i : (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}\,X)$ *and* $\mathcal{D}_i := \langle\lambda\vec{y}B_i, \vec{\mathbb{d}_i}\rangle$, *then*

$$\Gamma \vdash \mathbb{M}^k_\nu \vec{m}\vec{n} : (\lambda\vec{y}.\nu Z(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t})\,\mathsf{mon}\,X$$

- *If* $\Gamma \vdash m_i : (\forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z),\ \Gamma \vdash n_i : (\forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}^-\,X)$ *and* $\mathcal{D}_i := \langle\lambda\vec{y}B_i, \vec{\mathbb{d}_i}\rangle$, *then*

$$\Gamma \vdash \mathbb{M}^k_\nu \vec{m}\vec{n} : (\lambda\vec{y}.\nu Z(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t})\,\mathsf{mon}^-\,X$$

Proof. Trivial ⊣

Definition 3.8 *We will write* $\Gamma \vdash^{\mathsf{can}} m : \forall\vec{\xi}.\mathcal{F}\,\mathsf{mon}\,X$ *if* m *was obtained from* Γ *by one of the rules of the previous corollary (possibly using also the rules for universal quantifiers). We say that a monotonicity witness* m *is canonical if* $\vdash^{\mathsf{can}} m : \mathcal{F}\,\mathsf{mon}\,X$*.*

The following proposition corresponds to proposition 2.14.

Proposition 3.3 (First Functor Law) *If* $\Gamma \vdash^{\mathsf{can}} m : \forall\vec{\xi}.\mathcal{F}\,\mathsf{mon}\,X$ *and* $\Gamma \vdash^{\mathsf{can}} m^- : \forall\vec{\chi}.\mathcal{G}\,\mathsf{mon}^-\,X$ *then* m *and* m^- *satisfy the first functor law in* $\mathsf{MCICT}\eta$*, that is:*

$$m(\lambda z.z) \to^\star_{\beta\eta} \lambda y.y \quad m^-(\lambda z.z) \to^\star_{\beta\eta} \lambda y.y$$

Proof. Induction on $\vdash^{\mathsf{can}}$. The cases for $(\forall I), (\forall E)$ are trivial from the IH, the other cases can be distinguished according to the form of F.

- Case $F \equiv X\vec{t}$. We have $m \equiv \mathbb{M}_{\mathsf{id}} \equiv \lambda x.x$. Therefore

$$m(\lambda z.z) \equiv (\lambda x.x)(\lambda z.z) \to_\beta \lambda z.z =_\alpha \lambda y.y$$

- Case $X \notin FV(F)$. Then $m \equiv m^- \equiv \mathbb{M}_{\mathsf{triv}} \equiv \lambda f.\lambda x.x$. Therefore

$$m(\lambda z.z) \equiv m^-(\lambda z.z) \equiv (\lambda f.\lambda x.x)(\lambda z.z) \to_\beta \lambda x.x =_\alpha \lambda y.y$$

- Case $F \equiv A \to B$. We have $\Gamma \vdash^{\mathsf{can}} m_1 : (\lambda\vec{y}A)\,\mathsf{mon}^-\,X$, $\Gamma \vdash^{\mathsf{can}} m_2 : (\lambda\vec{y}B)\,\mathsf{mon}\,X$ and $m \equiv \mathbb{M}_{\to}m_1m_2$. By IH we have $m_i(\lambda z.z) \to^\star_{\beta\eta} \lambda u.u,\ i = 1, 2$. Therefore

$$\begin{gathered} m(\lambda z.z) \equiv \mathbb{M}_{\to}m_1m_2(\lambda z.z) \equiv \\ (\lambda m_1.\lambda m_2.\lambda f.\lambda x.\lambda y.m_2f(x(m_1fy)))m_1m_2(\lambda z.z) \to^\star_\beta \\ \lambda x.\lambda y.m_2(\lambda z.z)(x(m_1(\lambda z.z)y)) \to^\star_{\beta\eta} \lambda x.\lambda y.(\lambda u.u)(x(m_1(\lambda z.z)y)) \to^\star_{\beta\eta} \\ \lambda x.\lambda y.(\lambda u.u)(x((\lambda u.u)y)) \to^\star_\beta \lambda x.\lambda y.xy \to_\eta \lambda x.x =_\alpha \lambda y.y \end{gathered}$$

  The subcase for m^- is analogous.

- Case $F \equiv \forall\xi A$. Then $m^- \equiv \mathbb{M}_\forall m_1$ with $\Gamma \vdash^{\mathsf{can}} m_1 : \forall\xi.(\lambda\vec{y}.A)\,\mathsf{mon}^-\,X$, by IH we have $m_1(\lambda z.z) \to^\star_{\beta\eta} \lambda u.u$. Therefore

$$\begin{gathered} m^-(\lambda z.z) \equiv \mathbb{M}_\forall m_1(\lambda z.z) \equiv (\lambda m.\lambda f\lambda x.mfx)m_1(\lambda z.z) \to^\star_\beta \\ \lambda x.m_1(\lambda z.z)x \to^\star_{\beta\eta} \lambda x.(\lambda u.u)x \to_\beta \lambda x.x =_\alpha \lambda y.y \end{gathered}$$

  The subcase for m is analogous.

- Case $F \equiv A \wedge B$. Then $m \equiv \mathbb{M}_\wedge m_1m_2$ with $\Gamma \vdash^{\mathsf{can}} m_1 : (\lambda\vec{y}A)\,\mathsf{mon}\,X$, $\Gamma \vdash^{\mathsf{can}} m_2 : (\lambda\vec{y}B)\,\mathsf{mon}\,X$. By IH we have $m_i(\lambda z.z) \to^\star_{\beta\eta} \lambda u.u,\ i = 1, 2$. Therefore

$$\begin{gathered} m(\lambda z.z) \equiv \mathbb{M}_\wedge m_1m_2(\lambda z.z) \equiv \\ (\lambda m_1.\lambda m_2.\lambda f.\lambda x.\langle m_1f(x\pi_1), m_2f(x\pi_2)\rangle)m_1m_2(\lambda z.z) \to^\star_\beta \\ \lambda x.\langle m_1(\lambda z.z)(x\pi_1), m_2(\lambda z.z)(x\pi_2)\rangle \to^\star_\beta \lambda x.\langle(\lambda u.u)(x\pi_1), (\lambda u.u)(x\pi_2)\rangle \to^\star_\beta \\ \lambda x.\langle x\pi_1, x\pi_2\rangle \to_\eta \lambda x.x =_\alpha \lambda y.y \end{gathered}$$

  The subcase for m^- is analogous.

- Case $F \equiv \mu Z(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{t}$. Then $m \equiv \mathbb{M}^k_\mu\vec{m}\vec{n}$ with

$$\Gamma \vdash^{\mathsf{can}} m_i : \forall X.(\lambda\vec{y}B_i)\,\mathsf{mon}\,Z, \Gamma \vdash^{\mathsf{can}} n_i : \forall Z.(\lambda\vec{y}B_i)\,\mathsf{mon}\,X.$$

  By IH we have $n_i(\lambda z.z) \to^\star_{\beta\eta} \lambda u.u$.

$$\begin{gathered} m(\lambda z.z) \equiv \mathbb{M}^k_\mu\vec{m}\vec{n}(\lambda z.z) \equiv (\lambda\vec{m}.\lambda\vec{y}.\lambda f.\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x))\vec{m}\vec{n}(\lambda z.z) \to^\star_\beta \\ (\lambda f.\lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x))(\lambda z.z) \equiv (\lambda f.\lambda x.\mathsf{It}_k(\vec{m}, (\lambda w.\,\mathsf{in}_{k,i}(n_ifw)), x))(\lambda z.z) \\ \to_\beta \lambda x.\mathsf{It}_k(\vec{m}, (\lambda w.\,\mathsf{in}_{k,i}(n_i(\lambda z.z)w)), x) \to^\star_{\beta\eta} \lambda x.\mathsf{It}_k(\vec{m}, (\lambda w.\,\mathsf{in}_{k,i}((\lambda u.u)w)), x) \\ \to_\beta \lambda x.\mathsf{It}_k(\vec{m}, (\lambda w.\,\mathsf{in}_{k,i}\,w), x) \equiv \lambda x.\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1 \dots \mathbb{C}^k_k, x) \to_\eta \lambda x.x =_\alpha \lambda y.y \end{gathered}$$

- ○ Case $F \equiv \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{t}$. Then $m \equiv \mathbb{M}^k_\nu \vec{m}\vec{n}$ with

$$\Gamma \vdash^{\mathsf{can}} m_i : \forall X.(\lambda \vec{y} B_i) \,\mathsf{mon}\, Z, \Gamma \vdash^{\mathsf{can}} n_i : \forall Z.(\lambda \vec{y} B_i) \,\mathsf{mon}\, X.$$

By IH we have $n_i(\lambda z.z) \to^\star_{\beta\eta} \lambda u.u$.

$$\begin{gathered} m(\lambda z.z) \equiv \mathbb{M}^k_\nu \vec{m}\vec{n}(\lambda z.z) \equiv \big(\lambda \vec{m}.\lambda \vec{n}.\lambda f.\lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, \vec{s})\big)\vec{m}\vec{n}(\lambda z.z) \to^\star_\beta \\ \big(\lambda f.\lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, \vec{s})\big)(\lambda z.z) \equiv \big(\lambda f.\lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, n_i f\, \mathsf{out}_{k,i}\, x)\big)(\lambda z.z) \\ \to_\beta \lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, n_i(\lambda z.z)(\mathsf{out}_{k,i}\, x)) \to^\star_{\beta\eta} \lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, (\lambda u.u)(\mathsf{out}_{k,i}\, x)) \\ \to_\beta \lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, \mathsf{out}_{k,i}\, x) \equiv \lambda x.\, \mathsf{out}_k^{-1}(\vec{m}, \mathbb{D}^k_1 x, \dots, \mathbb{D}^k_k x) \to_\eta \lambda x.x =_\alpha \lambda y.y \end{gathered}$$

⊣

The following proposition implies that our framework includes all positive definitions.

Proposition 3.4 *If X ocurrs positively in $\mathcal{F}$ then there exists an m such that* $\vdash^{\mathsf{can}} m : \mathcal{F} \,\mathsf{mon}\, X$.
Proof. This well-known fact is proved by induction on F. ⊣

Je weniger die Leute davon wissen, wie Würste und Gesetze gemacht werden, desto besser schlafen sie.

Otto von Bismarck (1815-1898).

En el ll-Ahau *es cuando salió* Ah mucen cab *a poner vendas en los ojos de* Oxlahun ti ku, *Trece-deidad...*

Poema maya.

4 Realizability for MCICD

Realizability has been used extensively in proof theory to prove consistency and proof-theoretical strength of logical systems (see [Tro98]) and recently also as a tool in computer science to extract programs from proofs (see for example [Ber93, BBS02, Tat93, KrPa90]).
Realizability interpretations are given by saying what it means for computational objects of some kind to *realize* logical formulas. In our case the computational objects (programs) are modelled by terms taken from the type system MCICT^- whereas the specifications are formulas of the logic MCICD. The concept "the program t realizes the specification A" will be formalized by means of a new formula $t\ \mathtt{r}\ A$, which belongs to an extended logic $\mathsf{MCICD}^\star$.

4.1 The Logic $\mathsf{MCICD}^\star$

$\mathsf{MCICD}^\star$ is an extension of MCICD over the term system MCICT^- and with first order existential formulas and restricted formulas.

4.1.1 Definition of the Logic

We extend MCICD as follows:

- We add first-order existential and restricted formulas (defined below)
- We extend the term system to MCICT^-.
- Tags in clauses can be either function symbols (considered as constants added to MCICT^-) or closed terms of MCICT^-.

Existential Formulas

Existential formulas are ruled by:

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A[x := s]}{\Gamma \vdash_{\mathbb{E}} \mathsf{pack}\, t : \exists x A}\ (\exists I) \quad \frac{\Gamma \vdash_{\mathbb{E}} t : \exists x A \quad \Gamma, z : A[x := u] \vdash_{\mathbb{E}} r : B}{\Gamma \vdash_{\mathbb{E}} \mathsf{open}(t, z.r) : B}\ (\exists E)$$

where in the $(\exists E)$ rule, $u \notin FV(\Gamma, B, \exists x A)$.

Proof reduction is given by the following β-reduction rule:

$$\mathsf{open}(\mathsf{pack}\, t, z.r) \quad \mapsto_\beta \quad r[z := t]$$

The reader may have noticed that the rules for existential formulas are given only in partial Curry-style, i.e. the rules are traceable. The reason is that the rules in full Curry-style will cause the subject reduction property to fail, as in the following example:

The rules for existential in full Curry-style are:

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A[x := s]}{\Gamma \vdash_{\mathbb{E}} t : \exists x A}\ (\exists I') \quad \frac{\Gamma \vdash_{\mathbb{E}} t : \exists x A \quad \Gamma, z : A[x := u] \vdash_{\mathbb{E}} r : B}{\Gamma \vdash_{\mathbb{E}} r[z := t] : B}\ (\exists E')$$

where in the $(\exists E')$ rule, $u \notin FV(\Gamma, B, \exists x A)$.

Take $\Gamma = \{x : \forall x.C \to C \to A, y : B \to \exists x C, z : B\}$ with $x \notin FV(A, B)$ and therefore $x \notin FV(\Gamma)$. We have

$$\Gamma \vdash (\lambda uu)yz : \exists x C \quad \Gamma, v : C \vdash : xvv : A.$$

Therefore by $(\exists E')$ we get

$$\Gamma \vdash \big(xvv\big)[v := (\lambda uu)yz] : A$$

that is,

$$\Gamma \vdash x\Big((\lambda uu)yz\Big)\Big((\lambda uu)yz\Big) : A.$$

We have $x\Big((\lambda uu)yz\Big)\Big((\lambda uu)yz\Big) \to_\beta x\Big((\lambda uu)yz\Big)\Big(yz\Big)$, but

$$\Gamma \not\vdash x\Big((\lambda uu)yz\Big)\Big(yz\Big) : A$$

This can be seen because due to the variable condition we cannot get neither $\Gamma \vdash x : \exists x C \to \exists x C \to A$ nor $\Gamma \vdash (\lambda uu)yz : C, \Gamma \vdash yz : C$.

Restricted Formulas

We will need Parigot's restriction to be able to formulate realizability for disjunctions:

Restricted formulas are expressions of the form

$$A\restriction s_1 = t_1, \ldots, s_k = t_k$$

The restricted formula represents a conjunction

$$A \wedge s_1 = t_1 \wedge \ldots \wedge s_k = t_k.$$

Restriction behaves according to the following rules, where we abbreviate the sequence of equations as $\vec{s} = \vec{t}$:

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A \quad \Gamma \vdash_{\mathbb{E}} \vec{s} = \vec{t}}{\Gamma \vdash_{\mathbb{E}} r : A\restriction \vec{s} = \vec{t}}\ (\restriction I)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A\restriction \vec{s} = \vec{t}}{\Gamma \vdash_{\mathbb{E}} r : A}\ (\restriction E)$$

Observe that the treatment of equalities cannot be independent in this system as before, because now we have equalities inside restricted formulas which may appear in a context Γ. The notation $\Gamma \vdash_{\mathbb{E}} s = t$ ocurring in the (Eq) rule means now a derivation obtained with the above rules, the derived rules given in page 27 or the following rule:

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A\restriction \vec{s} = \vec{t}}{\Gamma \vdash_{\mathbb{E}} s_i = t_i}\ (\restriction E_R)$$

We fixed now a basic context of equalities:

$$\mathbb{E}_\beta := \{t = r \mid t \to_\beta r \text{ or } r \to_\beta t\},$$

Therefore we have β-equality but only for one-step reduction.

Unless stated otherwise, while working in MCICD*, we will write $\vdash$ for $\vdash_{\mathbb{E}_\beta}$.

4.1.2 Strong Normalization of MCICD*

This is proven as for MCICD by an embedding into MCICT$^-$. The case for restricted formulas being:

$$\big(A\restriction \vec{s} = \vec{t}\,\big)' := A'$$

4.1.3 Subject Reduction for MCICD*

In this section we prove subject reduction for MCICD* the proof is based in both Krivine's Proof for system F (see [Kri93]) and the proof for Rafalli's system AF2$^{\mu\nu}$ (see [Raf94]). MCICD* is the most complex system in this work, the subject-reduction of the source logic MCICD and of the type system MCICT can be easily achieved by adapting (simplifying) the proof in this section.

We fix some notation, if $\Gamma = \{x_1 : A_1, \ldots, x_k : A_k\}$ then

$$\begin{array}{rcl} \Gamma[\gamma := \chi] & := & \{x_1 : A_1[\gamma := \chi], \ldots, x_k : A_k[\gamma := \chi]\}. \\ \Gamma[\vec{y}/\vec{x}\,] & := & \{y_1 : A_1, \ldots, y_k : A_k\}. \end{array}$$

Definition 4.1 *If* Π *is a derivation of* $\Gamma \vdash_{\mathbb{E}} r : A$ *we will denote with* $\Pi[\gamma := \chi]$ *the derivation obtained by substituting every judgement* $\Delta \vdash_{\mathbb{E}'} s : B$ *in* Π *with* $\Delta[\gamma := \chi] \vdash_{\mathbb{E}'[\gamma:=\chi]} s : B[\gamma := \chi]$.

The next lemma shows that $\Pi[\gamma := \chi]$ is indeed a derivation of

$$\Gamma[\gamma := \chi] \vdash_{\mathbb{E}[\gamma:=\chi]} r : A[\gamma := \chi].$$

Moreover the proof implies that the structure of such derivation remains the same, i.e. if $\Delta \vdash s : B$ was obtained by the inference rule $\mathcal{R}$ within Π then $\Delta[\gamma := \chi] \vdash_{\mathbb{E}[\gamma:=\chi]} s : B[\gamma := \chi]$ is also obtained by $\mathcal{R}$ in $\Pi[\gamma := \chi]$.

Lemma 4.1 (Substitution Properties for Derivations) *The following properties hold:*

- *If* $\Gamma, x_1 : A_1, \ldots, x_k : A_k \vdash_{\mathbb{E}} r : B$ *and* $\Gamma \vdash_{\mathbb{E}} s_i : A_i$ *then*
$$\Gamma \vdash_{\mathbb{E}} r[\vec{x} := \vec{s}\,] : B. \quad (Dsp1)$$
- *If* Π *is a derivation of* $\Gamma \vdash t : A$ *then* $\Pi[x := r]$ *is a derivation of*
$$\Gamma[x := r] \vdash_{\mathbb{E}[x:=r]} t : A[x := r]. \quad (Dsp2)$$
- *If* Π *is a derivation of* $\Gamma \vdash t : A$ *then* $\Pi[X := \mathcal{F}]$ *is a derivation of*
$$\Gamma[X := \mathcal{F}] \vdash_{\mathbb{E}} t : A[X := \mathcal{F}]. \quad (Dsp3)$$
- *If* $\Gamma \vdash_{\mathbb{E}} r : A$ *then*
$$\Gamma[\vec{y}/\vec{x}\,] \vdash_{\mathbb{E}} r[\vec{x} := \vec{y}\,] : A. \quad (Dsp4)$$
- *If* $\Gamma \vdash_{\mathbb{E}} s = t$ *and* $\Gamma, x : A[x := s] \vdash_{\mathbb{E}} r : B[x := s]$ *then*
$$\Gamma, x : A[x := t] \vdash_{\mathbb{E}} r : B[x := t]. \quad (Dsp5)$$

Proof.

- (Dsp1). Induction on $\Gamma, x_1 : A_1, \ldots, x_k : A_k \vdash r : B$.
Simultaneously we need to prove that if $\Gamma, x : 1 : A_1, \ldots, x_k : A_k \vdash_{\mathbb{E}} s = t$ and $\Gamma \vdash_{\mathbb{E}} s_i : A_i$ then $\Gamma \vdash_{\mathbb{E}} s = t$.
- (Dsp2). Induction on $\vdash$. Simultaneosly we need to prove that if $\Gamma \vdash_{\mathbb{E}} s = t$ then $\Gamma[x := r] \vdash_{\mathbb{E}[x:=r]} s[x := r] = t[x := r]$.

- ○ (Dsp3). Induction on $\vdash$. Proving simultaneouly that if $\Gamma \vdash_{\mathbb{E}} s = t$ then $\Gamma[X := \mathcal{F}] \vdash_{\mathbb{E}} s = t$.

- ○ (Dsp4). Induction on $\vdash$. Proving simultaneously that if $\Gamma \vdash_{\mathbb{E}} s = t$ then $\Gamma[\vec{y}/\vec{x}\,] \vdash_{\mathbb{E}} s = t$.

- ○ (Dsp5). By weakening we have $\Gamma, x : A[x := s] \vdash_{\mathbb{E}} s = t$, therefore using (Eq) we get $\Gamma, x : A[x := s] \vdash_{\mathbb{E}} r : B[x := t]$ and again by weakening we have
$$\Gamma, y : A[x := t], x : A[x := s] \vdash_{\mathbb{E}} r : B[x := t].$$
Next observe that from the trivial $\Gamma, y : A[x := t] \vdash_{\mathbb{E}} y : A[x := t]$ we get by (Eq) (weakening needed again in the equality derivation)
$$\Gamma, y : A[x := t] \vdash_{\mathbb{E}} y : A[x := s],$$
Applying $(Dsp1)$ we conclude
$$\Gamma, y : A[x := t] \vdash_{\mathbb{E}} r[x := y] : B[x := t],$$
Finally $(Dsp4)$ yields $\Gamma, x : A[x := t] \vdash_{\mathbb{E}} r : B[x := t]$.

$\dashv$

Definition 4.2 *Given a formula A, a context Γ and an equational context $\mathbb{E}$ we define the set $\mathcal{C}_{\Gamma,\mathbb{E}}(A)$ of $\Gamma, \mathbb{E}$-instances of A as the least class of formulas such that:*

- ○ $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$ $\quad (I1)$

- ○ *If $B \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$ and $x \notin FV(\Gamma, \mathbb{E})$ then $B[x := t] \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$.* $\quad (I2)$

- ○ *If $B \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$ and $X \notin FV(\Gamma)$ then $B[X := \mathcal{F}] \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$.* $\quad (I3)$.

- ○ *If $B[x := r] \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$ and $\Gamma \vdash_{\mathbb{E}} r = s$ then $B[x := s] \in \mathcal{C}_{\Gamma,\mathbb{E}}(A)$.* $\quad (I4)$

To prove an inclusion between two sets of $\Gamma, \mathbb{E}$-instances say $\mathcal{C}_{\Gamma,\mathbb{E}}(A) \subseteq \mathcal{C}_{\Gamma,\mathbb{E}}(B)$ we will use the minimality of the class $\mathcal{C}_{\Gamma,\mathbb{E}}(A)$. Therefore it suffices to show that the four defining properties of $\mathcal{C}_{\Gamma,\mathbb{E}}(A)$ hold for $\mathcal{C}_{\Gamma,\mathbb{E}}(B)$. But $I2 - I4$ obviously hold for $\mathcal{C}_{\Gamma,\mathbb{E}}(B)$, for they are also part of its definition. Therefore we only need to prove $(I1)$ in detail, namely that $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$. This remark will be useful to prove the following

Lemma 4.2 (Properties of $\mathcal{C}_{\Gamma,\mathbb{E}}$) *The following properties hold:*

1. *If $x \notin FV(\Gamma, \mathbb{E})$ then $\mathcal{C}_{\Gamma,\mathbb{E}}(B[x := s]) \subseteq \mathcal{C}_{\Gamma,\mathbb{E}}(B)$.*

2. *If $X \notin FV(\Gamma)$ then $\mathcal{C}_{\Gamma,\mathbb{E}}(B[X := \mathcal{F}]) \subseteq \mathcal{C}_{\Gamma,\mathbb{E}}(B)$.*

3. *If $\Gamma \vdash_{\mathbb{E}} r = s$ then $\mathcal{C}_{\Gamma,\mathbb{E}}(B[x := s]) \subseteq \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := r])$.*

Proof. We will use the previous remark.

1. By I1, $B \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$ which implies, as $x \notin FV(\Gamma,\mathbb{E})$, that $B[x := s] \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$.
2. By I1, $B \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$ which implies, as $X \notin FV(\Gamma)$, that $B[X := \mathcal{F}] \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$.
3. By I1, $B[x := r] \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := r])$ which implies, as $\Gamma \vdash_{\mathbb{E}} r = s$, with I4, that $B[x := s] \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := r])$.

⊣

Definition 4.3 *A formula A is an open formula if it is neither an universal quantification nor a restricted formula. The interior of a formula A, denoted A° is defined as follows:*

$$\begin{array}{rcll} A^\circ & := & A, & \textit{if } A \textit{ is open.} \\ (\forall\gamma A)^\circ & := & A^\circ & \\ (A\upharpoonright \vec{s} = \vec{t}\,)^\circ & := & A^\circ & \end{array}$$

Observe that existential formulas $\exists x A$ are open.

Lemma 4.3 $A[\vec{x} := \vec{s}]^\circ = A^\circ[\vec{x} := \vec{s}]$.
Proof. Induction on A. If A is open then $A \equiv A^\circ$ and the claim is obvious.
$\big((\forall\gamma B)[\vec{x} := \vec{s}]\big)^\circ \equiv (\forall\gamma.B[\vec{x} := \vec{s}])^\circ \equiv B[\vec{x} := \vec{s}]^\circ \underset{IH}{\equiv} B^\circ[\vec{x} := \vec{s}] \equiv (\forall\gamma B)^\circ[\vec{x} := \vec{s}]$.
$\big((B\upharpoonright \vec{r} = \vec{t}\,)[\vec{x} := \vec{s}]\big)^\circ \equiv \big(B[\vec{x} := \vec{s}]\upharpoonright \vec{r}[\vec{x} := \vec{s}] = \vec{t}[\vec{x} := \vec{s}]\big)^\circ \equiv B[\vec{x} := \vec{s}]^\circ \underset{IH}{\equiv} B^\circ[\vec{x} := \vec{s}] \equiv (B\upharpoonright \vec{r} = \vec{t}\,)^\circ[\vec{x} := \vec{s}]$. ⊣

Lemma 4.4 $B[X := \mathcal{F}]^\circ = \begin{cases} B^\circ[X := \mathcal{F}] & \textit{If } B^\circ \neq X\vec{t} \\ B^\circ[X := \mathcal{F}^\circ] & \textit{If } B^\circ = X\vec{t} \end{cases}$

Proof. First assume $B^\circ = X\vec{t}$. Then B is either of the form $\forall\vec{\gamma}.X\vec{t}$ or $\forall\vec{\gamma}.X\vec{t}\upharpoonright \vec{s} = \vec{r}$. We analize the second case, as the first is easier:

$$B[X := \mathcal{F}]^\circ = (\forall\vec{\gamma}.X\vec{t}\upharpoonright \vec{s} = \vec{r}\,)[X := \mathcal{F}]^\circ = (\forall\vec{\gamma}.(X\vec{t}\,)[X := \mathcal{F}]\upharpoonright \vec{s} = \vec{r}\,)^\circ =$$

$$(\forall\vec{\gamma}.\mathcal{F}\vec{t}\upharpoonright \vec{s} = \vec{r}\,)^\circ = (\mathcal{F}\vec{t})^\circ = F[\vec{y} := \vec{t}]^\circ = F^\circ[\vec{y} := \vec{t}] = \mathcal{F}^\circ\vec{t} =$$

$$(X\vec{t})[X := \mathcal{F}^\circ] = B^\circ[X := \mathcal{F}^\circ]$$

For the case $B^\circ \neq X\vec{t}$ we show $B[X := \mathcal{F}]^\circ = B^\circ[X := \mathcal{F}]$ by induction on B.

- ○ If B is open then $B^\circ = B$. The assumption $B^\circ \neq X\vec{t}$ implies that $B[X := \mathcal{F}]$ is of the same form as B, i.e., is also open, therefore

$$B[X := \mathcal{F}]^\circ = B[X := \mathcal{F}] = B^\circ[X := \mathcal{F}].$$

- ○ If $B \equiv \forall\gamma A$ then $B^\circ = A^\circ \neq X\vec{t}$ and

$$B[X := \mathcal{F}]^\circ = \quad (\forall\gamma.A[X := \mathcal{F}])^\circ = A[X := \mathcal{F}]^\circ \underset{IH}{=}$$

$$A^\circ[X := \mathcal{F}] = B^\circ[X := \mathcal{F}]$$

- ○ If $B \equiv A{\restriction}\vec{s} = \vec{t}$ then $B^\circ = A^\circ \neq X\vec{t}$. Then

$$B[X := \mathcal{F}]^\circ = \quad (A[X := \mathcal{F}]{\restriction}\vec{s} = \vec{t}\,)^\circ = A[X := \mathcal{F}]^\circ \underset{IH}{=}$$

$$A^\circ[X := \mathcal{F}] = B^\circ[X := \mathcal{F}]$$

⊣

The concept of non-traceable rule is generalized as follows,

Definition 4.4 *We say that an inference rule is non-traceable if its application is not reflected in the proof-term system, i.e. if the proof-term of its conclusion equals that of its non-equational premiss. In our system the non-traceable rules are the four rules for* $\forall, \forall^2$*, the rule for equality* (Eq) *an the rules for restriction* $({\restriction}\, I), ({\restriction}\, E)$*. A not non-traceable rule is called traceable.*

Lemma 4.5 (Main Lemma) *Let* $\widetilde{A}$ *be an open formula. If* $\Gamma \vdash_{\mathbb{E}} t : \widetilde{A}$ *is derived from* $\Gamma \vdash_{\mathbb{E}^\star} t : A$ *using only non-traceable rules then* $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$
Proof. Induction on the number of steps in the derivation of $\Gamma \vdash_{\mathbb{E}} t : \widetilde{A}$ from $\Gamma \vdash_{\mathbb{E}^\star} t : A$. Case Analysis on the first rule used in that derivation.

- ○ $(\forall I)$. We have $\Gamma \vdash_{\mathbb{E}} t : \widetilde{A}$ from $\Gamma \vdash_{\mathbb{E}^\star} t : \forall x A$ where $x \notin FV(\Gamma)$, therefore by IH we get $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}((\forall x A)^\circ)$. But $(\forall x A)^\circ \equiv A^\circ$ therefore $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$.

- ○ $(\forall I^2)$ Analogous to $(\forall I)$.

- ○ $(\forall E)$. We have $A \equiv \forall x B$. $\Gamma \vdash_{\mathbb{E}} t : \widetilde{A}$ is obtained $\Gamma \vdash_{\mathbb{E}^\star} t : B[x := s]$. Therefore by IH we get $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := s]^\circ)$, which by lemma 4.3 is the same as $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ[x := s])$. Finally by property 1 of lemma 4.2 as w.l.o.g. $x \notin FV(\Gamma)$ we conclude $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$. That is $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$.

- ○ $(\forall^2 E)$. We have $A \equiv \forall X B$ and after $(\forall^2 E)$,$\Gamma \vdash_{\mathbb{E}^\star} t : B[X := \mathcal{F}]$. By IH we have $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[X := \mathcal{F}]^\circ)$. We have two subcases:

  - $B^\circ \neq X\vec{t}$. Lemma 4.4 implies that $B[X := \mathcal{F}]^\circ = B^\circ[X := \mathcal{F}]$. Therefore we have $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ[X := \mathcal{F}])$, which implies by lemma 4.2 part 2, as w.l.o.g. $X \notin FV(\Gamma)$, that $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$ i.e.,$\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$

  - $B^\circ \equiv X\vec{t}$. Lemma 4.4 yields $B[X := \mathcal{F}]^\circ = B^\circ[X := \mathcal{F}^\circ]$. Hence we have $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ[X := \mathcal{F}^\circ])$,which implies by lemma 4.2 part 2, as w.l.o.g. $X \notin FV(\Gamma)$, that $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$ i.e.,$\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$

- ○ (Eq). We have $A \equiv B[x := r]$ and $\mathbb{E}_\Gamma, \mathbb{E}^\star \vdash r = s$. $\Gamma \vdash_{\mathbb{E}} t : \widetilde{A}$ is obtained from $\Gamma \vdash_{\mathbb{E}^\star} t : B[x := s]$. By IH we get $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := s]^\circ)$, lemma 4.3 yields $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ[x := s])$. Finally by property 3 of lemma 4.2, as $\mathbb{E}_\Gamma, \mathbb{E} \vdash r = s$ (because $\mathbb{E}^\star \subseteq \mathbb{E}$), we conclude $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ[x := r])$ which by lemma 4.3 yields $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B[x := r]^\circ)$, i.e., $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$.

- ○ $(\upharpoonright I)$. we have $\Gamma \vdash_{\mathbb{E}^\star} t : A \upharpoonright \vec{s} = \vec{t}$ obtained from $\Gamma \vdash_{\mathbb{E}^\star} t : A$, $\mathbb{E}_\Gamma, \mathbb{E}^\star \vdash \vec{s} = \vec{t}$. The IH yields $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}\big((A \upharpoonright \vec{s} = \vec{t})^\circ\big)$, i.e. $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(A^\circ)$.

- ○ $(\upharpoonright E)$. We have $A \equiv B \upharpoonright \vec{s} = \vec{t}$ and $\Gamma \vdash_{\mathbb{E}^\star} t : B$ coming from $\Gamma \vdash_{\mathbb{E}^\star} t : B \upharpoonright \vec{s} = \vec{t}$. The IH yields $\widetilde{A} \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$. But $B^\circ = A^\circ$, therefore we are done.

$\dashv$

Definition 4.5 *A proof-term t is called an I-term if it was generated by an introduction rule, i.e., I-terms are terms of the following shapes:*

$$\lambda x r, \langle r, s\rangle, \mathsf{inl}\, r, \mathsf{inr}\, s, \mathsf{pack}\, r, \mathsf{in}_{k,j}\, r, \mathsf{CoIt}_k(\vec{m}, \vec{s}, r), \mathsf{CoRec}_k(\vec{m}, \vec{s}, r), \mathsf{out}_k^{-1}(\vec{m}, \vec{r})$$

Analogously E-terms are terms generated by an elimination rule, i.e. are terms of the following shapes:

$$rs, \pi_1 r, \pi_2 r, \mathsf{case}(r, x, s, y.t), \mathsf{open}(r, z.t), \mathsf{It}_k(\vec{m}, \vec{s}, r), \mathsf{Rec}_k(\vec{m}, \vec{s}, r), \mathsf{out}_{k,j}\, r$$

Lemma 4.6 (Generation Lemma) *If $\Gamma \vdash_{\mathbb{E}} t : A$, where A is an open formula then:*

- ○ *If t is the variable x then there exists a declaration $x : B \in \Gamma$ such that $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$.*

- ○ *If t is an I-term then $\Gamma \vdash_{\mathbb{E}} t : A$ is the conclusion of an instance of the rule generating t.*

- ○ *if t is an E-term then there exists a formula B such that $\Gamma \vdash_{\mathbb{E}} t : B$ is the conclusion of the rule generating t and $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$.*

Proof. Consider in the derivation $\Gamma \vdash_{\mathbb{E}} t : A$ the last step where a traceable rule $\mathcal{R}$ occurs, thus $\mathcal{R}$ is the rule generating t. Suppose that the conclusion of $\mathcal{R}$ is $\Gamma \vdash_{\mathbb{E}^\star} t : B$. The main lemma implies that $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$. Case Analysis on t.

- $t \equiv x$. Then $\mathcal{R}$ is (Var) and therefore exists $x : B \in \Gamma$ and as mentioned before $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B^\circ)$.

- t is an E-term. This case is immediate as $\mathcal{R}$ is the rule generating t.

- t is an I-term. Case analysis on the shape of t. We concentrate on $t \equiv \mathsf{in}_{k,j}\, r$. In this case $\mathcal{R}$ is (μI), $B \equiv \mu Y(\mathcal{D}_1, \dots, \mathcal{D}_l)\vec{\mathbb{c}}_j\vec{r}$ and $\Gamma \vdash r : \mathcal{G}_j[Y := \mu Y(\mathcal{D}_1, \dots, \mathcal{D}_l)]\vec{r}$. Clearly $B \equiv B^\circ$, therefore $A \in \mathcal{C}_{\Gamma,\mathbb{E}}(B)$. Let

  $$\begin{aligned} \mathcal{C} = \quad & \{(\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{\mathbb{c}}_j\vec{s}\,|\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{s}, \\ & \qquad \text{for some } k, \mathcal{C}_i, \vec{s}\,\}, \end{aligned}$$

  we need to show that $A \in \mathcal{C}$. We claim that $\mathcal{C}_{\Gamma,\mathbb{E}}(B) \subseteq \mathcal{C}$.

  $(I1)$ Obviously $B \in \mathcal{C}$.

  $(I2)$ Assume $R \in \mathcal{C}$ and $x \notin FV(\Gamma, \mathbb{E})$. We have

  $$\begin{aligned} R[x := t] \equiv \quad & (\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{\mathbb{c}}_j\vec{s})[x := t] = \\ & (\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)[x := t])\vec{\mathbb{c}}_j\vec{s}[x := t]. \end{aligned}$$

  $R \in \mathcal{C}$ implies $\Gamma \vdash r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{s}$. Next by $(Dsp2)$, as $x \notin FV(\mathbb{E}, \Gamma)$ we get

  $$\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)][x := t])\vec{s}[x := t].$$

  Using lemma 1.22 we obtain that

  $$\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[x := t][X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)[x := t]]\vec{s}[x := t],$$

  which is the same as

  $$\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[x := t][X := \mu X(\mathcal{C}_1[x := t], \dots, \mathcal{C}_k[x := t])\vec{s}[x := t].$$

  Therefore $R[x := t] \in \mathcal{C}$.

  $(I3)$ Assume $R \in \mathcal{C}$ and $Y \notin FV(\Gamma)$. We have

  $$\begin{aligned} R[Y := \mathcal{K}] \equiv \quad & \big(\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{\mathbb{c}}_j\vec{s}\big)[Y := \mathcal{K}] \equiv \\ & \mu X\Big(\mathcal{C}_1[Y := \mathcal{K}], \dots, \mathcal{C}_k[Y := \mathcal{K}]\Big)\vec{\mathbb{c}}_j\vec{s}. \end{aligned}$$

  $R \in \mathcal{C}$ implies $\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{s}$ which by $(Dsp3)$ as $Y \notin FV(\Gamma)$, implies

  $$\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)][Y := \mathcal{K}]\vec{s}.$$

Using lemma 1.22 we obtain

$$\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}_j[Y := \mathcal{K}][X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)[Y := \mathcal{K}])\vec{s},$$

i.e.

$$\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}_j[Y := \mathcal{K}][X := \mu X(\mathcal{C}_1[Y := \mathcal{K}], \dots, \mathcal{C}_k[Y := \mathcal{K}])]\vec{s}$$

Therefore $R[Y := \mathcal{K}] \in \mathcal{C}$.

$(I4)$ Assume $R[x := s] \in \mathcal{C}$ and $\Gamma \vdash_{\mathbb{E}} s = t$. We have $R[x := s] \equiv \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{\mathbb{c}_j}\vec{s}$ with $\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{s}$. As first order substitutions do not change the shape of the formulas we also have $R \equiv \mu X(\mathcal{C}'_1, \dots, \mathcal{C}'_k)\vec{\mathbb{c}_j}\vec{q}$ such that $\mathcal{C}_i = \langle \mathcal{F}'_i, \vec{\mathbb{c}_i} \rangle$ with $\mathcal{F}'_i[x := s] \equiv \mathcal{F}_i, \vec{q}[x := s] \equiv \vec{s}$. Therefore the above derivation can be rewritten as $\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}'_j[X := \mu X(\mathcal{C}'_1, \dots, \mathcal{C}'_k)]\vec{q}\ )[x := s]$, Now using (Eq) we get $\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}'_j[X := \mu X(\mathcal{C}'_1, \dots, \mathcal{C}'_k)]\vec{q}\ )[x := t]$ i.e. $\Gamma \vdash_{\mathbb{E}} r : (\mathcal{F}'_j[x := t][X := \mu X(\mathcal{C}'_1, \dots, \mathcal{C}'_k)[x := t]])\vec{q}\,[x := t]$ Therefore $R[x := t] \in \mathcal{C}$.

This concludes the proof of $\mathcal{C}_{\Gamma,\mathbb{E}}(B) \subseteq \mathcal{C}$. Therefore $A \in \mathcal{C}$.

Proposition 4.1 (One-step Subject Reduction) *If* $\Gamma \vdash t : A$ *and* $t \to^1_\beta \widehat{t}$ *(i.e.* $t \to_\beta \widehat{t}$ *in one step) then* $\Gamma \vdash \widehat{t} : A$.

Proof. Induction on $\vdash$.

- The case of (Var) is trivial as there is no redex.
- $(\to I)$. We have $A \equiv B \to C$, $\Gamma \vdash \lambda xr : B \to C$ from $\Gamma, x : B \vdash r : C$ and $\lambda xr \to_\beta \lambda x\widehat{r}$ with $r \to_\beta \widehat{r}$. By IH we get $\Gamma, x : B \vdash \widehat{r} : C$ which implies $\Gamma \vdash \lambda x\widehat{r} : B \to C$.
- $(\to E)$. We have $\Gamma \vdash rs : A$ from $\Gamma \vdash r : B \to A, \Gamma \vdash s : B$. The cases $rs \to r\widehat{s}$, $rs \to \widehat{r}s$ are immediate from IH. We analyze the case $r \equiv \lambda xq$ with $rs \equiv (\lambda x.q)s \to_\beta q[x := s]$. We have $\Gamma \vdash \lambda xq : B \to A$. As $B \to A$ is obviously open the generation lemma implies $\Gamma, x : B \vdash q : A$. Therefore as $\Gamma \vdash s : B$ $(Dsp1)$ yields $\Gamma \vdash q[x := s] : A$.
- $(\wedge I)$. IH.
- $(\wedge_1 E)$. We have $t \equiv \pi_1 r$. The interesting case $r \equiv \langle s, t \rangle$ and $\widehat{t} \equiv s$ is solved applying the generation lemma.
- $(\wedge_2 E)$. Analogous to the previous case.
- $(\vee_L I)$. IH.
- $(\vee_R I)$. IH.

- $(\vee E)$. We have $\Gamma \vdash \mathsf{case}(s, x.p, y.q) : A$ from $\Gamma \vdash s : B \vee C, \Gamma, x : B \vdash p : A, \Gamma, y : C \vdash q : A$. The interesting cases are $s \equiv \mathsf{inl}\, r, \mathsf{inr}\, r$. We analyze the case $s \equiv \mathsf{inl}\, r$ and $\widehat{t} \equiv p[x := r]$. From the assumption $\Gamma \vdash \mathsf{inl}\, r : B \vee C$ the generation lemma yields $\Gamma \vdash r : B$, this together with $\Gamma, x : B \vdash p : A$ yields by $(Dsp1)$, $\Gamma \vdash p[x := r] : A$, i.e., $\Gamma \vdash \widehat{t} : A$.

- $(\forall I)$. We have $\Gamma \vdash t : \forall x A$ from $\Gamma \vdash t : A$ with $x \notin FV(\Gamma)$. By IH we get $\Gamma \vdash \widehat{t} : A$ therefore by $(\forall I)$ we get $\Gamma \vdash \widehat{t} : \forall x A$.

- $(\forall E)$. We have $\Gamma \vdash t : A[x := s]$ from $\Gamma \vdash t : \forall x A$. By IH we get $\Gamma \vdash \widehat{t} : \forall x A$, therefore by $(\forall E)$ we conclude $\Gamma \vdash \widehat{t} : A[x := s]$.

- $(\forall^2 I)$. Analogous to $(\forall I)$.

- $(\forall^2 E)$. Analogous to $(\forall E)$.

- $(\restriction I)$. IH.

- $(\restriction E)$. IH.

- (Eq). IH.

- (μI). IH.

- (μE). $A \equiv \mathcal{K}\vec{t}$, $\Gamma \vdash \mathsf{It}_k(\vec{m}, \vec{s}, r) : \mathcal{K}\vec{t}$, from $\Gamma \vdash m_i : \mathcal{F}_i\, \mathsf{mon}\, X, \Gamma \vdash s_i : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathbb{c}}_i}$ and $\Gamma \vdash r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t}$. The only interesting case is $r \equiv \mathsf{in}_{k,i}\, q$, so that $\widehat{t} \equiv s_i\big(m_i(\lambda z.\mathsf{It}_k(\vec{m}, \vec{s}, z))q\big)$.
  It is easy to see that $\Gamma \vdash \lambda z.\mathsf{It}_k(\vec{m}, \vec{s}, z) : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq \mathcal{K}$, therefore $\Gamma \vdash m_i(\lambda z.\mathsf{It}_k(\vec{m}, \vec{s}, z)) : \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)] \subseteq \mathcal{F}_i[X := \mathcal{K}]$.
  On the other hand from $\Gamma \vdash \mathsf{in}_{k,i}\, q : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t}$ the generation lemma yields $\Gamma \vdash q : \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)]\vec{r}$ and $\vec{t} \equiv \vec{\mathbb{c}}_i\vec{r}$.
  Therefore we get $\Gamma \vdash m_i(\lambda z.\mathsf{It}_k(\vec{m}, \vec{s}, z))q : \mathcal{F}_i[X := \mathcal{K}]$ which implies $\Gamma \vdash s_i\big(m_i(\lambda z.\mathsf{It}_k(\vec{m}, \vec{s}, z))q\big) : \mathcal{K}^{\vec{\mathbb{c}}_i}\vec{r}$, i.e., $\Gamma \vdash \widehat{t} : A$.

- (μE^+). Similar to the previous case.

- (νI). IH.

- (νI^+). IH.

- (νI^i). IH.

- (νE). We have $A \equiv \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}}_i\vec{t}$ and $\Gamma \vdash \mathsf{out}_{k,j}\, s : A$ coming from $\Gamma \vdash s : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$.
  The interesting cases are $s \equiv \mathsf{CoIt}_k(\vec{m}, \vec{s}, r), \mathsf{CoRec}_k(\vec{m}, \vec{s}, r), \mathsf{out}^{-1}(\vec{m}, \vec{r})$. We analyze the case $s \equiv \mathsf{CoIt}_k(\vec{m}, \vec{s}, r)$ and $\widehat{t} = m_i\big(\lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z)\big)(s_i r)$. From the assumption $\Gamma \vdash \mathsf{CoIt}_k(\vec{m}, \vec{s}, r) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$, the generation lemma yields $\Gamma \vdash m_i : \mathcal{F}_i\, \mathsf{mon}\, X,\ \Gamma \vdash s_i : \mathcal{K} \subseteq \mathcal{F}_i[X := \mathcal{K}]^{\vec{\mathbb{c}}_i},\ \Gamma \vdash r : \mathcal{K}\vec{t}$. It is easy to see that $\Gamma \vdash \lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z) : \mathcal{K} \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$, which implies $\Gamma \vdash m_i\big(\lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z)\big) : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]$. On the other hand we have $\Gamma \vdash s_i r : \mathcal{F}_i[X := \mathcal{K}]\vec{\mathbb{c}}_i\vec{t}$. Therefore $\Gamma \vdash m_i\big(\lambda z.\mathsf{CoIt}_k(\vec{m}, \vec{s}, z)\big)(s_i r) : \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}}_i\vec{t}$, i.e. $\Gamma \vdash \widehat{t} : A$.

$\dashv$

Corollary 4.1 (Subject Reduction for $\mathsf{MCICD}^\star$) *If* $\Gamma \vdash_{\mathbb{E}} r : A$ *and* $r \to_\beta \widehat{r}$ *then* $\Gamma \vdash_{\mathbb{E}} \widehat{r} : A$.

Proof. Induction on the length of the reduction sequence $r \to_\beta \widehat{r}$. $\dashv$

4.2 The Realizability Interpretation

To define realizability we will use the following notation: given a n-ary predicate variable X, we denote with X^+ a $(n+1)$-ary predicate variable uniquely associated with X. We also set:

$$\begin{aligned} \mathbb{C}_i^k &:= \lambda x.\, \mathsf{in}_{k,i}\, x \\ \mathbb{D}_i^k &:= \lambda x.\, \mathsf{out}_{k,i}\, x \end{aligned}$$

Definition 4.6 *Given an* MCICT^-*-term* t *and an* MCICD*–formula* A *we define the* $\mathsf{MCICD}^\star$*–formula* $t \;\mathtt{r}\; A$ *as follows:*

$$\begin{array}{rcl} t \;\mathtt{r}\; X\vec{s} & := & X^+\vec{s}t \\ t \;\mathtt{r}\; A \to B & := & \forall z.z \;\mathtt{r}\; A \to tz \;\mathtt{r}\; B \\ t \;\mathtt{r}\; \forall x A & := & \forall x.t \;\mathtt{r}\; A \\ t \;\mathtt{r}\; \forall X A & := & \forall X^+.t \;\mathtt{r}\; A \\ t \;\mathtt{r}\; A \wedge B & := & (\pi_1 t \;\mathtt{r}\; A) \wedge (\pi_2 t \;\mathtt{r}\; B) \\ t \;\mathtt{r}\; A \vee B & := & \exists z.(z \;\mathtt{r}\; A{\restriction} t = \mathsf{inl}\, z) \vee (z \;\mathtt{r}\; B{\restriction} t = \mathsf{inr}\, z) \\ t \;\mathtt{r}\; \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{s} & := & \mu X^+(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}})\vec{s}t \\ t \;\mathtt{r}\; \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{s} & := & \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})\vec{s}t \end{array}$$

where for a comprehension predicate $\mathcal{F} := \lambda\vec{y}.F$ *we define*

$$\mathcal{F}^{\mathtt{r}} := \lambda\vec{y}, z.z \;\mathtt{r}\; F, \quad z \notin \vec{y} \cup FV(F),$$

and if $\mathcal{C}_i := \langle \mathcal{F}_i, \vec{\mathbb{c}_i} \rangle$, $\mathcal{D}_i := \langle \mathcal{G}_i, \vec{\mathbb{d}_i} \rangle$ *then*

$$\mathcal{C}_i^{\mathtt{r}} := \langle \mathcal{F}_i^{\mathtt{r}}, \vec{\mathbb{c}_i}, \mathbb{C}_i^k \rangle, \quad \mathcal{D}_i^{\mathtt{r}} := \langle \mathcal{G}_i^{\mathtt{r}}, \vec{\mathbb{d}_i}, \mathbb{D}_i^k \rangle.$$

Observe that the last tag in the clauses $\mathcal{C}_i^{\mathtt{r}}, \mathcal{D}_i^{\mathtt{r}}$ is a closed-term and that existential and restricted formulas are only needed to define realizability for disjunctions. For more interesting applications of restricted formulas see [Par92] and the appendix C of [Raf94]. As we can see the definition of realizability for a (co)inductive definition is again a (co)inductive definition naturally corresponding to the original definition. Therefore the definition of realizability for these cases is not reductive as in [Par92, Mir02].

Lemma 4.7 (Substitution Properties) *The following properties hold:*

(*i*) $(t \;\mathtt{r}\; A)[x := s] \equiv t[x := s] \;\mathtt{r}\; A[x := s]$.

(ii) $(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] \equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]$.

Proof. Induction on A. For part (i). Case $A \equiv \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{r}$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[x := s] &\equiv \big(\mu X^+(\mathcal{C}_1^{\mathtt{r}}, \dots, \mathcal{C}_k^{\mathtt{r}})\vec{r}t\big)[x := s]\\ &\equiv \mu X^+\big(\mathcal{C}_1^{\mathtt{r}}[x := s], \dots, \mathcal{C}_k^{\mathtt{r}}[x := s]\big)\vec{r}[x := s]t[x := s]\end{aligned}$$

$$\begin{aligned}t[x := s] \mathrel{\mathtt{r}} A[x := s] &\equiv t[x := s] \mathrel{\mathtt{r}} \big(\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{r}\big)[x := s]\\ &\equiv t[x := s] \mathrel{\mathtt{r}} \mu X\big(\mathcal{C}_1[x := s], \dots, \mathcal{C}_k[x := s]\big)\vec{r}[x := s]\\ &\equiv \mu X^+\big(\mathcal{C}_1[x := s]^{\mathtt{r}}, \dots, \mathcal{C}_k[x := s]^{\mathtt{r}}\big)\vec{r}[x := s]t[x := s]\end{aligned}$$

By IH we get $\mathcal{C}_i[x := s]^{\mathtt{r}} = \mathcal{C}_i^{\mathtt{r}}[x := s]$, therefore the equality holds.

We prove part (ii) in detail.

Case $A \equiv X\vec{r}$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv (X^+\vec{r}t)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\equiv \mathcal{F}^{\mathtt{r}}\vec{r}t\\ &\equiv t \mathrel{\mathtt{r}} \mathcal{F}\vec{r}\\ &\equiv t \mathrel{\mathtt{r}} (X\vec{r})[X := \mathcal{F}^{\mathtt{r}}]\\ &\equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]\end{aligned}$$

Case $A \equiv Y\vec{r}$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv (Y^+\vec{r}t)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\equiv Y^+\vec{r}t\\ &\equiv t \mathrel{\mathtt{r}} Y\vec{r}\\ &\equiv t \mathrel{\mathtt{r}} (Y\vec{r})[X := \mathcal{F}]\\ &\equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]\end{aligned}$$

Case $A \equiv B \to C$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv (\forall z.z \mathrel{\mathtt{r}} B \to tz \mathrel{\mathtt{r}} C)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\equiv \forall z.(z \mathrel{\mathtt{r}} B)[X^+ := \mathcal{F}^{\mathtt{r}}] \to (tz \mathrel{\mathtt{r}} C)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\underset{IH}{\equiv} \forall z.z \mathrel{\mathtt{r}} B[X := \mathcal{F}] \to tz \mathrel{\mathtt{r}} C[X := \mathcal{F}]\\ &\equiv t \mathrel{\mathtt{r}} (B[X := \mathcal{F}] \to C[X := \mathcal{F}])\\ &\equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]\end{aligned}$$

Case $A \equiv \forall Y B$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv \big(\forall Y^+.t \mathrel{\mathtt{r}} B\big)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\equiv \forall Y^+.(t \mathrel{\mathtt{r}} B)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\underset{IH}{\equiv} \forall Y^+.t \mathrel{\mathtt{r}} B[X := \mathcal{F}]\\ &\equiv t \mathrel{\mathtt{r}} \forall Y.B[X := \mathcal{F}]\\ &\equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]\end{aligned}$$

Case $A \equiv B \vee C$.

$$\begin{aligned}&(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] \equiv \big(\exists z.(z \mathrel{\mathtt{r}} B{\upharpoonright} t = \mathsf{inl}\, z) \vee (z \mathrel{\mathtt{r}} C{\upharpoonright} t = \mathsf{inr}\, z)\big)[X^+ := \mathcal{F}^{\mathtt{r}}]\\ &\equiv \exists z.\big((z \mathrel{\mathtt{r}} B)[X^+ := \mathcal{F}^{\mathtt{r}}]{\upharpoonright} t = \mathsf{inl}\, z\big) \vee \big((z \mathrel{\mathtt{r}} C)[X^+ := \mathcal{F}^{\mathtt{r}}]{\upharpoonright} t = \mathsf{inr}\, z\big)\\ &\underset{IH}{\equiv} \exists z.\big(z \mathrel{\mathtt{r}} B[X := \mathcal{F}]{\upharpoonright} t = \mathsf{inl}\, z\big) \vee \big(z \mathrel{\mathtt{r}} C[X := \mathcal{F}]{\upharpoonright} t = \mathsf{inr}\, z\big)\\ &\equiv t \mathrel{\mathtt{r}} \big(B[X := \mathcal{F}] \vee C[X := \mathcal{F}]\big) \equiv t \mathrel{\mathtt{r}} A[X := \mathcal{F}]\end{aligned}$$

Case $A \equiv \nu Y(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{r}$.

$$\begin{aligned}(t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv \big(\nu Y^+(\mathcal{D}_1^{\mathtt{r}}, \dots, \mathcal{D}_k^{\mathtt{r}})\vec{r}\,t\big)[X^+ := \mathcal{F}^{\mathtt{r}}] \\ &\equiv \nu Y^+(\mathcal{D}_1^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}], \dots, \mathcal{D}_k^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}])\vec{r}\,t\end{aligned}$$

Now if $\mathcal{D}_i \equiv \langle \mathcal{G}_i, \vec{\mathbb{c}_i} \rangle$ with $\mathcal{G}_i \equiv \lambda\vec{y}G_i$ then $\mathcal{D}_i^{\mathtt{r}} \equiv \langle \mathcal{G}_i^{\mathtt{r}}, \vec{\mathbb{d}}_i, \mathbb{D}_i^k \rangle$. Therefore $\mathcal{D}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}] \equiv \langle \mathcal{G}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}], \vec{\mathbb{d}}_i, \mathbb{D}_i^k \rangle$ and observe that

$$\begin{aligned}\mathcal{G}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}] &\equiv (\lambda\vec{y}, z.z \mathrel{\mathtt{r}} G_i)[X^+ := \mathcal{F}^{\mathtt{r}}] \\ &\equiv \lambda\vec{y}, z.(z \mathrel{\mathtt{r}} G_i)[X^+ := \mathcal{F}^{\mathtt{r}}] \\ &\underset{IH}{\equiv} \lambda\vec{y}, z.z \mathrel{\mathtt{r}} G_i[X := \mathcal{F}] \\ &\equiv (\lambda\vec{y}.G_i[X := \mathcal{F}])^{\mathtt{r}} \\ &\equiv \mathcal{G}_i[X := \mathcal{F}]^{\mathtt{r}}\end{aligned}$$

Therefore we have

$$\begin{aligned}\mathcal{D}_i[X := \mathcal{F}]^{\mathtt{r}} &\equiv \langle \mathcal{G}_i[X := \mathcal{F}], \vec{\mathbb{d}}_i \rangle^{\mathtt{r}} \\ &\equiv \langle \mathcal{G}_i[X := \mathcal{F}]^{\mathtt{r}}, \vec{\mathbb{d}}_i, \mathbb{D}_i^k \rangle \\ &\equiv \langle \mathcal{G}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}], \vec{\mathbb{d}}_i, \mathbb{D}_i^k \rangle \\ &\equiv \mathcal{D}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}]\end{aligned}$$

and

$$\begin{aligned}t \mathrel{\mathtt{r}} A[X := \mathcal{F}] &\equiv t \mathrel{\mathtt{r}} \nu Y(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{r}[X := \mathcal{F}] \\ &\equiv t \mathrel{\mathtt{r}} \nu Y(\mathcal{D}_i[X := \mathcal{F}], \dots, \mathcal{D}_k[X := \mathcal{F}])\vec{r} \\ &\equiv \nu Y^+(\mathcal{D}_1[X := \mathcal{F}]^{\mathtt{r}}, \dots, \mathcal{D}_k[X := \mathcal{F}]^{\mathtt{r}})\vec{r}\,t \\ &\equiv \nu Y^+(\mathcal{D}_1^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}], \dots, \mathcal{D}_i^{\mathtt{r}}[X^+ := \mathcal{F}^{\mathtt{r}}])\vec{r}\,t \\ &\equiv (t \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}]\end{aligned}$$

⊣

4.2.1 Realizing the Axioms

In this section we look for realizers of the closure and induction axioms.

Proposition 4.2 *Given an inductive predicate $\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)$ we have:*

$$\vdash \lambda x.\, \mathsf{in}_{k,j}\, x : \mathbb{C}_j^k \mathrel{\mathtt{r}} \mathsf{Cl}_{\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k), j}$$

Proof. $\mathbb{C}_j^k \mathrel{\mathtt{r}} \mathsf{Cl}_{\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k), j}$ unfolds to:

$$\forall\vec{y}\,\forall z.z \mathrel{\mathtt{r}} \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{y} \to \mathbb{C}_j^k z \mathrel{\mathtt{r}} (\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k))\vec{\mathbb{c}_j}\vec{y}$$

which by lemma 4.7 and definition of realizability for inductive predicates is the same as:

$$\forall\vec{y}\,\forall z.\mathcal{F}_j^{\mathtt{r}}[X^+ := \big(\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\big)^{\mathtt{r}}]\vec{y}z \to \mu X^+(\mathcal{C}_1^{\mathtt{r}}, \dots, \mathcal{C}_k^{\mathtt{r}})(\vec{\mathbb{c}_j}\vec{y}\,)(\mathbb{C}_j^k z)$$

which equals

$$\forall\vec{y}\,\forall z.\mathcal{F}_j^{\mathtt{r}}[X^+ := \mu X^+(\mathcal{C}_1^{\mathtt{r}},\dots,\mathcal{C}_k^{\mathtt{r}})]\vec{y}z \to \mu X^+(\mathcal{C}_1^{\mathtt{r}},\dots,\mathcal{C}_k^{\mathtt{r}})(\vec{\mathbb{c}_j}\vec{y}\,)(\mathbb{C}_j^k z)$$

This proves that

$$\mathbb{C}_j^k\ \mathtt{r}\ \mathsf{Cl}_{\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k),j} \equiv \mathsf{Cl}_{\mu X^+(\mathcal{C}_1^{\mathtt{r}},\dots,\mathcal{C}_k^{\mathtt{r}}),j}$$

Therefore the claim follows from proposition 3.1.

$\dashv$

Proposition 4.3 *Given a coinductive predicate $\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)$ we have:*

$$\vdash \lambda x.\,\mathsf{out}_{k,j}\,x : \mathbb{D}_j^k\ \mathtt{r}\ \mathsf{CoCl}_{\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k),j}$$

Proof. Analogous to proposition 4.2. $\dashv$

Proposition 4.4 *If $\mathcal{C}_i = \langle\mathcal{F}_i,\vec{\mathbb{c}_i}\rangle$, $\Theta \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^+$ for $1 \le i \le k$ and*

$$\mathbb{J} := \lambda\vec{x}\lambda\vec{y}\,\lambda z.\mathsf{It}_k(\vec{x},\vec{y}\,,z), \mathbb{R} := \lambda\vec{x}\lambda\vec{y}\,\lambda z.\mathsf{Rec}_k(\vec{x},\vec{y}\,,z)$$

then

(i). $\Theta \vdash \lambda\vec{x}.\lambda\vec{y}\,.\lambda z.\mathsf{It}_k(\vec{\mathsf{m}},\vec{\mathsf{s}},z) : \mathbb{J}\ \mathtt{r}\ \mathsf{Ind}_{\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)}$

(ii). $\Theta \vdash \lambda\vec{x}.\lambda\vec{y}\,.\lambda z.\mathsf{Rec}_k(\vec{\mathsf{m}},\vec{\mathsf{s}},z) : \mathbb{R}\ \mathtt{r}\ \mathsf{Ind}^+_{\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)}$

where $\mathsf{s_i} := \lambda u_i.y_i(x_i(\lambda v.v)u_i)$ $(1 \le i \le k)$ ($\mathsf{s_i}$ is some kind of η-expansion of y_i).

- Proof of part (i).

Set $\mu := \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)$ and $\mu^{\mathtt{r}} := \mu X^+(\mathcal{C}_1^{\mathtt{r}},\dots,\mathcal{C}_k^{\mathtt{r}})$. We want to show:

$$\begin{array}{ll}\Theta \vdash_{\mathsf{MCICD}^\star} \mathbb{J}\ \mathtt{r}\ \forall Z. & \dots,\mathcal{F}_i\,\mathsf{mon}\,X,\dots_{(1\le i\le k)} \to \\ & \dots,\mathcal{F}_i[X := Z] \subseteq Z^{\vec{\mathbb{c}_i}},\dots_{(1\le i\le k)} \to \\ & \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k) \subseteq Z\end{array}$$

which unfolds to

$$\begin{array}{ll}\forall Z^+.\forall\vec{m}. & \dots,m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X,\dots_{(1\le i\le k)} \to \\ & \forall\vec{f}.\ \dots,f_i\ \mathtt{r}\ \mathcal{F}_i[X := Z] \subseteq Z^{\vec{\mathbb{c}_i}},\dots_{(1\le i\le k)} \to \\ & \mathbb{J}\vec{m}\vec{f}\ \mathtt{r}\ \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k) \subseteq Z\end{array} \tag{4.1}$$

Assume

$$x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X \quad (1 \le i \le k) \tag{4.2}$$

and $y_i : f_i\ \mathtt{r}\ \mathcal{F}_i[X := Z] \subseteq Z^{\vec{\mathbb{c}_i}}$, that is

$$y_i : \forall\vec{v}.\forall u.u\ \mathtt{r}\ \mathcal{F}_i[X := Z]\vec{v} \to f_i u\ \mathtt{r}\ Z(\vec{\mathbb{c}_i}\vec{v}) \quad (1 \le i \le k) \tag{4.3}$$

we need to show $\mathbb{J}\vec{m}\vec{f}\ \mathtt{r}\ \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)\subseteq Z$, i.e.

$$\forall\vec{v}.\forall w.w\ \mathtt{r}\ (\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k))\vec{v}\rightarrow \mathbb{J}\vec{m}\vec{f}w\ \mathtt{r}\ Z\vec{v}$$

Assume

$$z : w\ \mathtt{r}\ (\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k))\vec{v}\equiv(\mu X^{+}(\mathcal{C}_1^{\mathtt{r}},\ldots,\mathcal{C}_k^{\mathtt{r}}))\vec{v}w, \tag{4.4}$$

and let

$$\mathcal{Q}:=\lambda\vec{x},z.\mathbb{J}\vec{m}\vec{f}z\ \mathtt{r}\ Z\vec{x},$$

Set

$$\begin{array}{rcl}\Gamma & := & \mathbb{\Theta}, x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X(1\le i\le k),\\ & & y_i : f_i\ \mathtt{r}\ \mathcal{F}_i[X:=Z]\subseteq Z^{\vec{\mathbb{c}_i}}\quad(1\le i\le k),\\ & & z : (\mu X^{+}(\mathcal{C}_1^{\mathtt{r}},\ldots,\mathcal{C}_k^{\mathtt{r}}))\vec{v}w\end{array}$$

We need to prove $\Gamma\vdash\mathcal{Q}\vec{v}w$.

Obviously $\Gamma\vdash\mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^{+}$ and $\Gamma\vdash z : (\mu X^{+}(\mathcal{C}_1^{\mathtt{r}},\ldots,\mathcal{C}_k^{\mathtt{r}}))\vec{v}w$, therefore using the elimination rule (μE) it suffices to show

$$\Gamma\vdash\mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\subseteq\mathcal{Q}^{\vec{\mathbb{c}_i},\mathbb{C}_i^k},\quad(1\le i\le k)$$

that is

$$\forall\vec{x}.\forall z.\mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\vec{x}z\rightarrow\mathcal{Q}(\vec{\mathbb{c}_i}\vec{x})(\mathbb{C}_i^k z)$$

Assume

$$u_i : \mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\vec{x}z, \tag{4.5}$$

and set $\Pi:=\Gamma,u_i:\mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\vec{x}z$. We need to prove

$$\Pi\vdash\mathcal{Q}(\vec{\mathbb{c}_i}\vec{x})(\mathbb{C}_i^k z) \tag{4.6}$$

The assumptions (4.2) unfold to:

$$\begin{array}{c}\forall X^{+}\forall Y^{+}\forall z.(\forall\vec{y}\,\forall w.w\ \mathtt{r}\ X\vec{y}\ \rightarrow zw\ \mathtt{r}\ Y\vec{y}\ )\qquad\rightarrow\\ (\forall\vec{y}\,\forall u.u\ \mathtt{r}\ \mathcal{F}_i\vec{y}\ \rightarrow m_i zu\ \mathtt{r}\ \mathcal{F}_i[X:=Y]\vec{y}\ )\end{array} \tag{4.7}$$

Next we instantiate the predicate variables $X^{+}:=\mathcal{Q},Y^{+}:=Z^{\mathtt{r}}$, to obtain:

$$\begin{array}{rl}x_i : & \forall z.(\forall\vec{y}\ \ \forall w.\mathcal{Q}\vec{y}\ \ w\rightarrow Z^{\mathtt{r}}\vec{y}\ \ (zw))\rightarrow\\ & \quad(\forall\vec{y}\ \ \forall u.\mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\vec{y}\ \ u\rightarrow m_i zu\ \mathtt{r}\ \mathcal{F}_i[X:=Z]\vec{y}\ \ )\end{array}$$

Next we substitute $z:=\mathbb{J}\vec{m}\vec{f}$:

$$\begin{array}{rl}x_i : & (\forall\vec{y}\ \ \forall w.\mathcal{Q}\vec{y}\ \ w\rightarrow Z^{\mathtt{r}}\vec{y}\ \ ((\mathbb{J}\vec{m}\vec{f})w))\rightarrow\\ & \quad(\forall\vec{y}\ \ \forall u.\mathcal{F}_i^{\mathtt{r}}[X^{+}:=\mathcal{Q}]\vec{y}\ \ u\rightarrow m_i(\mathbb{J}\vec{m}\vec{f})u\ \mathtt{r}\ \mathcal{F}_i[X:=Z]\vec{y}\ \ )\end{array}$$

Observing that $Z^{\mathtt{r}}\vec{y}\ \ (\mathbb{J}\vec{m}\vec{f}w) \equiv \mathbb{J}\vec{m}\vec{f}w \ \mathtt{r}\ Z\vec{y}$ we see that the antecedent of this implication is of the form $\forall\vec{y}\ \ \forall w.A \to A$, therefore we can eliminate the implication and obtain:

$$\Pi \vdash x_i(\lambda v.v) : \forall\vec{y}\,\forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]\vec{y}\;u \to m_i(\mathbb{J}\vec{m}\vec{f})u \ \mathtt{r}\ \mathcal{F}_i[X := Z]\vec{y}$$

Instantiating $\vec{y}\ , u := \vec{x}, z$ and using assumption (4.5) we get

$$\Pi \vdash x_i(\lambda v.v)u_i : m_i(\mathbb{J}\vec{m}\vec{f})z \ \mathtt{r}\ \mathcal{F}_i[X := Z]\vec{x}$$

On the other hand, from assumption (4.3), with $\vec{v}, u := \vec{x}, m_i(\mathbb{J}\vec{m}\vec{f})z$ we get:

$$\Pi \vdash y_i : m_i(\mathbb{J}\vec{m}\vec{f})z \ \mathtt{r}\ \mathcal{F}_i[X := Z]\vec{x} \to f_i(m_i(\mathbb{J}\vec{m}\vec{f})z) \ \mathtt{r}\ Z(\vec{\mathbb{c}_i}\vec{x})$$

Therefore

$$\Pi \vdash y_i(x_i(\lambda v.v)u_i) : f_i(m_i(\mathbb{J}\vec{m}\vec{f})z) \ \mathtt{r}\ Z(\vec{\mathbb{c}_i}\vec{x})$$

But $\mathbb{E}_\beta \vdash f_i(m_i(\mathbb{J}\vec{m}\vec{f})z) = \mathbb{J}\vec{m}\vec{f}(\mathbb{C}_i^k z)$. Hence, by (Eq)

$$\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathbb{J}\vec{m}\vec{f}(\mathbb{C}_i^k z) \ \mathtt{r}\ Z(\vec{\mathbb{c}_i}\vec{x}),$$

That is $\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathcal{Q}(\vec{\mathbb{c}_i}\vec{x})(\mathbb{C}_i^k z)$ and the goal (4.6) is proved. Therefore $\Gamma \vdash \lambda u_i.y_i(x_i(\lambda v.v)u_i) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}] \subseteq \mathcal{Q}^{\vec{\mathbb{c}_i}, \mathbb{C}_i^k}$, which by (μE^+) yields:

$$\Gamma \vdash \mathsf{It}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, z) : \mathcal{Q}vw$$

Finally discharging the assumptions $\vec{x}, \vec{y}, z$, we get:

$$\Theta \vdash_{\mathsf{MCICD}^\star} \lambda\vec{x}.\lambda\vec{y}\ .\lambda z.\mathsf{It}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, z) : \mathbb{J}\ \mathtt{r}\ \mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$$

- Proof of part (ii).
  Set $\mu := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ and $\mu^{\mathtt{r}} := \mu X^+(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}})$, we want to show:

$$\begin{aligned}\Theta \vdash_{\mathsf{MCICD}^\star} \mathbb{R}\ \mathtt{r}\ \forall Z. \quad & \ldots, \mathcal{F}_i \,\mathsf{mon}\, X, \ldots_{(1\le i\le k)} \to \\ & \ldots, \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\vec{\mathbb{c}_i}}, \ldots_{(1\le i\le k)} \to \\ & \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z\end{aligned}$$

which unfolds to

$$\begin{aligned}\forall Z^+.\forall\vec{m}. \quad & \ldots, m_i \ \mathtt{r}\ \mathcal{F}_i \,\mathsf{mon}\, X, \ldots_{(1\le i\le k)} \to \\ & \forall\vec{f}.\ \ldots, f_i \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\vec{\mathbb{c}_i}}, \ldots_{(1\le i\le k)} \to \\ & \mathbb{R}\vec{m}\vec{f}\ \mathtt{r}\ \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z\end{aligned} \tag{4.8}$$

Assume for $1 \le i \le k$

$$x_i : m_i \ \mathtt{r}\ \mathcal{F}_i \,\mathsf{mon}\, X \tag{4.9}$$

and $y_i : f_i \mathtt{r}\, \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\vec{\mathbb{c}_i}}$, that is

$$y_i : \forall \vec{v}.\forall u.u\, \mathtt{r}\, \mathcal{F}_i[X := \mu \wedge Z]\vec{v} \to f_i u\, \mathtt{r}\, Z(\vec{\mathbb{c}_i}\vec{v}). \tag{4.10}$$

We need to show $\mathbb{R}\vec{m}\vec{f}\, \mathtt{r}\, \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z$, i.e.

$$\forall \vec{v}.\forall w.w\, \mathtt{r}\, (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\vec{v} \to \mathbb{R}\vec{m}\vec{f}w\, \mathtt{r}\, Z\vec{v}$$

Assume

$$z : w\, \mathtt{r}\, (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\vec{v} \equiv (\mu X^+(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\vec{v}w, \tag{4.11}$$

and let

$$\mathcal{Q} := \lambda\vec{x}, z.\mathbb{R}\vec{m}\vec{f}z\, \mathtt{r}\, Z\vec{x},$$

Set

$$\begin{array}{rcl} \Gamma & := & \mathbb{\Theta}, x_i : m_i\, \mathtt{r}\, \mathcal{F}_i\, \mathsf{mon}\, X\ (1 \leq i \leq k), \\ & & y_i : f_i\, \mathtt{r}\, \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\vec{\mathbb{c}_i}}\ (1 \leq i \leq k), \\ & & z : (\mu X^+(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\vec{v}w \end{array}$$

We need to prove $\Gamma \vdash \mathcal{Q}\vec{v}w$.

Obviously $\Gamma \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\, \mathsf{mon}\, X^+$ and $\Gamma \vdash z : (\mu X^+(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\vec{v}w$, therefore using the elimination rule (μE^+) it suffices to show

$$\Gamma \vdash \mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}] \subseteq \mathcal{Q}^{\vec{\mathbb{c}_i}, \mathbb{C}_i^k},\ (1 \leq i \leq k)$$

that is

$$\forall \vec{x}.\forall z.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{x}z \to \mathcal{Q}(\vec{\mathbb{c}_i}\vec{x})(\mathbb{C}_i^k z).$$

Assume

$$u_i : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{x}z \quad (1 \leq i \leq k) \tag{4.12}$$

and set $\Pi := \Gamma, u_i : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{x}z$. We need to prove

$$\Pi \vdash \mathcal{Q}(\vec{\mathbb{c}_i}\vec{x})(\mathbb{C}_i^k z) \quad (1 \leq i \leq k) \tag{4.13}$$

The assumptions (4.9) unfold to:

$$\begin{array}{ll} x_i : \forall X^+ \forall Y^+ \forall z. & (\forall \vec{y}\ \forall w.w\, \mathtt{r}\, X\vec{y}\ \to zw\, \mathtt{r}\, Y\vec{y}\ ) \to \\ & (\forall \vec{y}\ \forall u.u\, \mathtt{r}\, \mathcal{F}_i\vec{y}\ \to m_i zu\, \mathtt{r}\, \mathcal{F}_i[X := Y]\vec{y}\ ) \end{array} \tag{4.14}$$

Next we instantiate the predicate variables $X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}, Y^+ := (\mu \wedge Z)^{\mathtt{r}}$ to obtain:

$$\begin{array}{ll} x_i : & \forall z.(\forall \vec{y}\, \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\vec{y}\, w \to (\mu \wedge Z)^{\mathtt{r}}\vec{y}\, (zw)) \to \\ & \quad (\forall \vec{y}\, \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{y}\, u \to m_i zu\, \mathtt{r}\, \mathcal{F}_i[X := \mu \wedge Z]\vec{y}\ ) \end{array}$$

Instantiate now $z := \lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle$:

$$\begin{aligned} x_i: \quad & (\forall \vec{y}\, \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\vec{y}\, w \to (\mu \wedge Z)^{\mathtt{r}}\vec{y}\, ((\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)w)) \to \\ & \quad (\forall \vec{y}\, \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{y}\, u \to m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)u \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\vec{y}\ ) \end{aligned}$$

Observing that $(\mathcal{F} \wedge \mathcal{G})^{\mathtt{r}} \equiv \lambda\vec{z}, u.\mathcal{F}^{\mathtt{r}}\vec{z}(\pi_1 u) \wedge \mathcal{G}^{\mathtt{r}}\vec{z}(\pi_2 u)$ and

$$\begin{aligned} \mathbb{E}_\beta \vdash \quad & \pi_1\big((\lambda x.\langle x, \pi_2\vec{m}\vec{f}x\rangle)w\big) = w \\ \mathbb{E}_\beta \vdash \quad & \pi_2\big((\lambda x.\langle x, \pi_2\vec{m}\vec{f}x\rangle)w\big) = \pi_2\vec{m}\vec{f}w \end{aligned}$$

using (Eq) it is easy to see that

$$\vdash \lambda v.v : \forall \vec{y}\, \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\vec{y}\, w \to (\mu \wedge Z)^{\mathtt{r}}\vec{y}\, ((\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)w)$$

therefore we can eliminate the implication and obtain

$$\begin{aligned} \Pi \vdash x_i(\lambda v.v) : \forall \vec{y}\, \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\vec{y}\, u \to & \\ & m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)u \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\vec{y} \end{aligned}$$

Instantiating $\vec{y}\ , u := \vec{x}, z$ and using assumption (4.12) we get

$$\Pi \vdash x_i(\lambda v.v)u_i : m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\vec{x}.$$

On the other hand, from assumptions (4.10), with $\vec{v}, u := \vec{x}, m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z$ we get:

$$\Pi \vdash y_i : m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z \ \mathtt{r}\ \mathcal{F}_i[X := \mu{\wedge}Z]\vec{x} \to f_i(m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z) \ \mathtt{r}\ Z(\vec{\mathbb{c}}_i\vec{x}).$$

Therefore

$$\Pi \vdash y_i(x_i(\lambda v.v)u_i) : f_i(m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z) \ \mathtt{r}\ Z(\vec{\mathbb{c}}_i\vec{x}).$$

But it is easy to see that $\mathbb{E}_\beta \vdash f_i(m_i(\lambda x.\langle x, \mathbb{R}\vec{m}\vec{f}x\rangle)z) = \mathbb{R}\vec{m}\vec{f}(\mathbb{C}_i^k z)$, hence using (Eq) we get:

$$\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathbb{R}\vec{m}\vec{f}(\mathbb{C}_i^k z) \ \mathtt{r}\ Z(\vec{\mathbb{c}}_i\vec{x}),$$

That is $\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathcal{Q}(\vec{\mathbb{c}}_i\vec{x})(\mathbb{C}_i^k z)$ and the goal (4.13) is proved. Therefore $\Gamma \vdash \lambda u_i.y_i(x_i(\lambda v.v)u_i) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}] \subseteq \mathcal{Q}^{\vec{\mathbb{c}}_i, \mathbb{C}_i^k}$, which by (μE^+) yields:

$$\Gamma \vdash \mathsf{Rec}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, z) : \mathcal{Q}vw$$

Finally, discharging the assumptions $\vec{x}, \vec{y}, z$, we get:

$$\emptyset \vdash_{\mathsf{MCICD}^\star} \lambda\vec{x}.\lambda\vec{y}\ .\lambda z.\mathsf{Rec}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, z) : \mathbb{R}\ \mathtt{r}\ \mathsf{Ind}^+_{\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)}$$

⊣

Proposition 4.5 *If* $\mathcal{D}_i = \langle \mathcal{F}_i, \vec{\mathbb{c}_i} \rangle$, $\Theta \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^+$ *for* $1 \leq i \leq k$, *and*

$$\mathbb{K} := \lambda\vec{x}\lambda\vec{y}\lambda z.\mathsf{CoIt}_k(\vec{x}, \vec{y}, z),\ \mathbb{Q} := \lambda\vec{x}\lambda\vec{y}\lambda z.\mathsf{CoRec}_k(\vec{x}, \vec{y}, z)$$

then

(i). $\Theta \vdash \lambda\vec{x}\lambda\vec{y}\lambda z.\mathsf{CoIt}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, \mathsf{pack}\,z) : \mathbb{K}\ \mathtt{r}\ \mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

(ii). $\Theta \vdash \lambda\vec{x}\lambda\vec{y}\lambda z.\mathsf{CoRec}_k(\vec{\mathsf{m}}, \vec{\mathsf{q}}, \mathsf{pack}\,z) : \mathbb{Q}\ \mathtt{r}\ \mathsf{CoInd}^+_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

where for $1 \leq i \leq k$, *we set:*

$$\begin{aligned} \mathsf{s_i} &:= \lambda v.\mathsf{open}(v, w.x_i(\lambda u.\,\mathsf{pack}\,u)(y_i w)), \\ \mathsf{q_i} &:= \lambda v.\mathsf{open}\Big(v, w.x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big)\Big)(y_i w)\Big) \end{aligned}$$

Proof. Set $\nu := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ and $\nu^{\mathtt{r}} := \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})$. For the first part we want to show:

$$\begin{aligned} \Theta \vdash \mathbb{K}\ \mathtt{r}\ \forall Z.\quad & \ldots, \mathcal{F}_i\,\mathsf{mon}\,X, \ldots_{(1\leq i\leq k)} \to \\ & \ldots, Z \subseteq \mathcal{F}_i[X := Z]^{\vec{\mathbb{c}_i}}, \ldots_{(1\leq i\leq k)} \to \\ & Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \end{aligned}$$

which unfolds to

$$\begin{aligned} \forall Z^+.\forall\vec{m}.\quad & \ldots, m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X, , \ldots_{(1\leq i\leq k)} \to \\ & \forall\vec{f}.\ \ldots, f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := Z]^{\vec{\mathbb{c}_i}}, \ldots_{(1\leq i\leq k)} \to \\ & \mathbb{K}\vec{m}\vec{f}\ \mathtt{r}\ Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \end{aligned} \tag{4.15}$$

Assume

$$x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X \quad (1 \leq i \leq k) \tag{4.16}$$

and $y_i : f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := Z]^{\vec{\mathbb{c}_i}}$, that is

$$y_i : \forall\vec{v}.\forall u.u\ \mathtt{r}\ Z\vec{v} \to f_i u\ \mathtt{r}\ \mathcal{F}_i[X := Z](\vec{\mathbb{c}_i}\vec{v}) \quad (1 \leq i \leq k) \tag{4.17}$$

we need to show $\mathbb{K}\vec{m}\vec{f}\ \mathtt{r}\ Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$, i.e.

$$\forall\vec{v}.\forall w.w\ \mathtt{r}\ Z\vec{v} \to \mathbb{K}\vec{m}\vec{f}w\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{v}$$

Assume

$$z : w\ \mathtt{r}\ Z\vec{v} \equiv Z^+\vec{v}w, \tag{4.18}$$

We will prove $\mathbb{K}\vec{m}\vec{f}w\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{v} \equiv \nu^{\mathtt{r}}\vec{v}(\mathbb{K}\vec{m}\vec{f}w)$ via the (νI) rule with the following predicate:

$$\mathcal{Q} := \lambda\vec{v}, y.\exists u.u\ \mathtt{r}\ Z\vec{v} \upharpoonright y = \mathbb{K}\vec{m}\vec{f}u,$$

Set

$$\begin{aligned} \Gamma := \quad & \Theta, x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X, \quad (1 \leq i \leq k) \\ & y_i : f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := Z]^{\vec{\mathbb{c}_i}}, \quad (1 \leq i \leq k) \\ & z : w\ \mathtt{r}\ Z\vec{v} \end{aligned}$$

We need to prove $\Gamma \vdash \nu^{\mathtt{r}} \vec{v}(\mathbb{K}\vec{m}\vec{f}w)$.

Clearly $\Gamma \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^+$ and easily we can derive

$$\Gamma \vdash \mathsf{pack}\,z : \mathcal{Q}\vec{v}(\mathbb{K}\vec{m}\vec{f}w),$$

therefore using the introduction rule (νI) it suffices to show

$$\Gamma \vdash \mathcal{Q} \subseteq \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]^{\vec{\mathbb{c}_i},\mathbb{D}_i^k}, \quad (1 \leq i \leq k)$$

that is

$$\forall \vec{x}.\forall z.\mathcal{Q}\vec{x}z \to \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z)$$

Assume

$$v : \mathcal{Q}\vec{x}z \tag{4.19}$$

and set $\Pi := \Gamma, v : \mathcal{Q}\vec{x}z$. We need to prove

$$\Pi \vdash \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z) \tag{4.20}$$

The assumptions (4.16) unfold to:

$$\begin{aligned} x_i : \quad & \forall X^+ \forall Y^+ \forall z.(\forall \vec{y} \forall v.v \;\mathtt{r}\; X\vec{y} \to zv \;\mathtt{r}\; Y\vec{y}) \qquad \to \\ & \qquad (\forall \vec{y} \forall v.v \;\mathtt{r}\; \mathcal{F}_i\vec{y} \to m_i zv \;\mathtt{r}\; \mathcal{F}_i[X := Y]\vec{y}). \end{aligned} \tag{4.21}$$

We instantiate the predicate variables $X^+ := Z^+, Y^+ := \mathcal{Q}$ to obtain:

$$\begin{aligned} x_i : \quad & \forall z.\Big(\forall \vec{y} \forall v.v \;\mathtt{r}\; Z\vec{y} \to \mathcal{Q}\vec{y}(zv)\Big) \to \\ & \quad \Big(\forall \vec{y} \forall v.v \;\mathtt{r}\; \mathcal{F}_i[X := Z]\vec{y} \to \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]\vec{y}(m_i zv)\Big). \end{aligned}$$

Next we substitute $z := \mathbb{K}\vec{m}\vec{f}$:

$$\begin{aligned} x_i : \quad & \Big(\forall \vec{y} \forall v.v \;\mathtt{r}\; Z\vec{y} \to \mathcal{Q}\vec{y}\big((\mathbb{K}\vec{m}\vec{f})v\big)\Big) \to \\ & \quad \Big(\forall \vec{y} \forall v.v \;\mathtt{r}\; \mathcal{F}_i[X := Z]\vec{y} \to \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]\vec{y}\big(m_i(\mathbb{K}\vec{m}\vec{f})v\big)\Big). \end{aligned}$$

The antecedent of this implication unfolds to:

$$\forall \vec{y} \forall v.v \;\mathtt{r}\; Z\vec{y} \;\; \to \exists u.u \;\mathtt{r}\; Z\vec{y} \restriction \mathbb{K}\vec{m}\vec{f}v = \mathbb{K}\vec{m}\vec{f}u$$

which is easily derivable:

$$u : v \;\mathtt{r}\; Z\vec{y} \vdash \mathsf{pack}\,u : \exists u.u \;\mathtt{r}\; Z\vec{y} \restriction \mathbb{K}\vec{m}\vec{f}v = \mathbb{K}\vec{m}\vec{f}u$$

that is,

$$\vdash \lambda u.\,\mathsf{pack}\,u : \forall \vec{y}\, \forall v.v \;\mathtt{r}\; Z\vec{y} \;\; \to \mathcal{Q}\vec{y}\, \big(\mathbb{K}\vec{m}\vec{f}v\big).$$

Therefore we can eliminate the implication and obtain

$$\begin{aligned} \Pi \vdash x_i(\lambda u.\,\mathsf{pack}\,u) : \quad & \forall \vec{y}\, \forall v.v \;\mathtt{r}\; \mathcal{F}_i[X := Z]\vec{y} \\ & \to \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]\vec{y}\, \big(m_i(\mathbb{K}\vec{m}\vec{f})v\big). \end{aligned} \tag{4.22}$$

Obviously

$$\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction x = \mathbb{K}\vec{m}\vec{f}u \vdash w : u \mathtt{\ r\ } Z\vec{x}$$

which allows to conclude by (4.17)

$$\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u \vdash y_i w : f_i u \mathtt{\ r\ } \mathcal{F}_i[X := Z]\vec{\mathbb{c}_i}\vec{x}$$

and using (4.22) we get

$$\begin{aligned}&\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u \vdash \\ &x_i(\lambda u.\, \mathsf{pack}\, u)(y_i w) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})\big(m_i(\mathbb{K}\vec{m}\vec{f})(f_i u)\big).\end{aligned}$$

We have

$$\mathbb{E}_\beta \vdash m_i(\mathbb{K}\vec{m}\vec{f})(f_i u) = \underset{k,i}{\mathsf{out}}(\mathbb{K}\vec{m}\vec{f}u) \text{ and } \mathbb{E}_\beta \vdash \underset{k,i}{\mathsf{out}}(\mathbb{K}\vec{m}\vec{f}u) = \mathbb{D}_i^k(\mathbb{K}\vec{m}\vec{f}u)$$

and by $(\restriction E_R)$,

$$\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u \vdash z = \mathbb{K}\vec{m}\vec{f}u,$$

therefore

$$\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u \vdash m_i(\mathbb{K}\vec{m}\vec{f})(f_i u) = \mathbb{D}_i^k z$$

and (Eq) yields

$$\begin{aligned}&\Pi, w : u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u \vdash \\ &x_i(\lambda u.\, \mathsf{pack}\, u)(y_i w) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z).\end{aligned}$$

Now we proceed to eliminate the extra assumption w using $(\exists E)$, with the previous derivation and the obvious $\Pi \vdash v : \exists u.u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u$ The proviso for $(\exists E)$ holds:

$$u \notin FV\Big(\Pi, \exists u.u \mathtt{\ r\ } Z\vec{x} \restriction z = \mathbb{K}\vec{m}\vec{f}u, \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z)\Big)$$

Therefore

$$\Pi \vdash \mathsf{open}\big(v, w.x_i(\lambda u.\, \mathsf{pack}\, u)(y_i w)\big) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z)$$

and the goal (4.20) is proved.
Therefore

$$\Gamma \vdash \lambda v.\mathsf{open}\big(v, w.x_i(\lambda u.\, \mathsf{pack}\, u)(y_i w)\big) : \mathcal{Q} \subseteq \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]^{\vec{\mathbb{c}_i}, \mathbb{D}_i^k},$$

that is $\Gamma \vdash \mathsf{s}_i : \mathcal{Q} \subseteq \mathcal{F}_i^{\mathtt{r}}[X^+ := \mathcal{Q}]^{\vec{\mathbb{c}_i}, \mathbb{D}_i^k}$, which by (νI) yields:

$$\Gamma \vdash \mathsf{CoIt}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, \mathsf{pack}\, z) : \nu^{\mathtt{r}}\vec{v}(\mathbb{K}\vec{m}\vec{f}w).$$

Finally discharging the assumptions $\vec{x}, \vec{y}, z$ we get:

$$\Theta \vdash \lambda\vec{x}.\lambda\vec{y}.\lambda z.\mathsf{CoIt}_k(\vec{\mathsf{m}}, \vec{\mathsf{s}}, \mathsf{pack}\, z) : \mathbb{K} \mathtt{\ r\ } \mathsf{CoInd}_{\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)}.$$

Next we prove part (ii):

We want to show:

$$\begin{array}{ll}\Theta \vdash & \mathbb{Q}\ \mathtt{r}\ \forall Z.\ \ldots, \mathcal{F}_i\, \mathsf{mon}\, X, \ldots_{(1\leq i\leq k)} \to \\ & \ldots, Z \subseteq \mathcal{F}_i[X := \nu \vee Z]^{\vec{\mathbb{c}_i}}, \ldots_{(1\leq i\leq k)} \to \\ & Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\end{array}$$

which unfolds to

$$\begin{array}{ll}\forall Z^{+}.\forall\vec{m}. & \ldots, m_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X, \ldots_{(1\leq i\leq k)} \to \\ & \forall\vec{f}.\ \ldots, f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := \nu \vee Z]^{\vec{\mathbb{c}_i}}, \ldots_{(1\leq i\leq k)} \to \\ & \mathbb{Q}\vec{m}\vec{f}\ \mathtt{r}\ Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\end{array} \tag{4.23}$$

Assume for $1 \leq i \leq k$

$$x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X \tag{4.24}$$

and $y_i : f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := \nu \vee Z]^{\vec{\mathbb{c}_i}}$, that is

$$y_i : \forall\vec{v}.\forall u.u\ \mathtt{r}\ Z\vec{v} \to f_i u\ \mathtt{r}\ \mathcal{F}_i[X := \nu \vee Z](\vec{\mathbb{c}_i}\vec{v}). \tag{4.25}$$

We need to show $\mathbb{Q}\vec{m}\vec{f}\ \mathtt{r}\ Z \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$, i.e.

$$\forall\vec{v}.\forall w.w\ \mathtt{r}\ Z\vec{v} \to \mathbb{Q}\vec{m}\vec{f}w\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{v}$$

Assume

$$z : w\ \mathtt{r}\ Z\vec{v} \equiv Z^{+}\vec{v}w, \tag{4.26}$$

and let

$$\mathcal{Q} := \lambda\vec{v}, y.\exists u.u\ \mathtt{r}\ Z\vec{v} \restriction y = \mathbb{Q}\vec{m}\vec{f}u$$

Set

$$\begin{array}{lll}\Gamma & := & \Theta, x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X, \quad (1 \leq i \leq k) \\ & & y_i : f_i\ \mathtt{r}\ Z \subseteq \mathcal{F}_i[X := \nu \vee Z]^{\vec{\mathbb{c}_i}}, \quad (1 \leq i \leq k) \\ & & z : w\ \mathtt{r}\ Z\vec{v}\end{array}$$

We need to prove $\Gamma \vdash \mathbb{Q}\vec{m}\vec{f}w\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{v}$, i.e., $\Gamma \vdash \nu^{\mathtt{r}}\vec{v}(\mathbb{Q}\vec{m}\vec{f}w)$
Obviously $\Gamma \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\, \mathsf{mon}\, X^{+}$ and easily we can derive

$$\Gamma \vdash \mathsf{pack}\, z : \mathcal{Q}\vec{v}(\mathbb{Q}\vec{m}\vec{f}w),$$

therefore using the introduction rule (νI^{+}) it suffices to show

$$\Gamma \vdash \mathcal{Q} \subseteq \mathcal{F}_i^{\mathtt{r}}[X^{+} := \nu^{\mathtt{r}} \vee \mathcal{Q}]^{\vec{\mathbb{c}_i}, \mathbb{D}_i^k},\ (1 \leq i \leq k)$$

that is

$$\forall\vec{x}\forall z.\mathcal{Q}\vec{x}z \to \mathcal{F}_i^{\mathtt{r}}[X^{+} := \nu^{\mathtt{r}} \vee \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z).$$

Assume

$$v : \mathcal{Q}\vec{x}z \tag{4.27}$$

and set $\Pi := \Gamma, v : \mathcal{Q}\vec{x}z$. We need to prove

$$\Pi \vdash \mathcal{F}_i^{\mathtt{r}}[X^+ := \nu^{\mathtt{r}} \vee \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z), \quad (1 \leq i \leq k) \tag{4.28}$$

The assumptions (4.24) unfold to:

$$\begin{aligned} x_i : \forall X^+ \forall Y^+ \forall z. \quad & (\forall \vec{y}\, \forall v. v \;\mathtt{r}\; X\vec{y} \;\to zv \;\mathtt{r}\; Y\vec{y}\;) \to \\ & (\forall \vec{y}\, \forall v. v \;\mathtt{r}\; \mathcal{F}_i\vec{y} \;\to m_i zv \;\mathtt{r}\; \mathcal{F}_i[X := Y]\vec{y}\;). \end{aligned} \tag{4.29}$$

We instantiate the predicate variables $X^+ := (\nu \vee Z)^{\mathtt{r}}, Y^+ := \nu^{\mathtt{r}} \vee \mathcal{Q}$ to obtain:

$$\begin{aligned} x_i : \quad & \forall z. \Big(\forall \vec{y}\, \forall v. (\nu \vee Z)^{\mathtt{r}} \vec{y}\, v \to (\nu^{\mathtt{r}} \vee \mathcal{Q})\vec{y}\, (zv)\Big) \to \\ & \Big(\forall \vec{y}\, \forall v. v \;\mathtt{r}\; \mathcal{F}_i[X := \nu \vee Z]\vec{y} \;\to \mathcal{F}_i^{\mathtt{r}}[X^+ := \nu^{\mathtt{r}} \vee \mathcal{Q}]\vec{y}\, (m_i zv)\Big). \end{aligned} \tag{4.30}$$

Now let us derive

$$\vdash \forall \vec{y} \forall v. (\nu \vee Z)^{\mathtt{r}} \vec{y} v \to (\nu^{\mathtt{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v). \tag{4.31}$$

The following derivations are easy:

$$v_1 : u \;\mathtt{r}\; \nu\vec{y} \upharpoonright v = \mathsf{inl}\, u \vdash [\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}\,]v = u$$
$$v_2 : u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u \vdash [\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}\,]v = \mathbb{Q}\vec{m}\vec{f}u.$$

From these it follows, using $(\upharpoonright E)$, (Eq) and $(\upharpoonright I\,)$, respectively:

$$v_1 : u \;\mathtt{r}\; \nu\vec{y} \upharpoonright v = \mathsf{inl}\, u \vdash v_1 : \nu^{\mathtt{r}}\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v)$$
$$v_2 : u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u \vdash v_2 : u \;\mathtt{r}\; Z\vec{y} \upharpoonright [\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v = \mathbb{Q}\vec{m}\vec{f}u$$

and therefore

$$v_1 : u \;\mathtt{r}\; \nu\vec{y} \upharpoonright v = \mathsf{inl}\, u \vdash \mathsf{inl}\, v_1 : (\nu^{\mathtt{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v)$$
$$v_2 : u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u \vdash \mathsf{pack}\, v_2 : \exists u. u \;\mathtt{r}\; Z\vec{y} \upharpoonright [\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v = \mathbb{Q}\vec{m}\vec{f}u.$$

The last derivation implies:

$$v_2 : u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u \vdash \mathsf{inr}\,\mathsf{pack}\, v_2 : (\nu^{\mathtt{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v).$$

Using the two previous derivations by $(\vee E)$ we get:

$$\begin{gathered} u : (\nu \vee Z)^{\mathtt{r}}\vec{y}\, v,\; v : u \;\mathtt{r}\; \nu\vec{y} \upharpoonright v = \mathsf{inl}\, u \vee u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u \vdash \\ \mathsf{case}(v, v_1.\, \mathsf{inl}\, v_1, v_2.\, \mathsf{inr}\,\mathsf{pack}\, v_2) : (\nu^{\mathtt{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v), \end{gathered}$$

by $(\exists E)$ using the previous derivation and the obvious

$$u : (\nu \vee Z)^{\mathtt{r}}\vec{y}\, v \vdash u : \exists u. u \;\mathtt{r}\; \nu\vec{y} \upharpoonright v = \mathsf{inl}\, u \vee u \;\mathtt{r}\; Z\vec{y} \upharpoonright v = \mathsf{inr}\, u$$

we get:

$$\begin{array}{l} u : (\nu \vee Z)^{\mathsf{r}}\vec{y}v \ \vdash \\ \mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big) : (\nu^{\mathsf{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v) \end{array}$$

and descharging the assumption u we conclude

$$\begin{array}{ll} \vdash & \lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big) : \\ & \forall\vec{y}\forall v.(\nu \vee Z)^{\mathsf{r}}\vec{y}v \to (\nu^{\mathsf{r}} \vee \mathcal{Q})\vec{y}([\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v) \end{array}$$

and (4.31) is derived.

Next we instantiate $z := [\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]$ in (4.30) and eliminate the implication using (4.31) obtaining:

$$\begin{array}{l} \Pi \vdash x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big)\Big) : \\ \forall\vec{y}\,\forall v.v\ \mathtt{r}\ \mathcal{F}_i[X := \nu \vee Z]\vec{y}\ \to \mathcal{F}_i^{\mathsf{r}}[X^+ := \nu^{\mathsf{r}} \vee \mathcal{Q}]\vec{y}\ (m_i[\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}]v). \end{array}$$

On the other hand, from $(\upharpoonright E)$ and assumption (4.25) we get:

$$\Pi, w : u\ \mathtt{r}\ Z\vec{x} \upharpoonright z = \mathbb{Q}\vec{m}\vec{f}u \vdash y_i w : f_i u\ \mathtt{r}\ \mathcal{F}_i[X := \nu \vee Z](\vec{\mathbb{c}_i x}),$$

therefore

$$\begin{array}{l} \Pi, w : u\ \mathtt{r}\ Z\vec{x} \upharpoonright z = \mathbb{Q}\vec{m}\vec{f}u \vdash \\ x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big)\Big)(y_i w) : \\ \mathcal{F}_i^{\mathsf{r}}[X^+ := \nu^{\mathsf{r}} \vee \mathcal{Q}](\vec{\mathbb{c}_i x})(m_i[\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}](f_i u)). \end{array}$$

Now observe that

$$\begin{array}{ll} \mathbb{E}_\beta \vdash & m_i[\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}](f_i u) = \mathsf{out}_{k,i}(\mathbb{Q}\vec{m}\vec{f}u) \\ \mathbb{E}_\beta \vdash & \mathsf{out}_{k,i}(\mathbb{Q}\vec{m}\vec{f}u) = \mathbb{D}_i^k(\mathbb{Q}\vec{m}\vec{f}u) \end{array}$$

and therefore

$$\Pi, w : u\ \mathtt{r}\ Z\vec{x} \upharpoonright z = \mathbb{Q}\vec{m}\vec{f}u \vdash z = \mathbb{Q}\vec{m}\vec{f}u$$

yields

$$\Pi, w : u\ \mathtt{r}\ Z\vec{x} \upharpoonright z = \mathbb{Q}\vec{m}\vec{f}u \vdash m_i[\mathsf{Id}, \mathbb{Q}\vec{m}\vec{f}](f_i u) = \mathbb{D}_i^k z.$$

Now (Eq) leads us to:

$$\begin{array}{l} \Pi, w : u\ \mathtt{r}\ Z\vec{x} \upharpoonright z = \mathbb{Q}\vec{m}\vec{f}u \vdash \\ x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\,v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\,v_2)\big)\Big)(y_i w) : \\ \mathcal{F}_i^{\mathsf{r}}[X^+ := \nu^{\mathsf{r}} \vee \mathcal{Q}](\vec{\mathbb{c}_i x})(\mathbb{D}_i^k z). \end{array}$$

Next using $\Pi \vdash v : \mathcal{Q}\vec{x}z$ by $(\exists E)$ we get:

$$\begin{aligned}\Pi \vdash \quad & \mathsf{open}\Big(v, w.x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\, v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\, v_2)\big)\Big)(y_i w)\Big) : \\ & \mathcal{F}_i^{\mathbf{r}}[X^+ := \nu^{\mathbf{r}} \vee \mathcal{Q}](\vec{\mathbb{c}_i}\vec{x})(\mathbb{D}_i^k z)\end{aligned}$$

and the goal (4.28) is proved.
Therefore

$$\begin{aligned}\Gamma \vdash \quad & \lambda v.\mathsf{open}\Big(v, w.x_i\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\,\mathsf{inl}\, v_1, v_2.\,\mathsf{inr}\,\mathsf{pack}\, v_2)\big)\Big)(y_i w)\Big) : \\ & \mathcal{Q} \subseteq \mathcal{F}_i^{\mathbf{r}}[X^+ := \nu^{\mathbf{r}} \vee \mathcal{Q}]^{\vec{\mathbb{c}_i}, \mathbb{D}_i^k},\end{aligned}$$

which by definiton of q_i and (νI^+) yields:

$$\Gamma \vdash \mathsf{CoRec}_k(\vec{\mathsf{m}}, \vec{\mathsf{q}}, \mathsf{pack}\, z) : \nu^{\mathbf{r}}\vec{v}(\mathbb{Q}\vec{m}\vec{f}w)$$

Finally, discharging the assumptions $\vec{x}, \vec{y}, z$, we get:

$$\mathbb{\Theta} \vdash \lambda\vec{x}.\lambda\vec{y}.\lambda z.\mathsf{CoRec}_k(\vec{\mathsf{m}}, \vec{\mathsf{q}}, \mathsf{pack}\, z) : \mathbb{Q}\ \mathtt{r}\ \mathsf{CoInd}^+_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$$

$\dashv$

Proposition 4.6 *If* $\mathcal{D}_i = \langle \mathcal{F}_i, \vec{\mathbb{c}_i}\rangle$, $\mathbb{\Theta} \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathbf{r}}\,\mathsf{mon}\, X^+$ *for* $1 \leq i \leq k$, *and*

$$\mathbb{I} := \lambda\vec{x}\lambda\vec{y}\ .\mathsf{out}_k^{-1}(\vec{x}, \vec{y}\ ).$$

then

$$\mathbb{\Theta} \vdash \lambda\vec{x}\lambda\vec{y}\ .\mathsf{out}_k^{-1}(\vec{\mathsf{m}}, \vec{\mathsf{s}}\ ) : \mathbb{I}\ \mathtt{r}\ \mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$$

where $\mathsf{s_i} := x_i(\lambda zz)y_i$ $(1 \leq i \leq k)$.
Proof. We have to proof $\mathbb{I}\ \mathtt{r}\ \mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, that is

$$\begin{aligned}\mathbb{I}\ \mathtt{r}\ \forall\vec{z}. \quad & \ldots, \mathcal{F}_i\,\mathsf{mon}\, X, \ldots_{(1\leq i\leq k)} \to \\ & \ldots, \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}_i}\vec{z}, \ldots_{(1\leq i\leq k)} \to \\ & \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{z}\end{aligned}$$

which unfolds to:

$$\begin{aligned}\forall\vec{z}\,\forall\vec{m}. \quad & \ldots, m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\, X, \ldots_{(1\leq i\leq k)} \to \\ & \forall\vec{f}.\ \ldots, f_i\ \mathtt{r}\ \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}_i}\vec{z}, \ldots_{(1\leq i\leq k)} \to \\ & \mathbb{I}\vec{m}\vec{f}\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{z}\end{aligned}$$

Set

$$\begin{aligned}\Gamma := \quad & \mathbb{\Theta}, x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\, X, \quad (1 \leq i \leq k) \\ & y_i : f_i\ \mathtt{r}\ \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}_i}\vec{z}\}, \quad (1 \leq i \leq k)\end{aligned}$$

Our goal is then

$$\Gamma \vdash \mathsf{out}_k^{-1}(\vec{\mathsf{m}}, \vec{\mathsf{s}}\,) : \mathbb{I}\vec{m}\vec{f}\ \mathtt{r}\ \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{z}. \tag{4.32}$$

Obviously we have

$$\Gamma \vdash \mathsf{m}_\mathsf{i} : \mathcal{F}_i^{\mathtt{r}}\ \mathsf{mon}\ X^+,\ \ 1 \leq i \leq k \tag{4.33}$$

On the other hand from $\Gamma \vdash x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\ \mathsf{mon}\ X$ is easy to derive

$$\Gamma \vdash x_i(\lambda zz)y_i : m_i(\lambda zz)f_i\ \mathtt{r}\ \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)](\vec{\mathbb{c}_i}\vec{z}),$$

which by lemma 4.7 simplifies to:

$$\Gamma \vdash x_i(\lambda zz)y_i : \mathcal{F}_i^{\mathtt{r}}[X^+ := \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})](\vec{\mathbb{c}_i}\vec{z}\,)(m_i(\lambda zz)f_i) \tag{4.34}$$

Now observe that

$$\begin{array}{ll} \mathbb{E}_\beta \vdash & m_i(\lambda zz)f_i = \mathsf{out}_{k,i}\,\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,) \\ \mathbb{E}_\beta \vdash & \mathsf{out}_{k,i}\,\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,) = \mathbb{D}_i^k\big(\,\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,)\big), \end{array}$$

therefore we get

$$\mathbb{E}_\beta \vdash m_i(\lambda zz)f_i = \mathbb{D}_i^k\big(\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,)\big)$$

and derivation (4.34) becomes

$$\Gamma \vdash x_i(\lambda zz)y_i : \mathcal{F}_i^{\mathtt{r}}[X^+ := \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})](\vec{\mathbb{c}_i}\vec{z}\,)\big(\mathbb{D}_i^k\big(\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,)\big)\big) \tag{4.35}$$

From derivations (4.33) and (4.35) and definition of s_i we get by rule (νI^i):

$$\Gamma \vdash \mathsf{out}_k^{-1}(\vec{\mathsf{m}}, \vec{\mathsf{s}}\,) : \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})\vec{z}\,\mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,)$$

which as $\mathbb{E}_\beta \vdash \mathbb{I}\vec{m}\vec{f} = \mathsf{out}_k^{-1}(\vec{m}, \vec{f}\,)$ is the same as

$$\Gamma \vdash \mathsf{out}_k^{-1}(\vec{x}, \vec{y}\,) : \nu X^+(\mathcal{D}_1^{\mathtt{r}}, \ldots, \mathcal{D}_k^{\mathtt{r}})\vec{z}\ (\mathbb{I}\vec{m}\vec{f})$$

But by definition of realizability this is the same as derivation (4.32), which was our goal.

⊣

The additional requirements $\mathcal{F}_i^{\mathtt{r}}\ \mathsf{mon}\ X^+$ in propositions 4.4, 4.5 and 4.6 are somehow unpleasing, we would like to obtain them from the fact that $\mathcal{F}_i\ \mathsf{mon}\ X$ is realizable. Unfortunately, this is not true in general but we have the following result:

Proposition 4.7 *If* $\Theta \vdash_{\mathsf{MCICD}^\star} \widehat{m} : m\ \mathtt{r}\ \mathcal{F}\ \mathsf{mon}\ X$ *and* $\mathbb{E} \vdash m(\lambda xx) = \lambda xx$ *then* $\Theta \vdash_{\mathsf{MCICD}^\star,\mathbb{E}} \widehat{m} : \mathcal{F}^{\mathtt{r}}\ \mathsf{mon}\ X^+$.

Proof. Instantiating $z := \lambda xx$ in $m \mathtt{\ r\ } \mathcal{F} \,\mathsf{mon}\, X$ (cf. formula (4.14), page 110) and using that $\mathbb{E}_\beta \vdash (\lambda xx)w = w$ we get

$$\begin{aligned}\Theta \vdash_{\mathsf{MCICD}^\star} \widehat{m} :\ & \forall X^+ . \forall Y^+ . (\forall \vec{y}\, \forall w . w \mathtt{\ r\ } X\vec{y} \ \to w \mathtt{\ r\ } Y\vec{y}\ ) \\ & \to (\forall \vec{y}\, \forall u . u \mathtt{\ r\ } \mathcal{F}\vec{y} \ \to m(\lambda xx)u \mathtt{\ r\ } \mathcal{F}[X := Y]\vec{y}\ )\end{aligned}$$

By lemma 4.7 we have

$$m(\lambda xx)u \mathtt{\ r\ } \mathcal{F}[X := Y]\vec{y} \ \equiv (m(\lambda xx)u \mathtt{\ r\ } \mathcal{F}\vec{y}\ )[X^+ := Y^{\mathtt{r}}]$$

But $Y^{\mathtt{r}} \equiv Y^+$, therefore we obtain:

$$\begin{aligned}\Theta \vdash_{\mathsf{MCICD}^\star} \widehat{m} :\ & \forall X^+ . \forall Y^+ . (\forall \vec{y}\, \forall w . X^+ \vec{y}\, w \to Y^+ \vec{y}\, w) \to \\ & (\forall \vec{y}\, \forall u . \mathcal{F}^{\mathtt{r}} \vec{y}\, u \to \mathcal{F}^{\mathtt{r}}[X^+ := Y^+]\vec{y}\, (m(\lambda xx)u)\end{aligned}$$

But $\mathbb{E}, \mathbb{E}_\beta \vdash m(\lambda xx)u = u$, because by assumption $\mathbb{E} \vdash m(\lambda xx) = \lambda xx$ and $\mathbb{E}_\beta \vdash (\lambda xx)u = u$. This yields

$$\begin{aligned}\Theta \vdash_{\mathsf{MCICD}^\star, \mathbb{E}} \widehat{m} :\ & \forall X^+ . \forall Y^+ . X^+ \subseteq Y^+ \to \\ & \mathcal{F}^{\mathtt{r}} \subseteq \mathcal{F}^{\mathtt{r}}[X^+ := Y^+]\end{aligned}$$

That is $\mathcal{F}^{\mathtt{r}} \,\mathsf{mon}\, X^+$ and the proposition follows. ⊣

This proposition says that, assuming the first functor law, the realizability of the monotonicity of $\mathcal{F}$ with respect to X implies the monotonicity of the realizability predicate $\mathcal{F}^{\mathtt{r}}$ with respect to X^+.
This result allows to obtain a realizability soundness theorem where both source and target logical differ essentially only on the underlying object-term system. This is an important difference with the treatment in [Tat94].

4.2.2 The Soundness Theorem

We come to the main result of this chapter, a soundness theorem for our realizability interpretation, which guarantees the correctness of program extraction.

Definition 4.7 *Given an* MCICD*-proof-term r we define the* MCICD**-proof-term $\widetilde{r}$ as follows:*

$$\begin{array}{rcl@{\qquad}rcl}
\widetilde{x} & := & x & \widetilde{\lambda x.r} & := & \lambda x.\widetilde{r} \\
\widetilde{rs} & := & \widetilde{r}\,\widetilde{s} & \widetilde{\langle r, s\rangle} & := & \langle \widetilde{r}, \widetilde{s}\,\rangle \\
\widetilde{\pi_1 r} & := & \pi_1\,\widetilde{r} & \widetilde{\pi_2 r} & := & \pi_2\,\widetilde{r} \\
\widetilde{\mathsf{inl}\, s} & := & \mathsf{pack}(\mathsf{inl}\,\widetilde{s}) & \widetilde{\mathsf{inr}\, s} & := & \mathsf{pack}(\mathsf{inr}\,\widetilde{s}) \\
\widetilde{\mathsf{case}(r, y.s, z.t)} & := & \mathsf{open}(\widetilde{r}, w.\mathsf{case}(w, y.\widetilde{s}, z.\widetilde{t}\,)) & & &
\end{array}$$

$$\begin{array}{rcl}
\widetilde{\mathsf{in}_{k,i}\, t} & := & \mathsf{in}_{k,i}\, \widetilde{t} \\
\widetilde{\mathsf{It}_k(\vec{m}, \vec{s}, t)} & := & \mathsf{It}_k(\widetilde{\vec{m}}, \vec{\mathsf{s}}\,[\vec{m}, \vec{s}\,], \widetilde{t}\ ) \\
\widetilde{\mathsf{Rec}_k(\vec{m}, \vec{s}, t)} & := & \mathsf{Rec}_k(\widetilde{\vec{m}}, \vec{\mathsf{s}}\,[\vec{m}, \vec{s}\,], \widetilde{t}\ ) \\
\widetilde{\mathsf{out}_{k,i}\, t} & := & \mathsf{out}_{k,i}\, \widetilde{t} \\
\widetilde{\mathsf{CoIt}_k(\vec{m}, \vec{s}, t)} & := & \mathsf{CoIt}_k(\widetilde{\vec{m}}, \vec{\mathsf{r}}\,[\vec{m}, \vec{s}], \mathsf{pack}\, \widetilde{t}\ ) \\
\widetilde{\mathsf{CoRec}_k(\vec{m}, \vec{s}, t)} & := & \mathsf{CoRec}_k(\widetilde{\vec{m}}, \vec{\mathsf{q}}\,[\vec{m}, \vec{s}\,], \mathsf{pack}\, \widetilde{t}\ ) \\
\widetilde{\mathsf{out}_k^{-1}(\vec{m}, \vec{s})} & := & \mathsf{out}_k^{-1}(\widetilde{\vec{m}}, \vec{\mathsf{t}}\,[\vec{m}, \vec{s}]\ )
\end{array}$$

where in the cases for (co)iteration, (co)recursion and inversion we have:

$$\begin{array}{rcl}
\mathsf{s}[x,y] & := & \lambda u.\widetilde{y}(\widetilde{x}(\lambda v.v)u) \\
\mathsf{r}[x,y] & := & \lambda v.\mathsf{open}\big(v, w.\widetilde{x}(\lambda u.\, \mathsf{pack}\, u)(\widetilde{y}w)\big) \\
\mathsf{q}[x,y] & := & \lambda v.\mathsf{open}\Big(v, w.\widetilde{x}\Big(\lambda u.\mathsf{open}\big(u, v.\mathsf{case}(v, v_1.\, \mathsf{inl}\, v_1, v_2.\, \mathsf{inr}\, \mathsf{pack}\, v_2)\big)\Big)(\widetilde{y}w)\Big) \\
\mathsf{t}[x,y] & := & \widetilde{x}(\lambda zz)\widetilde{y}
\end{array}$$

and we define $\vec{\mathsf{s}}\,[\vec{x}, \vec{y}\,] := \mathsf{s}[x_1, y_1], \ldots, \mathsf{s}[x_k, y_k]$ *(the same for* r, q*).*

Definition 4.8 *Given a proof-term t, and a subterm m of t such that m occurs in $\vec{m}$ for some subterm of t of one of the following forms*

$$\mathsf{It}_k(\vec{m}, \vec{s}, r), \mathsf{Rec}_k(\vec{m}, \vec{s}, r), \mathsf{CoIt}_k(\vec{m}, \vec{s}, r), \mathsf{CoRec}_k(\vec{m}, \vec{s}, r), \mathsf{out}_k^{-1}(\vec{m}, \vec{s}),$$

we say that m is an on-display *monotonicity witness of t. The set of all on-display monotonicity witnesses of t will be denoted by $\mathcal{W}(t)$.*

Observe for example that

$$\mathcal{W}(\mathsf{It}_k(\vec{m}, \vec{s}, r)) = \mathcal{W}(\vec{m}) \cup \mathcal{W}(\vec{s}) \cup \mathcal{W}(r) \cup \{\vec{m}\}$$

Definition 4.9 *Given a derivation $\Gamma \vdash_{\mathbb{E}} s : A$ we define*

$$\begin{array}{rl}
\mathbb{FFL}(s) := & \{m(\lambda zz) = \lambda y.y \mid m \in \mathcal{W}(s)\} \\
\\
\mathbb{E}^\star(s) := & \mathbb{E} \cup \mathbb{FFL}(s) \\
\\
\mathbb{E}^\star(\vec{s}\,) := & \mathbb{E}^\star(s_1) \cup \ldots \cup \mathbb{E}^\star(s_k)
\end{array}$$

The equations in $\mathbb{FFL}(s)$ represent the first functor law for every on-display monotonicity witness m occurring in s.

Given a context $\Gamma = \{x_1 : A_1, \ldots, x_k : A_k\}$ we set

$$\Gamma^{\mathtt{r}} := \{x_1 : x_1\ \mathtt{r}\ A_1, \ldots, x_k : x_k\ \mathtt{r}\ A_k\},$$

where w.l.o.g. $x_i \notin FV(A_i)$.

We are now ready to prove the soundness theorem of our realizability interpretation.

Theorem 4.1 (Soundness of Realizability for MCICD**)** *If* $\Gamma \vdash_{\mathsf{MCICD},\mathbb{E}} s : A$ *then* $\Gamma^{\mathtt{r}} \vdash_{\mathsf{MCICD}^\star,\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A$
Proof. Induction on $\vdash_{\mathsf{MCICD},\mathbb{E}}$.
Case (Var) If $\Gamma, x : A \vdash x : A$ then obviously also

$$\Gamma^{\mathtt{r}}, x : x \mathrel{\mathtt{r}} A \vdash \widetilde{x} : x \mathrel{\mathtt{r}} A.$$

because $\widetilde{x} = x$.
Case $(\rightarrow I)$. We have $\Gamma \vdash \lambda xs : A \rightarrow B$ coming from $\Gamma, x : A \vdash s : B$. The IH yields $\Gamma^{\mathtt{r}}, x : x \mathrel{\mathtt{r}} A \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} B$. which by $(\rightarrow I)$ yields

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \lambda x\widetilde{s} : x \mathrel{\mathtt{r}} A \rightarrow s \mathrel{\mathtt{r}} B$$

We have $\mathbb{E}_\beta \vdash s = (\lambda xs)x$ and w.l.o.g. $x \notin FV(\Gamma^{\mathtt{r}}, \mathbb{E}^\star(s))$. therefore we get

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \lambda x\widetilde{s} : \forall x. x \mathrel{\mathtt{r}} A \rightarrow (\lambda xs)x \mathrel{\mathtt{r}} B$$

But as $\mathbb{E}^\star(s) = \mathbb{E}^\star(\lambda xs)$ this is the same as $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\lambda xs)} \widetilde{\lambda xs} : \lambda xs \mathrel{\mathtt{r}} A \rightarrow B$.
Case $(\rightarrow E)$. We have $\Gamma \vdash st : B$ coming from $\Gamma \vdash s : A \rightarrow B$, $\Gamma \vdash t : A$. The IH yields $\Gamma \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A \rightarrow B$, that is $\Gamma \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : \forall z.z \mathrel{\mathtt{r}} A \rightarrow sz \mathrel{\mathtt{r}} B$, and $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(t)} \widetilde{t} : t \mathrel{\mathtt{r}} A$. Instantiating $z := t$ and eliminating the implication we get $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s) \cup \mathbb{E}^\star(t)} \widetilde{s}\widetilde{t} : st \mathrel{\mathtt{r}} B$, which is the same as $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(st)} \widetilde{st} : st \mathrel{\mathtt{r}} B$.
Case $(\forall I)$. Assume $\Gamma \vdash_{\mathbb{E}} s : \forall xA$ coming from $\Gamma \vdash_{\mathbb{E}} s : A$ where $x \notin FV(\Gamma, \mathbb{E})$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A$. We can assume w.l.o.g. $x \notin FV(\Gamma^{\mathtt{r}}, \mathbb{E}^\star(s))$, therefore by $(\forall I)$ we get $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : \forall x.s \mathrel{\mathtt{r}} A$, i.e. $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} \forall xA$.
Case $(\forall E)$. We have $\Gamma \vdash_{\mathbb{E}} s : A[x := r]$ coming from $\Gamma \vdash_{\mathbb{E}} s : \forall xA$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} \forall xA$, i.e. $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : \forall x.s \mathrel{\mathtt{r}} A$, which by $(\forall E)$ implies $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : (s \mathrel{\mathtt{r}} A)[x := r]$. As we can assume w.l.o.g. $x \notin FV(s)$ then, by lemma 4.7, we conclude $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A[x := r]$.
Case $(\forall^2 I)$. Assume $\Gamma \vdash_{\mathbb{E}} s : \forall XA$ coming from $\Gamma \vdash_{\mathbb{E}} s : A$ where $X \notin FV(\Gamma)$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A$. As $X \notin FV(\Gamma)$ then $X^+ \notin FV(\Gamma^{\mathtt{r}})$ therefore $(\forall^2 I)$ yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : \forall X^+.s \mathrel{\mathtt{r}} A$. But this is exactly $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} \forall XA$.
Case $(\forall^2 E)$. We have $\Gamma \vdash_{\mathbb{E}} s : A[X := \mathcal{F}]$ coming from $\Gamma \vdash_{\mathbb{E}} s : \forall XA$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} \forall XA$. i.e. $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : \forall X^+.s \mathrel{\mathtt{r}} A$, which by $(\forall^2 E)$ yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : (s \mathrel{\mathtt{r}} A)[X^+ := \mathcal{F}^{\mathtt{r}}]$, which by lemma 4.7, is the same as $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A[X := \mathcal{F}]$.
Case (Eq). We have $\Gamma \vdash_{\mathbb{E}} s : A[x := t]$ coming from $\Gamma \vdash s : A[x := r]$ and $\mathbb{E} \vdash r = t$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A[x := r]$. Observe now that w.l.o.g. $x \notin FV(s)$ therefore we have $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : (s \mathrel{\mathtt{r}} A)[x := r]$. Now by weakening we get $\mathbb{E}^\star(s) \vdash r = t$ which implies $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} r = t$, therefore by (Eq) we get $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : (s \mathrel{\mathtt{r}} A)[x := t]$ which again as $x \notin FV(s)$ is the same as $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A[x := t]$.
Case $(\wedge I)$. Assume $\Gamma \vdash \langle s, t\rangle : A \wedge B$ from $\Gamma \vdash s : A$, $\Gamma \vdash t : B$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)}: \widetilde{s} : s \mathrel{\mathtt{r}} A$ and $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(t)}: \widetilde{t} : t \mathrel{\mathtt{r}} B$. As $\mathbb{E}_\beta \vdash s = \pi_1\langle s, t\rangle$, $\mathbb{E}_\beta \vdash t = \pi_2\langle s, t\rangle$. then we have $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)}: \widetilde{s} : \pi_1\langle s, t\rangle \mathrel{\mathtt{r}} A$ and $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(t)}: \widetilde{t} : \pi_2\langle s, t\rangle \mathrel{\mathtt{r}} B$ and by $(\wedge I)$ we get $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s) \cup \mathbb{E}^\star(t)}: \langle \widetilde{s}, \widetilde{t}\rangle : \langle s, t\rangle \mathrel{\mathtt{r}} A \wedge B$, which is the same as

$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\langle s,t\rangle)}: \widetilde{\langle s,t\rangle} : \langle s,t\rangle \mathrel{\mathtt{r}} A \wedge B$.
Case $(\wedge_2 E)$. We have $\Gamma \vdash \pi_2 s : B$ from $\Gamma \vdash s : A \wedge B$. The IH yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A \wedge B$, i.e., $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : (\pi_1 s \mathrel{\mathtt{r}} A) \wedge (\pi_2 s \mathrel{\mathtt{r}} B)$ which by $(\wedge_2 E)$ yields $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \pi_2 \widetilde{s} : \pi_2 s \mathrel{\mathtt{r}} B$. But this is the same as $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\pi_2 s)} \widetilde{\pi_2 s} : \pi_2 s \mathrel{\mathtt{r}} B$.
Case $(\wedge_1 E)$. Analogous to the previous case.
Case $(\vee_L I)$ Assume $\Gamma \vdash_{\mathbb{E}} \mathsf{inl}\, s : A \vee B$ coming from $\Gamma \vdash_{\mathbb{E}} s : A$. The IH yields

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A.$$

from this and the obvious $\vdash \mathsf{inl}\, s = \mathsf{inl}\, s$ we get $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} A \upharpoonright \mathsf{inl}\, s = \mathsf{inl}\, s$ and $(\vee_L I)$ yields

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \mathsf{inl}\, \widetilde{s} : (s \mathrel{\mathtt{r}} A \upharpoonright \mathsf{inl}\, s = \mathsf{inl}\, s) \vee (s \mathrel{\mathtt{r}} B \upharpoonright \mathsf{inl}\, s = \mathsf{inr}\, s)$$

which is the same as

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \mathsf{inl}\, \widetilde{s} : \Big((z \mathrel{\mathtt{r}} A \upharpoonright \mathsf{inl}\, s = \mathsf{inl}\, z) \vee (z \mathrel{\mathtt{r}} B \upharpoonright \mathsf{inl}\, s = \mathsf{inr}\, z)\Big)[z := s]$$

Therefore $(\exists I)$ yields

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \mathsf{pack}(\mathsf{inl}\, \widetilde{s}) : \exists z.(z \mathrel{\mathtt{r}} A \upharpoonright \mathsf{inl}\, s = \mathsf{inl}\, z) \vee (z \mathrel{\mathtt{r}} B \upharpoonright \mathsf{inl}\, s = \mathsf{inr}\, z)$$

But as $\mathbb{E}^\star(s) = \mathbb{E}^\star(\mathsf{inl}\, s)$ this is exactly $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\mathsf{inl}\, s)} \widetilde{\mathsf{inl}\, s} : \mathsf{inl}\, s \mathrel{\mathtt{r}} A \vee B$.
Case $(\vee_R I)$. Analogous to the previous case.
Case $(\vee E)$. Assume $\Gamma \vdash_{\mathbb{E}} \mathsf{case}(q, y.s, z.t) : C$ from $\Gamma \vdash_{\mathbb{E}} q : A \vee B,\ \Gamma, y : A \vdash_{\mathbb{E}} s : C,\ \Gamma, z : B \vdash_{\mathbb{E}} t : C$. The IH yields

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(q)} \widetilde{q} : q \mathrel{\mathtt{r}} A \vee B \tag{4.36}$$

$$\Gamma^{\mathtt{r}}, y : y \mathrel{\mathtt{r}} A \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s \mathrel{\mathtt{r}} C \tag{4.37}$$

$$\Gamma^{\mathtt{r}}, z : z \mathrel{\mathtt{r}} B \vdash_{\mathbb{E}^\star(t)} \widetilde{t} : t \mathrel{\mathtt{r}} C \tag{4.38}$$

Set $D := (u \mathrel{\mathtt{r}} A \upharpoonright q = \mathsf{inl}\, u) \vee (u \mathrel{\mathtt{r}} B \upharpoonright q = \mathsf{inr}\, u)$. We have

$$\Gamma^{\mathtt{r}}, v : D[u := y] \vdash v : (y \mathrel{\mathtt{r}} A \upharpoonright q = \mathsf{inl}\, y) \vee (y \mathrel{\mathtt{r}} B \upharpoonright q = \mathsf{inr}\, y) \tag{4.39}$$

On the other hand we have

$$\Gamma^{\mathtt{r}}, u : y \mathrel{\mathtt{r}} A \upharpoonright q = \mathsf{inl}\, y \vdash u : y \mathrel{\mathtt{r}} A$$

and by (4.37)

$$\Gamma^{\mathtt{r}}, u : y \mathrel{\mathtt{r}} A \upharpoonright q = \mathsf{inl}\, y, y : y \mathrel{\mathtt{r}} A \vdash \widetilde{s} : s \mathrel{\mathtt{r}} C,$$

Now from

$$\Gamma^{\mathtt{r}}, u : y \mathrel{\mathtt{r}} A \upharpoonright q = \mathsf{inl}\, y \vdash q = \mathsf{inl}\, y$$

and $s = \mathsf{case}(\mathsf{inl}\, y, y.s, z.t) \in \mathbb{E}_\beta$ we get

$$\Gamma^{\mathtt{r}}, u : y \ \mathtt{r}\ A{\upharpoonright} q = \mathsf{inl}\, y \vdash s = \mathsf{case}(q, y.s, z.t),$$

therefore

$$\Gamma^{\mathtt{r}}, u : y \ \mathtt{r}\ A{\upharpoonright} q = \mathsf{inl}\, y, y : y \ \mathtt{r}\ A \vdash \widetilde{s} : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C,$$

and by $(Dsp1)$ (cf. lemma 4.1)

$$\Gamma^{\mathtt{r}}, u : y \ \mathtt{r}\ A{\upharpoonright} q = \mathsf{inl}\, y \vdash_{\mathbb{E}^\star(s)} \widetilde{s}\,[y := u] : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C. \qquad (4.40)$$

Now using (4.38) and assuming w.l.o.g. $y \notin FV(\Gamma^{\mathtt{r}}, B, C)$ we get by $(Dsp2)$

$$\Gamma^{\mathtt{r}}, z : y \ \mathtt{r}\ B \vdash_{\mathbb{E}^\star(t)[z:=y]} \widetilde{t} : t[z := y] \ \mathtt{r}\ C$$

On the other hand using $t[z := y] = \mathsf{case}(\mathsf{inr}\, y, y.s, z.t) \in \mathbb{E}_\beta$ we get

$$\Gamma^{\mathtt{r}}, w : y \ \mathtt{r}\ B{\upharpoonright} q = \mathsf{inr}\, y \vdash_{\mathbb{E}^\star(t)[z:=y]} t[z := y] = \mathsf{case}(q, y.s, z.t).$$

The two previous derivations imply by weakening and (Eq):

$$\Gamma^{\mathtt{r}}, w : y \ \mathtt{r}\ B{\upharpoonright} q = \mathsf{inr}\, y, z : z \ \mathtt{r}\ B \vdash_{\mathbb{E}^\star(t)[z:=y]} \widetilde{t} : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C$$

and from the obvious $\Gamma^{\mathtt{r}}, w : y \ \mathtt{r}\ B{\upharpoonright} q = \mathsf{inr}\, y \vdash_{\mathbb{E}^\star(t)[z:=y]} w : y \ \mathtt{r}\ B$, $(Dsp1)$ yields

$$\Gamma^{\mathtt{r}}, w : y \ \mathtt{r}\ B{\upharpoonright} q = \mathsf{inr}\, y \vdash_{\mathbb{E}^\star(t)[z:=y]} \widetilde{t}[z := w] : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C \qquad (4.41)$$

Derivations (4.40),(4.41) and (4.39) yield by $(\vee E)$:

$$\Gamma^{\mathtt{r}}, v : D[u := y] \vdash \mathsf{case}(v, u.\widetilde{s}\,[y := u], w.\widetilde{t}\,[z := w]) : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C,$$

which making explicity the equational contexts and by α-conversion is the same as

$$\Gamma^{\mathtt{r}}, v : D[u := y] \vdash_{\mathbb{E}^\star(s) \cup \mathbb{E}^\star(t)[z:=y]} \mathsf{case}(v, y.\widetilde{s}, z.\widetilde{t}\,) : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C.$$

Next observe that derivation (4.36) unfolds to $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(q)} \widetilde{q} : \exists u.D$.
Therefore, as $u \notin FV(\Gamma^{\mathtt{r}}, \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C, \exists u.D)$, $(\exists E)$ yield

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s) \cup \mathbb{E}^\star(t)[z:=y] \cup \mathbb{E}^\star(q)} \mathsf{open}(\widetilde{q}, v.\mathsf{case}(v, y.\widetilde{s}, z.\widetilde{t}\,)) : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C.$$

Finally, as w.l.o.g. $y \notin FV(\Gamma^{\mathtt{r}}, s, q, C)$, $(Dsp2)$ yield

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s) \cup \mathbb{E}^\star(t) \cup \mathbb{E}^\star(q)} \mathsf{open}(\widetilde{q}, v.\mathsf{case}(v, y.\widetilde{s}, z.\widetilde{t}\,)) : \mathsf{case}(q, y.s, z.t) \ \mathtt{r}\ C.$$

which is the same as

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\mathsf{case}(q,y.s,z.t))} \widetilde{\mathsf{case}(q,y.s,z.t)} : \mathsf{case}(q,y.s,z.t)\ \mathtt{r}\ C$$

Case (μI). We have $\Gamma \vdash \mathsf{in}_{k,j}\, t : \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)\overrightarrow{\mathbb{c}_j}\vec{s}$ coming from $\Gamma \vdash t : \mathcal{F}_j[X := \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)]\vec{s}$.
By IH we have $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(t)} \widetilde{t} : t\ \mathtt{r}\ \mathcal{F}_j[X := \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)]\vec{s}$ and by proposition 4.2

$$\begin{aligned} \vdash \lambda x.\, \mathsf{in}_{k,j}\, x : \quad & \forall\vec{y}\forall z.z\ \mathtt{r}\ \mathcal{F}_j[X := \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)]\vec{y} \\ & \to \mathbb{C}^k_j z\ \mathtt{r}\ (\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k))\overrightarrow{\mathbb{c}_j}\vec{y} \end{aligned}$$

Therefore instantiating $\vec{y}, z := \vec{s}, t$ and eliminating the implication we have

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(t)} (\lambda x.\mathsf{in}_{k,j}x)\widetilde{t} : \mathbb{C}^k_j t\ \mathtt{r}\ \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)\overrightarrow{\mathbb{c}_j}\vec{s}$$

Finally using subject reduction, the definition of $\widetilde{\mathsf{in}_{k,j}\, t}$ and observing that $\mathbb{E}^\star(\mathsf{in}_{k,j}\, t) = \mathbb{E}^\star(t)$ and $\mathbb{C}^k_j t = \mathsf{in}_{k,j}\, t \in \mathbb{E}_\beta$ we get

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\mathsf{in}_{k,j}\, t)} \widetilde{\mathsf{in}_{k,j}\, t} : \mathsf{in}_{k,j}\, t\ \mathtt{r}\ \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)\overrightarrow{\mathbb{c}_j}\vec{s}.$$

Case (μE). We have $\Gamma \vdash \mathsf{It}_k(\vec{m},\vec{s},r) : \mathcal{K}\vec{t}$ from $\Gamma \vdash r : \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)\vec{t}$,
$\Gamma \vdash m_i : \mathcal{F}_i\, \mathsf{mon}\, X,\ \Gamma \vdash s_i : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\overrightarrow{\mathbb{c}_i}},\ 1 \leq i \leq k.$
By IH we have

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(m_i)} \widetilde{m_i} : m_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X \quad (1 \leq i \leq k)$$

which by proposition 4.7 leads to

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(m_i)\cup\{m_i(\lambda z.z)=\lambda y.y\}} \widetilde{m_i} : \mathcal{F}^{\mathtt{r}}_i\, \mathsf{mon}\, X^+, \quad (1 \leq i \leq k)$$

therefore, by proposition 4.4, part (i), we get

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural} \lambda\vec{x}.\lambda\vec{y}\,.\lambda z.\mathsf{It}_k(\vec{\mathsf{m}},\vec{\mathsf{s}},z) : \mathbb{J}\ \mathtt{r}\ \mathsf{Ind}_{\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)}$$

with $\mathsf{m_i} := \widetilde{m_i}, \mathsf{s_i} := (\lambda u_i.y_i(x_i(\lambda v.v)u_i))$ and

$$\mathbb{E}^\natural := \mathbb{E}^\star(\vec{m}) \cup \{m_i(\lambda z.z) = \lambda y.y \mid 1 \leq i \leq k\}.$$

Instantiating $Z^+ := \mathcal{K}^{\mathtt{r}}$ in $\mathbb{J}\ \mathtt{r}\ \mathsf{Ind}_{\mu X(\mathcal{C}_1,\dots,\mathcal{C}_k)}$ (cf. formula (4.1), page 107) and using lemma 4.7 we get:

$$\begin{aligned} \Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural} \lambda\vec{x}.\lambda\vec{y}\,.\lambda z.\mathsf{It}_k(\vec{\mathsf{m}},\vec{\mathsf{s}},z) : \quad & \forall\vec{n}.\ \dots, n_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X, \dots_{(1\leq i\leq k)} \to \\ & \forall\vec{f}.\ \dots, f_i\ \mathtt{r}\ \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\overrightarrow{\mathbb{c}_i}}, \dots_{(1\leq i\leq k)} \to \\ & \mathbb{J}\vec{n}\vec{f}\ \mathtt{r}\ \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k) \subseteq \mathcal{K} \end{aligned}$$

Next instantiate $n_i, f_i := m_i, s_i$:

$$\begin{aligned} \Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural} \lambda\vec{x}.\lambda\vec{y}\,.\lambda z.\mathsf{It}_k(\vec{\mathsf{m}},\vec{\mathsf{s}},z) : \quad & \dots, m_i\ \mathtt{r}\ \mathcal{F}_i\, \mathsf{mon}\, X, \dots_{(1\leq i\leq k)} \to \\ & \dots, s_i\ \mathtt{r}\ \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\overrightarrow{\mathbb{c}_i}}, \dots_{(1\leq i\leq k)} \to \\ & \mathbb{J}\vec{m}\vec{s}\ \mathtt{r}\ \mu X(\mathcal{C}_1,\dots,\mathcal{C}_k) \subseteq \mathcal{K} \end{aligned}$$

By IH we have $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s_i)} \widetilde{s_i} : s_i \;\mathtt{r}\; \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathfrak{C}_i}}$, hence we can eliminate both implications with $\mathbb{E}^\natural \cup \mathbb{E}^\star(\vec{s})$ and apply subject reduction of the logic to get:

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural \cup \mathbb{E}^\star(\vec{s})} \lambda z.\mathsf{It}_k(\vec{\mathsf{m}}, \vec{\mathsf{t}_\mathsf{s}}, z) : \mathbb{J}\vec{m}\vec{s} \;\mathtt{r}\; \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq \mathcal{K}$$

where $\mathsf{m_i} := \widetilde{m_i}, \mathsf{t_{s_i}} := \lambda u_i.\widetilde{s_i}(\widetilde{m_i}(\lambda v.v)u_i)$, that is,

$$\begin{array}{ll}\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural \cup \mathbb{E}^\star(\vec{s})} & \lambda z.\mathsf{It}_k(\vec{\mathsf{m}}, \vec{\mathsf{t}_\mathsf{s}}, z) : \\ & \forall \vec{y}\, \forall u.u \;\mathtt{r}\; (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\vec{y} \;\to \mathbb{J}\vec{m}\vec{s}u \;\mathtt{r}\; \mathcal{K}\vec{y}\end{array}$$

Again by IH we have $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(r)} \widetilde{r} : r \;\mathtt{r}\; \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t}$. Next instantiating $\vec{y}, u := \vec{t}, r$ and eliminating the implication we obtain:

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\natural \cup \mathbb{E}^\star(\vec{s}) \cup \mathbb{E}^\star(r)} \big(\lambda z.\mathsf{It}_k(\vec{\mathsf{m}}, \vec{\mathsf{t}_\mathsf{s}}, z)\big)\widetilde{r} : \mathbb{J}\vec{m}\vec{s}r \;\mathtt{r}\; \mathcal{K}\vec{t}$$

Finally, observing that $\mathbb{E}^\natural \cup \mathbb{E}^\star(\vec{s}) \cup \mathbb{E}^\star(r) = \mathbb{E}^\star(\mathsf{It}_k(\vec{m}, \vec{s}, r))$ and $\mathbb{J}\vec{m}\vec{s}r = \mathsf{It}_k(\vec{m}, \vec{s}, r) \in \mathbb{E}_\beta$, using subject reduction and the definitions of $\mathbb{J}, \mathsf{It}_k\widetilde{(\vec{m}, \vec{s}, r)}$ we get

$$\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(\mathsf{It}_k(\vec{m}, \vec{s}, r))} \mathsf{It}_k\widetilde{(\vec{m}, \vec{s}, r)} : \mathsf{It}_k(\vec{m}, \vec{s}, r) \;\mathtt{r}\; \mathcal{K}\vec{t}.$$

Case (μE^+). Use the IH and part (ii) of proposition 4.4.
Case (νI). Use the IH and part (i) of proposition 4.5.
Case (νI^+). Use the IH and part (ii) of proposition 4.5.
Case (νI^i). Use the IH and proposition 4.6.
Case (νE). Use the IH and proposition 4.3. ⊣

De flores es la alfombra: muchas hay en tu casa y
entre el musgo acuático canta y trina Xayacamachan:
embriaga su corazón la flor de cacao.
Poema Nahuatl

Es gibt nichts praktischeres als eine gute Theorie.
Immanuel Kant (1724-1804)

5 Programming with Proofs

The applications of lambda calculus to computer science and logic via the Curry-Howard correspondence are well-known, see for example [Bar97, Ber97]. In this chapter we consider a nice application, namely a version of the *programming with proofs* paradigm which was introduced by Krivine and Parigot in [KrPa90] for AF2 (see also [Lei83] for a first explicit formulation of the method). Later in [Par92] Parigot extends the method to conventional inductive definitions whereas in [Raf94] Raffalli adds conventional coinductive definitions. Our contribution is to extend the paradigm to our system of clausular (co)inductive definitions, being this extension the main application of our realizability interpretation.

5.1 Semantics

In this section we define a classical tarskian semantics for MCICD*, and therefore for MCICD also, needed to present the important concept of data type in a model, which is central for programming with proofs (see [Kri93, Par92]), and allows to establish a relation between modified realizability and our realizability concept.

5.1.1 Syntactical Models for the Term System

We start the semantics definition by given a syntactical model of the term system MCICT.

Definition 5.1 (Valuation) *Given a set D, a D-valuation is a function $\nu : Var \to D$. Given a valuation ν, a variable x and $d \in D$ we define the valuation*

$\nu[x/d] : Var \to D$ as:

$$\nu[x/d](y) = \begin{cases} d & if\ y \equiv x \\ \nu(y) & otherwise \end{cases}$$

The set of D-valuations will be denoted with $\mathsf{Val}(D)$.

Definition 5.2 (Applicative Structure) *An applicative structure is a tuple*

$$\mathcal{D} = \left\langle D, \mathsf{app}, \pi_1^\star, {\pi_2}^\star, \mathsf{inl}^\star, \mathsf{inr}^\star, \mathsf{in}_{k,i}^\star, \mathsf{out}_{k,i}^\star \right\rangle$$

where

- *D has at least two elements.*
- *$\mathsf{app} : D \times D \to D$ and we agree to represent app as concatenation, i.e., for $d_1, d_2 \in D$ we set $d_1 d_2 := \mathsf{app}(d_1, d_2)$.*
- *$\pi_1^\star, {\pi_2}^\star, \mathsf{inl}^\star, \mathsf{inr}^\star, \mathsf{in}_{k,i}^\star, \mathsf{out}_{k,i}^\star : D \to D$.*

Definition 5.3 (Syntactical Model) *A Syntactical Model for MCICT is a pair*

$$\mathfrak{D} = \left\langle \mathcal{D}, \mathsf{Sem}_{\mathcal{D}} \right\rangle$$

such that

- *$\mathcal{D}$ is an applicative structure with universe D.*
- *$\mathsf{Sem}_{\mathcal{D}} : \Lambda_{\mathsf{MCICT}} \times \mathsf{Val}(D) \to D$ and we agree to denote*

$$\mathsf{Sem}_{\mathcal{D}}(t, \nu) =: t^{\mathfrak{D}}[\nu]$$

- *$x^{\mathfrak{D}}[\nu] = \nu(x)$. $(MVar)$*
- *If $\forall x \in FV(r).\nu(x) = \nu'(x)$ then $r^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu']$. $(Coinc)$*
- *$(rs)^{\mathfrak{D}}[\nu] = \mathsf{app}(r^{\mathfrak{D}}[\nu], s^{\mathfrak{D}}[\nu]) \equiv r^{\mathfrak{D}}[\nu] s^{\mathfrak{D}}[\nu]$. $(MApp)$*
- *$(\pi_1 r)^{\mathfrak{D}}[\nu] = \pi_1^\star(r^{\mathfrak{D}}[\nu])$ and $(\pi_2 r)^{\mathfrak{D}}[\nu] = {\pi_2}^\star(r^{\mathfrak{D}}[\nu])$. $(MProj)$*
- *$(\mathsf{inl}\, s)^{\mathfrak{D}}[\nu] = \mathsf{inl}^\star(s^{\mathfrak{D}}[\nu])$ and $(\mathsf{inr}\, s)^{\mathfrak{D}}[\nu] = \mathsf{inr}^\star(s^{\mathfrak{D}}[\nu])$. $(MInj)$*
- *$(\mathsf{in}_{k,i}\, s)^{\mathfrak{D}}[\nu] = \mathsf{in}_{k,i}^\star(s^{\mathfrak{D}}[\nu])$. (MIn)*
- *$(\mathsf{out}_{k,i}\, s)^{\mathfrak{D}}[\nu] = \mathsf{out}_{k,i}^\star(s^{\mathfrak{D}}[\nu])$. $(MOut)$*
- *$\forall d \in D.\, \mathsf{app}\left((\lambda x r)^{\mathfrak{D}}[\nu], d\right) = r^{\mathfrak{D}}[\nu[x/d]]$. $(M\beta_{\to})$*
- *$\pi_1^\star\left(\langle r, s\rangle^{\mathfrak{D}}[\nu]\right) = r^{\mathfrak{D}}[\nu]$ and ${\pi_2}^\star\left(\langle r, s\rangle^{\mathfrak{D}}[\nu]\right) = s^{\mathfrak{D}}[\nu]$. $(M\beta_{\times})$*
- *$\mathsf{case}(\mathsf{inl}\, r, x.s, y.t)^{\mathfrak{D}}[\nu] = s^{\mathfrak{D}}[\nu[x/r^{\mathfrak{D}}[\nu]]]$ and*

  $\mathsf{case}(\mathsf{inr}\, r, x.s, y.t)^{\mathfrak{D}}[\nu] = t^{\mathfrak{D}}[\nu[y/r^{\mathfrak{D}}[\nu]]]$. $(M\beta_{+})$

- $\mathsf{It}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t)^{\mathfrak{D}}[\nu] = \Big(s_i\big(m_i\big(\lambda x.\mathsf{It}_k(\vec{m},\vec{s},x)\big)t\big)\Big)^{\mathfrak{D}}[\nu]. \quad (M\beta_{\mathsf{It}})$
- $\mathsf{Rec}_k(\vec{m},\vec{s},\mathsf{in}_{k,i}\,t)^{\mathfrak{D}}[\nu] = \Big(s_i\big(m_i\big(\langle \mathsf{Id}, \lambda z.\mathsf{Rec}_k(\vec{m},\vec{s},z)\rangle\big)t\big)\Big)^{\mathfrak{D}}[\nu]. \quad (M\beta_{\mathsf{Rec}})$
- $\Big(\mathsf{out}_{k,i}\,\mathsf{CoIt}_k(\vec{m},\vec{s},t)\Big)^{\mathfrak{D}}[\nu] = \Big(m_i\big(\lambda z.\mathsf{CoIt}_k(\vec{m},\vec{s},z)\big)(s_i t)\Big)^{\mathfrak{D}}[\nu]. \quad (M\beta_{\mathsf{CoIt}})$
- $\Big(\mathsf{out}_{k,i}\,\mathsf{CoRec}_k(\vec{m},\vec{s},t)\Big)^{\mathfrak{D}}[\nu] = \Big(m_i\big([\mathsf{Id}, \lambda z.\mathsf{CoRec}_k(\vec{m},\vec{s},z)]\big)(s_i t)\Big)^{\mathfrak{D}}[\nu].$

  $(M\beta_{\mathsf{CoRec}})$
- $\Big(\mathsf{out}_{k,i}\,\mathsf{out}_k^{-1}(\vec{m},\vec{t})\Big)^{\mathfrak{D}}[\nu] = \Big(m_i(\lambda z.z)t_i\Big)^{\mathfrak{D}}[\nu]. \quad (M\beta_{\mathsf{Inv}})$
- *If* $\forall d \in D. r^{\mathfrak{D}}[\nu[x/d]] = s^{\mathfrak{D}}[\nu'[x/d]]$ *then*
  $$(\lambda x r)^{\mathfrak{D}}[\nu] = (\lambda x s)^{\mathfrak{D}}[\nu']. \quad (M\xi_{\to})$$
- *If* $r_1^{\mathfrak{D}}[\nu] = r_2^{\mathfrak{D}}[\nu']$ *and* $s_1^{\mathfrak{D}}[\nu] = s_2^{\mathfrak{D}}[\nu']$ *then*
  $$\langle r_1, s_1\rangle^{\mathfrak{D}}[\nu] = \langle r_2, s_2\rangle^{\mathfrak{D}}[\nu']. \quad (M\xi_{\times})$$
- *If* $t_1^{\mathfrak{D}}[\nu] = t_2^{\mathfrak{D}}[\nu']$, $\forall d \in D. q_1^{\mathfrak{D}}[\nu[y/d]] = q_2^{\mathfrak{D}}[\nu'[y/d]]$ *and*
  $$\forall d \in D. r_1^{\mathfrak{D}}[\nu[z/d]] = r_2^{\mathfrak{D}}[\nu'[z/d]]$$

  then
  $$\mathsf{case}(t_1, y.q_1, z.r_1)^{\mathfrak{D}}[\nu] = \mathsf{case}(t_2, y.q_2, z.r_2)^{\mathfrak{D}}[\nu']. \quad (M\xi_{+})$$
- *If* $\vec{m}_1^{\mathfrak{D}}[\nu] = \vec{m}_2^{\mathfrak{D}}[\nu']$, $\vec{s_1}^{\mathfrak{D}}[\nu] = \vec{s_2}^{\mathfrak{D}}[\nu']$ *and* $r_1^{\mathfrak{D}}[\nu] = r_2^{\mathfrak{D}}[\nu']$ *then the following four equalities hold*
  $$\begin{array}{lcl} \mathsf{It}_k(\vec{m}_1,\vec{s}_1,r_1)^{\mathfrak{D}}[\nu] & = & \mathsf{It}_k(\vec{m}_2,\vec{s}_2,r_2)^{\mathfrak{D}}[\nu'] \quad (M\xi_{\mathsf{It}}) \\ \mathsf{Rec}_k(\vec{m}_1,\vec{s}_1,r_1)^{\mathfrak{D}}[\nu] & = & \mathsf{Rec}_k(\vec{m}_2,\vec{s}_2,r_2)^{\mathfrak{D}}[\nu'] \quad (M\xi_{\mathsf{Rec}}) \\ \mathsf{CoIt}_k(\vec{m}_1,\vec{s}_1,r_1)^{\mathfrak{D}}[\nu] & = & \mathsf{CoIt}_k(\vec{m}_2,\vec{s}_2,r_2)^{\mathfrak{D}}[\nu'] \quad (M\xi_{\mathsf{CoIt}}) \\ \mathsf{CoRec}_k(\vec{m}_1,\vec{s}_1,r_1)^{\mathfrak{D}}[\nu] & = & \mathsf{CoRec}_k(\vec{m}_2,\vec{s}_2,r_2)^{\mathfrak{D}}[\nu']. \quad (M\xi_{\mathsf{CoRec}}) \end{array}$$
- *If* $\vec{m}_1^{\mathfrak{D}}[\nu] = \vec{m}_2^{\mathfrak{D}}[\nu']$, $\vec{t_1}^{\mathfrak{D}}[\nu] = \vec{t_2}^{\mathfrak{D}}[\nu']$ *then*
  $$\mathsf{out}_k^{-1}(\vec{m}_1, \vec{t_1})^{\mathfrak{D}}[\nu] = \mathsf{out}_k^{-1}(\vec{m}_2, \vec{t_2})^{\mathfrak{D}}[\nu'] \quad (M\xi_{\mathsf{Inv}})$$

Definition 5.4 (Extensionality) *We say that a syntactical model $\mathfrak{D}$ is extensional if the following holds*

- $(\lambda x.rx)^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu]$, *if* $x \notin FV(r)$. $\quad (M\eta_{\to})$
- $\langle \pi_1 r, \pi_2 r\rangle^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu]. \quad (M\eta_{\times})$

- ○ $\mathsf{case}(r, y.\,\mathsf{inl}\,y, z.\,\mathsf{inr}\,z)^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu]$. $(M\eta_{+})$
- ○ $\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, r)^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu]$. $(M\eta_{\,\mathsf{It}})$
- ○ $\mathsf{out}_k^{-1}\left(\vec{m}, \mathsf{out}_{k,1}\,r, \ldots, \mathsf{out}_{k,k}\,r\right)^{\mathfrak{D}}[\nu] = r^{\mathfrak{D}}[\nu]$. $(M\eta_{\,\mathsf{Inv}})$

Lemma 5.1 (Term Substitution Properties) *The following properties hold for every syntactical model $\mathfrak{D}$:*

- ○ *If $\vec{x} \notin FV(r)$ then $t^{\mathfrak{D}}[\nu[\vec{x}/\vec{d}]] = t^{\mathfrak{D}}[\nu]$.* $(Tsp1)$
- ○ $t[x := s]^{\mathfrak{D}}[\nu] = t^{\mathfrak{D}}[\nu\big[x/s^{\mathfrak{D}}[\nu]\big]]$. $(Tsp2)$

Proof. For $(Tsp1)$ we have $\vec{x} \notin FV(r)$ implies $\nu(y) = \nu[\vec{x}/\vec{d}](y)$ for all $y \in FV(r)$. Therefore by the $(Coinc)$ property we are done.
$(Tsp2)$ is proved by induction on t.
Case $t \equiv x)$.

$$t[x := s]^{\mathfrak{D}}[\nu] = s^{\mathfrak{D}}[\nu] = \nu\big[x/s^{\mathfrak{D}}[\nu]\big](x) \underset{(MVar)}{=} x^{\mathfrak{D}}[\nu\big[x/s^{\mathfrak{D}}[\nu]\big]]$$

Cases $t \equiv rs, \pi_1 r, \pi_2 s, \mathsf{inl}\,s, \mathsf{inr}\,s, \mathsf{in}_{k,i}\,r, \mathsf{out}_{k,i}\,r)$.
Use IH and $(MApp), (Mproj), (Minj), (MIn), (MOut)$ respectively.
Case $t \equiv \lambda yr)$. Goal is $(\lambda y.r[x := s])^{\mathfrak{D}}[\nu] = (\lambda yr)^{\mathfrak{D}}[\nu\big[x/s^{\mathfrak{D}}[\nu]\big]]$.
By IH we have $r[x := s]^{\mathfrak{D}}[\nu[y/d]] = r^{\mathfrak{D}}[\nu[y/d][x/s^{\mathfrak{D}}[\nu]]] = r^{\mathfrak{D}}[\nu[x/s^{\mathfrak{D}}[\nu]][y/d]]$ for all $d \in D$. Therefore by $(M\xi_{\to})$ we are done.
The remaining cases are solved similarly to the previous one via the IH and the respective $(M\xi)$ rule. ⊣

Proposition 5.1 (Soundness of Term Interpretation) *For every two given terms r, s, if $r \to_{\beta\eta} s$ then $\forall \nu \in \mathsf{Val}(D).r^{\mathfrak{D}}[\nu] = s^{\mathfrak{D}}[\nu]$.*
Proof. Induction on $\to_{\beta\eta}$. ⊣

Definition 5.5 *We define the applicative structure*

$$\mathcal{D}_{\mathcal{T}} := \langle D_{\mathcal{T}}, \mathsf{app}, \pi_1^{\star}, {\pi_2}^{\star}, \mathsf{inl}^{\star}, \mathsf{inr}^{\star}, \mathsf{in}^{\star}_{k,i}, \mathsf{out}^{\star}_{k,i}\rangle$$

as follows:

- ○ *For a given term r set $\| r \| := \{s \in \Lambda_{\mathsf{MCICT}} | r =_{\beta\eta} s\}$.*
- ○ $D_{\mathcal{T}} := \{\| r \| \,| r \in \Lambda_{\mathsf{MCICT}}\}$.
- ○ $\mathsf{app} : D_{\mathcal{T}} \times D_{\mathcal{T}} \to D_{\mathcal{T}}$, $\| r \| \| s \| = \| rs \|$.
- ○ $\pi_1^{\star} : D_{\mathcal{T}} \to D_{\mathcal{T}}$, $\pi_1^{\star} \| r \| = \| \pi_1 r \|$.
- ○ ${\pi_2}^{\star} : D_{\mathcal{T}} \to D_{\mathcal{T}}$, ${\pi_2}^{\star} \| r \| = \| \pi_2 r \|$.
- ○ $\mathsf{inl}^{\star} : D_{\mathcal{T}} \to D_{\mathcal{T}}$, $\mathsf{inl}^{\star} \| s \| = \| \mathsf{inl}\,s \|$.

- ○ $\mathsf{inr}^\star : D_{\mathcal{T}} \to D_{\mathcal{T}},\ \mathsf{inr}^\star \parallel s \parallel = \parallel \mathsf{inr}\, s \parallel$.
- ○ $\mathsf{in}^\star_{k,i} : D_{\mathcal{T}} \to D_{\mathcal{T}},\ \mathsf{in}^\star_{k,i} \parallel r \parallel = \parallel \mathsf{in}_{k,i}\, r \parallel$.
- ○ $\mathsf{out}^\star_{k,i} : D_{\mathcal{T}} \to D_{\mathcal{T}},\ \mathsf{out}^\star_{k,i} \parallel r \parallel = \parallel \mathsf{out}_{k,i}\, r \parallel$.

Definition 5.6 *Define* $\mathsf{Sem}_{\mathcal{D}_{\mathcal{T}}} : \Lambda_{\mathsf{MCICT}} \times \mathsf{Val}(D_{\mathcal{T}}) \to D_{\mathcal{T}}$ *as*

$$t^{\mathfrak{D}}[\nu] := \parallel t[\vec{x} := \vec{s}\,] \parallel,$$

where $FV(t) = \vec{x}$ *and* $\nu(x_i) = \parallel s_i \parallel$ *for* $1 \le i \le k$.

Proposition 5.2 $\mathfrak{D}_{\mathcal{T}} = \langle \mathcal{D}_{\mathcal{T}}, \mathsf{Sem}_{\mathcal{D}_{\mathcal{T}}} \rangle$ *is an extensional syntactical model.*
Proof. We prove the properties of the definition:

- ○ $(MVar)$. $x^{\mathfrak{D}}[\nu] = \parallel x[x := s] \parallel = \parallel s \parallel$, but by definition of $x^{\mathfrak{D}}[\nu]$ we have $\nu(x) = \parallel s \parallel$. Therefore $x^{\mathfrak{D}}[\nu] = \nu(x)$.
- ○ $(Coinc)$. Assume $\forall x \in FV(r).\nu(x) = \nu'(x)$. We have $r^{\mathfrak{D}}[\nu] = \parallel r[\vec{x} := \vec{s}] \parallel$ with $FV(r) = \vec{x}$ and $\nu(x_i) = \parallel s_i \parallel$ and as $x_i \in FV(r)$ we also have $\nu'(x_i) = \parallel s_i \parallel$. Therefore $r^{\mathfrak{D}}[\nu] = \parallel r[\vec{x} := \vec{s}] \parallel = r^{\mathfrak{D}}[\nu']$.
- ○ $(MApp)$. Take $FV(rt) = \vec{x},\ \nu(\vec{x}) = \parallel \vec{s} \parallel, FV(r) = \vec{y}, \nu(\vec{y}) = \parallel \vec{s_1} \parallel$, $FV(t) = \vec{z}, \nu(\vec{z}) = \parallel \vec{s_2} \parallel$. So we have $\vec{x} = \vec{y}, \vec{s}$ and $\parallel \vec{s} \parallel = \parallel \vec{s_1}, \vec{s_2} \parallel$.

$$\begin{aligned}(rt)^{\mathfrak{D}}[\nu] &= \parallel (rt)[\vec{x} := \vec{s}] \parallel \\ &= \parallel r[\vec{x} := \vec{s}]t[\vec{x} := \vec{s}] \parallel \\ &= \parallel r[\vec{x} := \vec{s}] \parallel \parallel t[\vec{x} := \vec{s}] \parallel \\ &= \parallel r[\vec{y} := \vec{s_1}\,] \parallel \parallel t[\vec{z} := \vec{s_2}\,] \parallel \\ &= r^{\mathfrak{D}}[\nu] t^{\mathfrak{D}}[\nu]\end{aligned}$$

- ○ $(MProj), (MInj), (MIn), (MOut)$. These cases are solved analogously to $(MApp)$.
- ○ $(M\beta_{\to})$. Take $d := \parallel t \parallel \in D_{\mathcal{T}},\ FV(\lambda x r) = \vec{y}, \nu(\vec{y}) = \parallel \vec{s} \parallel$.

$$\begin{aligned}\mathsf{app}\left((\lambda xr)^{\mathfrak{D}}[\nu], \parallel t \parallel\right) &= \mathsf{app}\left(\ \parallel (\lambda xr)[\vec{y} := \vec{s}] \parallel, \parallel t \parallel\ \right) \\ &= \parallel (\lambda x.r[\vec{y} := \vec{s}])t \parallel \\ &= \parallel r[\vec{y} := \vec{s}][x := t] \parallel\end{aligned}$$

Next observe that $FV(r) = \vec{y}, x$ and $\nu[x/ \parallel t \parallel](\vec{y}) = \nu(\vec{y})$. Moreover, as $x \notin \vec{y} \cup FV(\vec{s})$ (by definition of substitution) we have $r[\vec{y} := \vec{s}][x := t] = r[\vec{y}, x := \vec{s}, t]$. Therefore

$$\begin{aligned}\parallel r[\vec{y} := \vec{s}][x := t] \parallel &= \parallel r[\vec{y}, x := \vec{s}, t] \parallel \\ &= r^{\mathfrak{D}}[\nu[x/ \parallel t \parallel]].\end{aligned}$$

Therefore $app\left((\lambda xr)^{\mathfrak{D}}[\nu], \parallel t \parallel\right) = r^{\mathfrak{D}}[\nu[x/ \parallel t \parallel]]$.

- ○ $(M\beta_{\times}), (M\beta_{+}), (M\beta_{\mathsf{It}}), (M\beta_{\mathsf{Rec}}), (M\beta_{\mathsf{CoIt}}), (M\beta_{\mathsf{CoRec}}), (M\beta_{\mathsf{Inv}})$. These cases are similar to $(M\beta_{\to})$.

- $(M\xi_{\rightarrow})$. Assume $\forall \parallel t \parallel \in D_{\mathcal{T}}.r^{\mathfrak{D}}[\nu[x/ \parallel t \parallel]] = s^{\mathfrak{D}}[\nu'[x/ \parallel t \parallel]]$. In particular we have $r^{\mathfrak{D}}[\nu[x/ \parallel x \parallel]] = s^{\mathfrak{D}}[\nu'[x/ \parallel x \parallel]]$. So if $FV(r) = \vec{x}$, $FV(s) = \vec{y}$, $\nu[x/ \parallel x \parallel](\vec{x}) = \parallel \vec{t} \parallel$, $\nu[x/ \parallel x \parallel](\vec{y}) = \parallel \vec{q} \parallel$ we have

  $$\parallel r[\vec{x} := \vec{t}\,] \parallel = \parallel s[\vec{y} := \vec{q}\,] \parallel$$

  So we have $r[\vec{x} := \vec{t}\,] =_{\beta\eta} s[\vec{y} := \vec{q}]$, which implies $\lambda x.r[\vec{x} := \vec{t}\,] =_{\beta\eta} \lambda x.s[\vec{y} := \vec{q}]$. Therefore

  $$\parallel (\lambda xr)[\vec{x} := \vec{t}\,] \parallel = \parallel (\lambda xs)[\vec{y} := \vec{q}\,] \parallel \tag{5.1}$$

  As $x \notin FV(\lambda xr, \lambda xs)$ by $(Coinc)$ it suffices to show

  $$(\lambda xr)^{\mathfrak{D}}[\nu[x/ \parallel x \parallel]] = (\lambda xs)^{\mathfrak{D}}[\nu[x/ \parallel x \parallel]].$$

  Assume that $(\lambda xr)^{\mathfrak{D}}[\nu[x/ \parallel x \parallel]] = \parallel \lambda x.r[\vec{z} := \vec{s}] \parallel$, so we have $\vec{z} = FV(\lambda xr) = \{\vec{x}\} \setminus \{x\}$ and $\nu[x/ \parallel x \parallel](\vec{z}) = \vec{s}$. But as $\nu[x/ \parallel x \parallel] = \parallel x \parallel$ and $FV(r) = \vec{z}, x$ we conclude $r[\vec{x} := \vec{t}] = r[\vec{z}, x := \vec{s}, x] = r[\vec{z} := \vec{s}]$. Therefore $\parallel \lambda x.r[\vec{z} := \vec{s}] \parallel = \parallel \lambda x.r[\vec{x} := \vec{t}] \parallel$. Analogously we get $(\lambda xs)^{\mathfrak{D}}[\nu[x/ \parallel x \parallel]] = \parallel (\lambda xs)[\vec{y} := \vec{q}\,] \parallel$ and by (5.1) we are done.

- The remaining $(M\xi)$ rules are solved analogously.

Finally we show that the model is extensional, which is easy because our definition of extensionality is just saying that the η equalities must hold, we show for example $(M\eta_{\,\mathsf{It}})$.
First observe that if $\vec{x} := FV(\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, r))$ and $\vec{z} := FV(r)$ we have $\vec{z} \subseteq \vec{x}$. So if $\nu(\vec{z}) = \parallel \vec{t} \parallel$ and $\nu(\vec{x}) = \parallel \vec{s} \parallel$ then w.l.o.g. $\vec{t} \subseteq \vec{s}$. So we can assume $r[\vec{x} := \vec{s}\,] = r[\vec{z} := \vec{t}\,]$.

$$\begin{aligned}
\mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, r)^{\mathfrak{D}}[\nu] \quad &= \parallel \mathsf{It}_k(\vec{m}, \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, r)[\vec{x} := \vec{s}\,] \parallel \\
&= \parallel \mathsf{It}_k\big(\vec{m}[\vec{x} := \vec{s}\,], \mathbb{C}^k_1, \ldots, \mathbb{C}^k_k, r[\vec{x} := \vec{s}\,]\big) \parallel \\
&= \parallel r[\vec{x} := \vec{s}\,] \parallel \\
&= \parallel r[\vec{z} := \vec{t}] \parallel \\
&= r^{\mathfrak{D}}[\nu].
\end{aligned}$$

⊣

5.1.2 Semantics for the Logic MCICD*

Now that we have a notion of term interpretation we can introduce a notion of satisfaction for MCICD*-formulas.

Definition 5.7 (Valuation) *The concept of valuation is extended as follows: A valuation in a set D is a function $\nu : Var \rightarrow D \cup \mathcal{P}(D)$ such that if x (X) is a first-order (second-order) variable then $\nu(x) \in D$ $(\nu(X) \in \mathcal{P}(D))$.*

Definition 5.8 (Satisfaction) *The notion of satisfaction*

$$\nu \models_{\mathcal{M}} A$$

between a model $\mathcal{M}$, a valuation $\nu \in \mathsf{Val}(\mathcal{M})$, and a formula A is the usual one for formulas of second order logic, defined with help of the previously developed term interpretation, and for restrictions and (co)inductive definitions is defined as follows:

$$\begin{aligned} \nu \models A{\restriction}\vec{s} = \vec{t} \quad &:\Leftrightarrow \quad \nu \models A \textit{ and } \nu \models \vec{s} = \vec{t} \\ \nu \models (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\vec{t} \quad &:\Leftrightarrow \quad \nu \models (\forall X.\vec{\mathcal{F}} \,\mathsf{mon}\, X, \vec{\mathcal{F}} \subseteq X^{\vec{\mathbb{c}}} \to X\vec{t}\,) \\ \nu \models (\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k))\vec{t} \quad &:\Leftrightarrow \quad \nu \models (\exists X.\vec{\mathcal{F}} \,\mathsf{mon}\, X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t}\,) \end{aligned}$$

where

$$\begin{aligned} &\vec{\mathcal{F}} \,\mathsf{mon}\, X := \mathcal{F}_1 \,\mathsf{mon}\, X, \ldots, \mathcal{F}_k \,\mathsf{mon}\, X \\ &\vec{\mathcal{F}} \subseteq X^{\vec{\mathbb{c}}} := \mathcal{F}_1 \subseteq X^{\vec{\mathbb{c}_1}}, \ldots, \mathcal{F}_k \subseteq X^{\vec{\mathbb{c}_k}}, \\ &X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} := X \subseteq \mathcal{F}_1^{\vec{\mathbb{c}_1}} \wedge \ldots \wedge X \subseteq \mathcal{F}_k^{\vec{\mathbb{c}_k}} \end{aligned}$$

Lemma 5.2 (Substitution Properties) *The following properties hold:*

- *If $\forall \gamma \in FV(A).\nu(\gamma) = \nu'(\gamma)$ then*

  $$\nu \models A \textit{ if and only if } \nu' \models A. \quad (FCoinc)$$

- *If $\vec{x} \notin FV(A)$ and $\vec{d} \in |\mathcal{M}|$ then*

  $$\nu \models A \textit{ if and only if } \nu[\vec{x}/\vec{d}] \models A. \quad (Fsp1)$$

- *If $\vec{X} \notin FV(A)$ and $\vec{\mathcal{R}} \subseteq |\mathcal{M}|^n$ then*

  $$\nu \models A \textit{ if and only if } \nu[\vec{X}/\vec{\mathcal{R}}] \models A. \quad (Fsp2)$$

- $\nu \models A[x := s]$ *if and only if* $\nu\big[x/s^{\mathcal{M}}[\nu]\big] \models A$. $(Fsp3)$

- *Set $\mathcal{F}^\nu := \{\vec{d} \in |\mathcal{M}|^n \mid \nu[\vec{x}/\vec{d}] \models \mathcal{F}\vec{x}\}$. Then*

  $$\nu \models A[X := \mathcal{F}] \textit{ if and only if } \nu[X/\mathcal{F}^\nu] \models A. \quad (Fsp4)$$

- *If $u \notin FV(A)$ then*

  $$\nu[x/a] \models A \textit{ if and only if } \nu[u/a] \models A[x := u]. \quad (Fsp5)$$

- *If $Y \notin FV(A)$ then*

$$\nu[X/\mathcal{R}] \models A \text{ if and only if } \nu[Y/\mathcal{R}] \models A[X := Y]. \quad (Fsp6)$$

Proof.

- $(FCoinc)$. Induction on A.
- $(Fsp1)$. Immediate from $(FCoinc)$.
- $(Fsp2)$. Immediate from $(FCoinc)$.
- $(Fsp3)$. Induction on A.
- $(Fsp4)$. Induction on A.
- $(Fsp5)$. Immediate from $(Fsp3)$.
- $(Fsp6)$. Immediate from $(Fsp4)$.

⊣

Identity Models

Definition 5.9 *An identity model is a model $\mathcal{M}$ such that for all valuation $\nu \in \mathsf{Val}(\mathcal{M})$:*

$$\nu \models_{\mathcal{M}} r = s \;\Leftrightarrow\; r^{\mathcal{M}}[\nu] = s^{\mathcal{M}}[\nu]$$

Definition 5.10 *Given an arbitrary model $\mathcal{M}$ we define the model $\mathcal{M}^\star$ as follows:*

- *Define the relation $\sim$ on $|\mathcal{M}|$ as follows:*

  $$a \sim b \;:\Leftrightarrow\; \nu[x, y/a, b] \models_{\mathcal{M}} x = y$$

  for some valuation, and therefore for all valuations $\nu \in \mathsf{Val}(\mathcal{M})$.
  It is clear that $\sim$ is an equivalence relation, and we set

  $$\| a \| = \{b \mid a \sim b\} \text{ and } \| A \| = \{\| a \| \;\mid a \in A\}$$

- *Define the universe of $\mathcal{M}^\star$ as:*

  $$|\mathcal{M}^\star| := |\mathcal{M}| / \sim$$

- *Given a valuation $\nu \in \mathsf{Val}(\mathcal{M})$ define the valuation $\widetilde{\nu} \in \mathsf{Val}(\mathcal{M}^\star)$ as follows:*

  $$\widetilde{\nu}(x) = \| \nu(x) \| \qquad \widetilde{\nu}(X) = \| \nu(X) \|$$

- *Given a valuation $\nu \in \mathsf{Val}(\mathcal{M}^\star)$ define a valuation $\nu^\sharp \in \mathsf{Val}(\mathcal{M})$ as follows:*

$$\begin{array}{rcl} \nu^\sharp(x) = a & :\Leftrightarrow & \nu(x) = \| a \| \\ \nu^\sharp(X) = A & :\Leftrightarrow & \nu(X) = \| A \| \end{array}$$

  Observe that $\nu^\sharp$ is not uniquely determined and that

$$\nu(x) = \| \nu^\sharp(x) \| \qquad \nu(X) = \| \nu^\sharp(X) \|$$

- *Define the term interpretation as follows:*

$$r^{\mathcal{M}^\star}[\nu] := \| r^{\mathcal{M}}[\nu^\sharp] \|$$

Proposition 5.3 *The following properties hold:*

1. *The term interpretation in $\mathcal{M}^\star$ is well-defined, that is, if both $\nu_1^\sharp, \nu_2^\sharp$ work as in the previous definition then $\| r^{\mathcal{M}}[\nu_1^\sharp] \| = \| r^{\mathcal{M}}[\nu_2^\sharp] \|$.*
2. *$\nu \models_{\mathcal{M}} A$ if and only if $\widetilde{\nu} \models_{\mathcal{M}^\star} A$.*
3. *$\mathcal{M}$ and $\mathcal{M}^\star$ are elementary equivalent, that is:*

$$\mathcal{M} \models A \;\Leftrightarrow\; \mathcal{M}^\star \models A.$$

4. *$\mathcal{M}^\star$ is an identity model.*

Proof.

1. Induction on r.
2. Induction on A.
3. From part 2.
4. Take $\mu \in \mathsf{Val}(\mathcal{M}^\star)$ we have to show that

$$\mu \models_{\mathcal{M}^\star} r = s \;\Leftrightarrow\; r^{\mathcal{M}^\star}[\mu] = s^{\mathcal{M}^\star}[\mu].$$

   It is easy to see that there is a $\nu \in \mathsf{Val}(\mathcal{M})$ such that $\mu = \widetilde{\nu}$.
   We have $r^{\mathcal{M}^\star}[\mu] = r^{\mathcal{M}^\star}[\widetilde{\nu}] = \| r^{\mathcal{M}}[\nu] \|$ and analogously $s^{\mathcal{M}^\star}[\mu] = \| s^{\mathcal{M}}[\nu] \|$.
   So it suffices to show

$$\widetilde{\nu} \models_{\mathcal{M}^\star} r = s \;\Leftrightarrow \| r^{\mathcal{M}}[\nu] \| = \| s^{\mathcal{M}}[\nu] \|$$

$$\begin{array}{rcl} \widetilde{\nu} \models_{\mathcal{M}^\star} r = s & \Leftrightarrow & \nu \models_{\mathcal{M}} r = s \\[2ex] & \Leftrightarrow & \nu\big[x, y / r^{\mathcal{M}}[\nu], s^{\mathcal{M}}[\nu]\big] \models_{\mathcal{M}} x = y \\[2ex] & \Leftrightarrow & r^{\mathcal{M}}[\nu] \sim s^{\mathcal{M}}[\nu] \\[2ex] & \Leftrightarrow & \| r^{\mathcal{M}}[\nu] \| = \| s^{\mathcal{M}}[\nu] \| \;. \end{array}$$

$\dashv$

The last two properties of the previous proposition shows that to consider only identity models is a harmless restriction, so we can work with every model and assume that it is an identity model.

We present now the main result of this section:

Theorem 5.1 (Soundness of the Logic $\mathsf{MCICD}^\star$) *If* $\Gamma \vdash_{\mathsf{MCICD}^\star,\mathbb{E}} s : A$ *then* $\Gamma, \mathbb{E} \models A$.
Proof. Induction on $\vdash_{\mathsf{MCICD}^\star,\mathbb{E}}$. The case (Var) as well as those involving $\to, \wedge$ and $\vee$ are straightforward.
Case $(\forall I)$. Assume $\nu \models \Gamma, \mathbb{E}$ and observe that as $x \notin FV(\Gamma, \mathbb{E})$ by lemma (5.2), property $(Fsp1)$, we get $\nu[x/a] \models \Gamma, \mathbb{E}$ for every $a \in |\mathcal{M}|$. Therefore the IH yields $\nu[x/a] \models A$ for every $a \in |\mathcal{M}|$, which by definition lead us to $\nu \models \forall x A$.
Case $(\forall E)$. Assume $\nu \models \Gamma, \mathbb{E}$. The IH yields $\nu \models \forall x A$ which in particular lead us to $\nu\big[x/s^{\mathcal{M}}[\nu]\big] \models A$. Finally by $(Fsp3)$ we get $\nu \models A[x := s]$.
Case $(\forall^2 I)$. Analogous to $(\forall I)$ using $(Fsp2)$.
Case $(\forall^2 E)$. Analogous to $(\forall E)$ using $(Fsp4)$.
Case (Eq). Assume $\nu \models \Gamma, \mathbb{E}$ then the IH yields $\nu \models A[x := s]$ and $\nu \models s = t$. By $(Fsp3)$, $\nu \models A[x := s]$ is the same as $\nu\big[x/s^{\mathcal{M}}[\nu]\big] \models A$ and as $\nu \models s = t$ and we can assume that $\mathcal{M}$ is an identity model then $s^{\mathcal{M}}[\nu] = t^{\mathcal{M}}[\nu]$, hence we get $\nu\big[x/t^{\mathcal{M}}[\nu]\big] \models A$, which again by $(Fsp3)$ equals $\nu \models A[x := t]$.
Case $(\upharpoonright I)$. Assume $\nu \models \Gamma, \mathbb{E}$. The IH yields $\nu \models A$ and $\nu \models \vec{s} = \vec{t}$, therefore by definition we get $\nu \models A{\upharpoonright}\vec{s} = \vec{t}$.
Case $(\upharpoonright E)$. Assume $\nu \models \Gamma, \mathbb{E}$. The IH yields $\nu \models A{\upharpoonright}\vec{s} = \vec{t}$ which by definition implies in particular $\nu \models A$.
Case $(\exists I)$. Assume $\nu \models \Gamma, \mathbb{E}$. The IH yields $\nu \models A[x := s]$. Hence, by $(Fsp3)$ we have $\nu\big[x/s^{\mathcal{M}}[\nu]\big] \models A$, which implies by definition $\nu \models \exists x A$.
Case $(\exists E)$. We have $\Gamma \vdash_{\mathbb{E}} B$ coming from $\Gamma \vdash \exists x A$ and $\Gamma, A[x := u] \vdash B$ with $u \notin FV(\Gamma, B, \exists x A)$.
Assume $\nu \models \Gamma, \mathbb{E}$. The first premisse yields, by IH, $\nu \models \exists x A$, i.e.,

$$\nu[x/a] \models A \quad \text{for some } a \in |\mathcal{M}|. \tag{5.2}$$

The goal is $\nu \models B$. We analyse two cases:

- $x \equiv u$. In this case the second premisse becomes $\Gamma, A \vdash B$ and we have $x \notin FV(\Gamma)$, hence as $\nu \models \Gamma$ we get, by $(Fsp1)$, $\nu[x/a] \models \Gamma$. This together with (5.2) yields $\nu[x/a] \models \Gamma, A$, which by the second premisse and IH leads to $\nu[x/a] \models B$. Finally as $x \notin FV(B)$, $(Fsp1)$ yields $\nu \models B$.

- $x \not\equiv u$. As $u \notin FV(\exists x A)$, this case implies $u \notin FV(A)$. Next observe that $a = u^{\mathcal{M}}\big[\nu[u/a]\big]$, which by (5.2) yields $\nu\Big[x/u^{\mathcal{M}}\big[\nu[u/a]\big]\Big] \models A$. Observe now that as $u \notin FV(A)$, we get, using $(Fsp1)$,

$$\nu\Big[u/a\Big]\Big[x/u^{\mathcal{M}}\big[\nu[u/a]\big]\Big] \models A$$

and by $(Fsp3)$ we conclude $\nu[u/a] \models A[x := u]$. This together with $\nu[u/a] \models \Gamma$ (recall that $u \notin FV(\Gamma)$) yield, by the second premisse and IH, $\nu[u/a] \models B$. Finally as $u \notin FV(B)$, $(Fsp1)$ yields $\nu \models B$.

Case (μI). Assume $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)]\vec{t} \tag{5.3}$$

Our goal is to prove

$$\nu \models \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{\mathbb{c}}_i\vec{t}$$

Take $\mathcal{R} \subseteq |\mathcal{M}|^n$, we will show

$$\nu[X/\mathcal{R}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X, \vec{\mathcal{F}} \subseteq X^{\vec{\mathbb{c}}} \to X\vec{\mathbb{c}}_i\vec{t} \tag{5.4}$$

Assume

$$\nu[X/\mathcal{R}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X \tag{5.5}$$

and

$$\nu[X/\mathcal{R}] \models \vec{\mathcal{F}} \subseteq X^{\vec{\mathbb{c}}} \tag{5.6}$$

The goal becomes

$$\nu[X/\mathcal{R}] \models X\vec{\mathbb{c}}_i\vec{t} \tag{5.7}$$

by (5.3),using $(Fsp4)$, we have

$$\nu[X/\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)^\nu] \models \mathcal{F}_i\vec{t}. \tag{5.8}$$

Assumption (5.5) implies

$$\nu[X, Y/\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)^\nu, \mathcal{R}] \models X \subseteq Y \to \mathcal{F}_i \subseteq \mathcal{F}_i[X := Y] \tag{5.9}$$

Take $\nu' := \nu[X, Y/\mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)^\nu, \mathcal{R}]$, we will show $\nu' \models X \subseteq Y$.

Take $\vec{r} \in |\mathcal{M}|$ and assume $\nu'[\vec{z}/\vec{r}] \models X\vec{z}$. This yields by $(Fsp4)$ $\nu[Y/\mathcal{R}][\vec{z}/\vec{r}] \models \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{z}$ which in particular implies

$$\nu[Y/\mathcal{R}][\vec{z}/\vec{r}][X/\mathcal{R}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X \to \vec{\mathcal{F}} \subseteq X^{\vec{\mathbb{c}}} \to X\vec{z}.$$

Now we can eliminate both implications using (5.5),(5.6) after applying $(Fsp1)$,$(Fsp2)$, getting $\nu[Y/\mathcal{R}][\vec{z}/\vec{r}][X/\mathcal{R}] \models X\vec{z}$, that is $\vec{r} \in \mathcal{R}$ which yields $\nu'[\vec{z}/\vec{r}] \models Y\vec{z}$ and therefore $\nu' \models X \subseteq Y$.

From this, by (5.9), we get $\nu' \models \mathcal{F}_i \subseteq \mathcal{F}_i[X := Y]$. On the other hand by (5.8) using $(Fsp2)$, we get $\nu' \models \mathcal{F}_i\vec{t}$, so we conclude $\nu' \models \mathcal{F}_i[X := Y]\vec{t}$ which, by $(Fsp2)$, coincides with $\nu[Y/\mathcal{R}] \models \mathcal{F}_i[X := Y]\vec{t}$. But, by $(Fsp6)$, the last fact is the same as $\nu[X/\mathcal{R}] \models \mathcal{F}_i\vec{t}$. Therefore by (5.6) we get $\nu[X/\mathcal{R}] \models X^{\vec{\mathbb{c}}_i}\vec{t}$, which is the same as $\nu[X/\mathcal{R}] \models X\vec{\mathbb{c}}_i\vec{t}$ and (5.7), and therefore (5.4), is proved.

Case (μE). Take a valuation ν such that $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \mu X(\mathcal{C}_1, \dots, \mathcal{C}_k)\vec{t} \tag{5.10}$$

$$\nu \models \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathfrak{c}}_i} \tag{5.11}$$

$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X \tag{5.12}$$

Our goal is $\nu \models \mathcal{K}\vec{t}$.

By (5.10) we have

$$\nu[X/\mathcal{K}^\nu] \models \vec{\mathcal{F}} \,\mathsf{mon}\, X \to \vec{F} \subseteq X^{\vec{\mathfrak{c}}} \to X\vec{t}$$

which by $(Fsp4)$ yields

$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X, \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathfrak{c}}_i} \to \mathcal{K}\vec{t}.$$

(5.12) lead us to

$$\nu \models \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathfrak{c}}_i} \to \mathcal{K}\vec{t}.$$

Therefore by (5.11) we conclude

$$\nu \models \mathcal{K}\vec{t}$$

Case (μE^+). Take a valuation ν such that $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t} \tag{5.13}$$

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathfrak{c}}_i} \tag{5.14}$$

$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X \tag{5.15}$$

Our goal is $\nu \models \mathcal{K}\vec{t}$.

It is obvious that $\nu \models \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K} \subseteq \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$, therefore by (5.15), using $(Fsp4)$, we conclude

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)]$$

Next observe that by the previous case (μI),

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)] \subseteq \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)^{\vec{\mathfrak{c}}_i}$$

holds, which by transitivity of $\subseteq$ yields

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)^{\vec{\mathfrak{c}}_i}$$

This fact together with (5.14) and observing that

$$\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)^{\vec{\mathfrak{c}}_i} \wedge \mathcal{K}^{\vec{\mathfrak{c}}_i} \equiv (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K})^{\vec{\mathfrak{c}}_i}$$

allow to conclude

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K})^{\vec{\mathfrak{c}}_i} \tag{5.16}$$

On the other hand (5.13) and (5.15), using $(Fsp4)$, imply

$$\nu \models \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K})^{\vec{\mathbb{c}_i}} \rightarrow (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K})\vec{t}$$

Therefore by (5.16) we conclude

$$\nu \models (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K})\vec{t}$$

which clearly implies $\nu \models \mathcal{K}\vec{t}$.
Case (νE). Assume $\nu \models \Gamma, \mathbb{E}$. The IH yields $\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$, therefore there exists $\mathcal{R} \subseteq |\mathcal{M}|^n$ such that

$$\nu[X/\mathcal{R}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X \tag{5.17}$$

$$\nu[X/\mathcal{R}] \models X \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}} \tag{5.18}$$

$$\nu[X/\mathcal{R}] \models X\vec{t} \tag{5.19}$$

By $(5.18), (5.19)$ we have

$$\nu[X/\mathcal{R}] \models \mathcal{F}_i\vec{\mathbb{c}_i}\vec{t} \tag{5.20}$$

We prove now $\nu[X, Y/\mathcal{R}, \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)^\nu] \models X \subseteq Y$. Take

$$\nu' := \nu[X, Y/\mathcal{R}, \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)^\nu] \text{ and } \nu'' := \nu'[\vec{z}/\vec{r}].$$

We assume $\nu'' \models X\vec{z}$, the goal is $\nu'' \models Y\vec{z}$. By the previous assumption, using $(Fsp2)$, we have $\nu[X/\mathcal{R}][\vec{z}/\vec{r}] \models X\vec{z}$; by (5.18),using $(Fsp1)$, we get $\nu[X/\mathcal{R}][\vec{z}/\vec{r}] \models X \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}$, analogously by (5.17) we have $\nu[X/\mathcal{R}][\vec{z}/\vec{r}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X$. Therefore $\nu[X/\mathcal{R}][\vec{z}/\vec{r}] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X \wedge X \subseteq \vec{F}^{\vec{\mathbb{c}}} \wedge X\vec{z}$, which leads to $\nu[\vec{z}/\vec{r}] \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{z}$. This fact, using $(Fsp4)$,implies

$$\nu[Y/\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)^\nu][\vec{z}/\vec{r}] \models Y\vec{z}$$

and by $(Fsp2)$, we conclude $\nu'' \models Y\vec{z}$.
Therefore we have $\nu' \models X \subseteq Y$ which by (5.17) yields

$$\nu' \models \mathcal{F}_i^{\vec{\mathbb{c}_i}} \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := Y]$$

and by (5.20), using $(Fsp2)$,

$$\nu' \models \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := Y]\vec{t}$$

which, again by $(Fsp2)$, implies

$$\nu[Y/\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)^\nu] \models \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := Y]\vec{t},$$

but, by $(Fsp4)$ this is the same as

$$\nu[X/\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)^\nu] \models \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{t}.$$

and by $(Fsp4)$ we conclude

$$\nu \models \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{t}.$$

and we are done.

Case (νI). Take a valuation ν such that $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \mathcal{K}\vec{t} \tag{5.21}$$

$$\nu \models \mathcal{K} \subseteq \mathcal{F}_i[X := \mathcal{K}]^{\vec{\mathbb{c}_i}} \tag{5.22}$$

$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X \tag{5.23}$$

Our goal is $\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$.

The three previous facts yield, using $(Fsp4)$,

$$\nu[X/\mathcal{K}^\nu] \models \mathcal{F}_i \,\mathsf{mon}\, X \wedge X \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}} \wedge X\vec{t}$$

for every $1 \leq i \leq k$. Therefore

$$\nu \models \exists X.\vec{\mathcal{F}} \,\mathsf{mon}\, X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t}$$

and the goal is proved.

Case (νI^+). Take a valuation ν such that $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \mathcal{K}\vec{t} \tag{5.24}$$

$$\nu \models \mathcal{K} \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\vec{\mathbb{c}_i}} \tag{5.25}$$

$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X \tag{5.26}$$

Our goal is $\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$, i.e.,

$$\nu \models \exists X.\vec{\mathcal{F}} \,\mathsf{mon}\, X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t} \tag{5.27}$$

It is obvious that

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K},$$

therefore by (5.26), using $(Fsp4)$, we conclude

$$\nu \models \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]^{\vec{\mathbb{c}_i}} \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\vec{\mathbb{c}_i}}$$

Next observe that by the previous case (νE),

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]^{\vec{\mathbb{c}_i}}$$

holds, which by transitivity of $\subseteq$ yields

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\vec{\mathbb{c}_i}}$$

This fact and (5.25) allow to conclude

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K} \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\vec{\mathbb{c}_i}} \tag{5.28}$$

On the other hand (5.24) implies $\nu \models (\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K})\vec{t}$, which together with (5.28) and (5.26), using $(Fsp4)$, yield

$$\nu[X/(\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K})^{\nu}] \models \vec{\mathcal{F}}\, \mathsf{mon}\, X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t}$$

and the goal (5.27) is proved.

Case (νI^i). Assume $\nu \models \Gamma, \mathbb{E}$. by IH we have

$$\nu \models \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\vec{\mathbb{c}_i}\vec{t} \quad (1 \leq i \leq k) \tag{5.29}$$

$$\nu \models \vec{\mathcal{F}}\, \mathsf{mon}\, X \tag{5.30}$$

By the previous case (νE) we also have

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)] \ \ 1 \leq i \leq k$$

which implies

$$\nu \models \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq \bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]$$

which by (5.30), using $(Fsp4)$, implies

$$\nu \models \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)] \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}\Big[X := \bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\Big].$$

Obviously

$$\nu \models \bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)] \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]$$

Therefore

$$\begin{aligned} \nu \models \quad & \textstyle\bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)] \\ & \subseteq \mathcal{F}_i^{\vec{\mathbb{c}_i}}\Big[X := \textstyle\bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\Big] \end{aligned} \tag{5.31}$$

On the other hand (5.29) yields

$$\nu \models \Big(\bigwedge_{1 \leq i \leq k} \mathcal{F}_i^{\vec{\mathbb{c}_i}}[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\Big)\vec{t}$$

This fact together with (5.30) and (5.31), using $(Fsp4)$, lead us to

$$\nu\Big[X/\Big(\bigwedge_{1\le i\le k}\mathcal{F}_i^{\vec{\mathbb{c}}_i}[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)]\Big)^{\nu}\Big] \models \vec{\mathcal{F}}\,\mathsf{mon}\,X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t}$$

Therefore

$$\nu \models \exists X.\vec{\mathcal{F}}\,\mathsf{mon}\,X \wedge X \subseteq \vec{\mathcal{F}}^{\vec{\mathbb{c}}} \wedge X\vec{t}$$

and the case is done. ⊣

Proposition 5.4 (Validity of the First Functor Law) *If* $\vdash^{\mathsf{can}} m : \mathcal{F}\,\mathsf{mon}\,X$ *then* $\mathcal{M} \models m(\lambda x.x) = \lambda y.y$.
Proof. Immediate from propositions 3.3 and 5.1 assuming that $\mathcal{M}$ is an identity model. ⊣

Proposition 5.5 (Semantical Soundness of Realizability) *If* $\Gamma \vdash_{\mathsf{MCICD},\mathbb{E}} s : A$*,* $\mathcal{W}(s)$ *comprises only canonical witnesses and* $\mathcal{M} \models \Gamma^{\mathtt{r}}, \mathbb{E}$ *then* $\mathcal{M} \models s\ \mathtt{r}\ A$*.*
Proof. Assume $\Gamma \vdash_{\mathsf{MCICD},\mathbb{E}} s : A$, theorem 4.1 implies $\Gamma^{\mathtt{r}} \vdash_{\mathbb{E}^\star(s)} \widetilde{s} : s\ \mathtt{r}\ A$, and theorem 5.1 implies $\Gamma^{\mathtt{r}}, \mathbb{E}^\star(s) \models s\ \mathtt{r}\ A$. By hypothesis we have $\mathcal{M} \models \Gamma^{\mathtt{r}}$ and as $\mathcal{W}(s)$ comprises only canonical witnesses we have, by prop. 5.4, $\mathcal{M} \models \mathbb{FFL}(s)$. By assumption we also have $\mathcal{M} \models \mathbb{E}$ therefore we conclude $\mathcal{M} \models \mathbb{E}^\star(s)$, then $\mathcal{M} \models \Gamma^{\mathtt{r}}, \mathbb{E}^\star(s)$ and therefore $\mathcal{M} \models s\ \mathtt{r}\ A$. ⊣

Corollary 5.1 (Conservation Lemma) *Let* $\mathbb{E}$ *be a set of equations such that* $\mathcal{M} \models \mathbb{E}$*. If* $\vdash_{\mathbb{E}} s : A$ *and* $\mathcal{W}(s)$ *comprises only canonical witnesses then* $\mathcal{M} \models s\ \mathtt{r}\ A$*.*
Proof. Immediate from proposition 5.5. ⊣

5.2 Formal Data Types

We come to the central concept of the programming with proofs paradigm, that of formal data type.

Definition 5.11 *Let* $D[x]$ *be a formula with* $FV(D) = \{x\}$*. We say that* D *is a data type in* $\mathcal{M}$ *if*

$$\mathcal{M} \models \forall x \forall y. y\ \mathtt{r}\ D[x] \leftrightarrow y = x \wedge D[x]$$

With this concept we can represent typed terms in our untyped setting. The direction (←) of this definition can be simplified to $D[x] \rightarrow x\ \mathtt{r}\ D[x]$ which means that every inhabitant of the type D realizes its own inhabitation. The direction (→) says that every realizer y of the inhabitation of D by x is already that inhabitant x.
As a consequence of realizability soundness and conservation lemma we have the following

Corollary 5.2 (Correctness Lemma) *Let f be a function symbol, $\mathcal{D}_i$, $\mathcal{E}$ data types in $\mathcal{M}$ and s_i an inhabitant of $\mathcal{D}_i$ (i.e. $\mathcal{M} \models \mathcal{D}_i s_i$).*
If $\mathcal{W}(t)$ comprises only canonical witnesses, $\mathcal{M}$ satisfies $\mathbb{E}$ and

$$\vdash_{\mathsf{MCICD},\mathbb{E}} t : \forall x_1 \dots \forall x_n . \mathcal{D}_1 x_1, \dots, \mathcal{D}_n x_n \to \mathcal{E} f(x_1, \dots, x_n),$$

then

$$\mathcal{M} \models t s_1 \dots s_n = f(s_1, \dots, s_n).$$

Therefore the MCICT*-term t is a program to compute the function $f^{\mathcal{M}}$.*

Proof. Use the conservation lemma and observe that $\mathcal{M} \models \mathcal{D}_i s_i$ implies $\mathcal{M} \models s_i \;\mathtt{r}\; \mathcal{D}_i s_i$. ⊣

This corollary provides a method of programming: to obtain a program for a function f we just have to derive

$$\mathcal{D}_1 x_1, \dots, \mathcal{D}_n x_n \vdash_{\mathsf{MCICD}} \mathcal{E} f(x_1, \dots, x_n).$$

5.2.1 A Connection with Modified Realizability

The soundness theorem 4.1 shows that our realizability interpretation is good to extract program from proofs. However, the methods of program extraction via realizability are often developed with Kreisel's modified realizability (see [Ben98, Ber93, BBS02]). In this section we show a connection between both concepts of realizability.

Modified realizability ($\mathtt{mr}$) is usually defined in a typed setting, the definition of $t \;\mathtt{mr}\; A$ for first order formulas is the same as for $t \;\mathtt{r}\; A$ except for the case of a universal quantifier. For this case we have

$$t^{\rho \to \tau(A)} \;\mathtt{mr}\; \forall x^\rho . A := \forall x^\rho . (tx)^{\tau(A)} \;\mathtt{mr}\; A$$

where $\tau(A)$ is a type assigned to A. The essential point is that the realizer is a function with domain ρ.

The quantification over typed variables can be represented in our untyped setting with a universal quantifier relativized to a data type $\mathcal{D}$ corresponding to ρ, by defining

$$\forall_{\mathcal{D}} x . A := \forall x . \mathcal{D} x \to A.$$

If we want to consider now a definition in the spirit of modified realizability for the relativized universal formula $\forall_{\mathcal{D}} x . A$ we must state

$$t \;\mathtt{r}\; \forall_{\mathcal{D}} x . A := \forall_{\mathcal{D}} x . tx \;\mathtt{r}\; A,$$

Note that here the realizer t also behaves as a function with domain $\mathcal{D}$. However this definition is not neccesary, for it is equivalent to the original one as the following proposition shows.

Proposition 5.6 *Let $\mathcal{D}$ be a data type in $\mathcal{M}$. Then*

$$\mathcal{M} \models \big(\forall_{\mathcal{D}} x. tx \ \mathtt{r} \ A \big) \leftrightarrow \ t \ \mathtt{r} \ \forall_{\mathcal{D}} x. A$$

Proof. $\Rightarrow$) Assume that $\mathcal{M} \models \forall x. \mathcal{D}x \to tx \ \mathtt{r} \ A$. It suffices to show $\mathcal{M}[x, y/r, s] \models y \ \mathtt{r} \ \mathcal{D}x \to ty \ \mathtt{r} \ A$ for every $r, s \in |\mathcal{M}|$. Suppose $\mathcal{M}[x, y/r, s] \models y \ \mathtt{r} \ \mathcal{D}x$, as $\mathcal{D}$ is a data type then $\mathcal{M}[x, y/r, s] \models y = x \ \wedge \ \mathcal{D}x$. Hence $\mathcal{M}[x, y/r, s] \models \mathcal{D}x$, which by the main assumption yields $\mathcal{M}[x, y/r, s] \models tx \ \mathtt{r} \ A$ and as $\mathcal{M}[x, y/r, s] \models y = x$ we conclude $\mathcal{M}[x, y/r, s] \models ty \ \mathtt{r} \ A$. Therefore $\mathcal{M} \models t \ \mathtt{r} \ \forall_{\mathcal{D}} x. A$.

$\Leftarrow$) Suppose $\mathcal{M} \models t \ \mathtt{r} \ \forall x. \mathcal{D}x \to A$. It suffices to show that $\mathcal{M}[x/s] \models \mathcal{D}x \to tx \ \mathtt{r} \ A$ for $s \in |\mathcal{M}|$. Suppose $\mathcal{M}[x/s] \models \mathcal{D}x$, this implies that $\mathcal{M}[x/s] \models x \ \mathtt{r} \ \mathcal{D}x$, for $\mathcal{D}$ is a data type. Our main assumption implies that $\mathcal{M}[x/s] \models x \ \mathtt{r} \ \mathcal{D}x \to tx \ \mathtt{r} \ A$. Thus $\mathcal{M}[x/s] \models tx \ \mathtt{r} \ A$, and therefore $\mathcal{M} \models t \ \mathtt{r} \ \forall_{\mathcal{D}} x. A$. ⊣

5.2.2 The Canonical Model

We define now the canonical model which will be used to apply the programming with proofs paradigm to obtain some programs.

Definition 5.12 *The canonical model of* MCICD* *is the full identity model for second order logic $\mathfrak{M}$ with universe $D_{\mathcal{T}}$, i.e, the universe is the set of* MCICT*-terms modulo $\beta\eta$-equivalence.*

Our logic has some parameters not determined a priori but only when defining a new data type or a function to be programmed, these are the names of functions to be programmed like pred, add, append, length, or the names for tags of a data type, like nil, cons, head, tail. Every time that we define a data type or want to program a function, we will add these parameters and their interpretations to the canonical model, this expansion is called the *intended model*. As the canonical model alone is not of our interest, we agree to denote the intended model with $\mathfrak{M}$ exactly like the canonical model.

The following proposition will be useful later (see page 150).

Proposition 5.7 *Let $\mathcal{D}$ be a data type with tags $\mathcal{C} = \{\mathbb{c}_1, \ldots, \mathbb{c}_n\}$ and having at least two elements. Let $\mathbb{E}$ be a set of equations of the language $\mathcal{L} = \mathcal{C} \cup \{f_1, \ldots, f_k\}$. Assume that there are interpretations of $f_1, \ldots, f_k$ in the intended domain $\mathcal{T}_{\mathcal{D}}$ satisfying $\mathbb{E}$. Then there are extensions of the interpretations of $\mathbb{c}_1, \ldots, \mathbb{c}_n, f_0, \ldots, f_k$ to the intended model satisfying $\mathbb{E}$.*
Proof. The intended domain is the term model $\mathcal{T}_{\mathcal{D}}$ with universe

$$\mathsf{T}_{\mathcal{D}} := \{t \mid \ \vdash \mathcal{D}t\}$$

and interpretations for every tag in $\mathcal{C}$. Denote with $\mathcal{T}_{\mathcal{D}}^\star$ the expansion of $\mathcal{T}_{\mathcal{D}}$ to the language $\mathcal{L}$. As by assumption we have $\mathcal{T}_{\mathcal{D}} \models \mathbb{E}$ then also $\mathcal{T}_{\mathcal{D}}^\star \models \mathbb{E}$. Set

$$\mathbb{E}^\star = \{r = s \mid r, s \in \mathsf{Term}(\mathcal{L}) \text{ and } \mathcal{T}_{\mathcal{D}}^\star \models r = s\}$$

So $\mathbb{E}^\star$ is the set of equalities between terms of $\mathcal{L}$ which are valid in $\mathcal{T}_\mathcal{D}^\star$. In particular $\mathbb{E} \subseteq \mathbb{E}^\star$ and clearly we have $\mathcal{T}_\mathcal{D}^\star \models \mathbb{E}^\star$.
Now take $\mathcal{K} = \{c_n \mid n \in \mathbb{N}\}$ a set of constants such that $\mathcal{K} \cap \mathcal{L} = \varnothing$ and set $\mathcal{L}' = \mathcal{L} \cup \mathcal{K}$.
For $r, s \in \mathsf{Term}(\mathcal{L}')$ define the equivalence relation $\sim$ as:

$$r \sim s \quad \Leftrightarrow \vdash_{\mathbb{E}^\star} r = s$$

We denote with $\mathcal{T}_\mathcal{D}^{\star\star}$ the model with universe $|\mathcal{T}_\mathcal{D}^{\star\star}| = \mathsf{Term}(\mathcal{L}')/\sim$. The model $\mathcal{T}_\mathcal{D}^{\star\star}$ has the following properties:

- ○ $\mathcal{T}_\mathcal{D}^{\star\star} \models \mathbb{E}^\star$.
  Take $r = s \in \mathbb{E}^\star$, therefore $\mathbb{E}^\star \vdash r = s$, i.e. $r \sim s$ which, using a similar reasoning as in section 5.1.2, yields $\mathcal{T}_\mathcal{D}^{\star\star} \models r = s$.

- ○ If $t_1, t_2 \in |\mathcal{T}_\mathcal{D}|$ and $\mathcal{T}_\mathcal{D} \not\models t_1 = t_2$ then $\mathcal{T}_\mathcal{D}^{\star\star} \not\models t_1 = t_2$. We prove the contrapositive, assume $\mathcal{T}_\mathcal{D}^{\star\star} \models t_1 = t_2$, this implies $\vdash_{\mathbb{E}^\star} t_1 = t_2$ and therefore $\mathcal{T}_\mathcal{D}^\star \models t_1 = t_2$. But as $t_1, t_2 \in |\mathcal{T}_\mathcal{D}|$ we also get $\mathcal{T}_\mathcal{D} \models t_1 = t_2$.

- ○ If $t \in |\mathcal{T}_\mathcal{D}|$ and $c \in \mathcal{K}$ then $\mathcal{T}_\mathcal{D}^{\star\star} \not\models t = c$.
  If we assume $\mathcal{T}_\mathcal{D}^{\star\star} \models t = c$ then $\mathbb{E}^\star \vdash t = c$ which, as c has no ocurrence in $\mathbb{E}^\star, t$ allows to get $\vdash_{\mathbb{E}^\star} \forall x.t = x$, which yields $\mathcal{T}_\mathcal{D} \models \forall x.t = x$. But this contradicts the fact that $\mathcal{D}$ has at least two elements.
  This fact implies that in $|\mathcal{T}_\mathcal{D}^{\star\star}|$ the constants of $\mathcal{K}$ are not interpreted as elements of $\mathsf{T}_\mathcal{D}$.

- ○ If $c, d \in \mathcal{K}$ then $\mathcal{T}_\mathcal{D}^{\star\star} \not\models c = d$. Otherwise we get $\vdash_{\mathbb{E}^\star} c = d$ which yields $\vdash_{\mathbb{E}^\star} \forall x, y.x = y$. But this contradicts the fact that $|\mathcal{T}_\mathcal{D}|$ has at least two elements.

The second property helps to construct an isomorphism h between $\mathcal{T}_\mathcal{D}$ and a submodel of $\mathcal{T}_\mathcal{D}^{\star\star}$.
By the last two properties there are countable many elements in $|\mathcal{T}_\mathcal{D}^{\star\star}|$that are not interpretations of the terms in $\mathsf{T}_\mathcal{D}$. Therefore we can extend the isomorphism h to a bijection $\widehat{h} : |\mathcal{T}_\mathcal{D}^{\star\star}| \to \mathsf{MCICT}/\beta\eta$ which preserves the interpretation of elements of $\mathsf{T}_\mathcal{D}$.
Finally as $\mathbb{E} \subseteq \mathbb{E}^\star$ the first property yields $\mathcal{T}_\mathcal{D}^{\star\star} \models \mathbb{E}$ and as terms ocurring in equations in $\mathbb{E}$ belong to $\mathsf{T}_\mathcal{D}$ and $|\mathfrak{M}| = \mathsf{MCICT}/\beta\eta$ we get using $\widehat{h}$ that $\mathfrak{M} \models \mathbb{E}$. ⊣

5.2.3 Examples of Data Types

In this section we show some examples of useful data types in the intended model $\mathfrak{M}$. Through the whole section data type will mean data type in $\mathfrak{M}$.

The Unit Predicate

The first example of a data type is the unit predicate defined as

$$\mathbb{1} := \lambda y.\star = y.$$

It is obvious that $\mathfrak{M} \models \forall z.\mathbb{1}z \leftrightarrow \star = z$. Therefore to prove that $\mathbb{1}$ is a data type it suffices to show that

$$\mathfrak{M} \models \forall y.y\ \mathtt{r}\ \mathbb{1}\star \leftrightarrow y = \star \wedge \mathbb{1}\star.$$

This will hold if we interpret $\star$ as the (equivalence class of the) identity, $\star^{\mathfrak{M}} := \lambda zz$.

Take an arbitrary valuation ν and a term r, set $\nu' := \nu[r/y]$. We prove $\nu' \models y\ \mathtt{r}\ \mathbb{1}\star \leftrightarrow y = \star \wedge \mathbb{1}\star$. First assume $\nu' \models y = \star \wedge \mathbb{1}\star$. As $\nu' \models y = \star$ it suffices to show $\nu' \models \star\ \mathtt{r}\ \mathbb{1}\star$. We have

$$\begin{aligned} \star\ \mathtt{r}\ \mathbb{1}\star \equiv{}& \ \star\ \mathtt{r}\ (\star = \star)\\ \equiv{}& \ \star\ \mathtt{r}\ \forall X.X\star \to X\star\\ \equiv{}& \ \forall X^{+}.\forall u.u\ \mathtt{r}\ X\star \to \star u\ \mathtt{r}\ X\star. \end{aligned}$$

But as $\star^{\mathfrak{M}} \equiv \lambda zz$ it suffices to show

$$\nu' \models \forall X^{+}.\forall u.u\ \mathtt{r}\ X\star \to u\ \mathtt{r}\ X\star,$$

which is trivial.

Next assume that $\nu' \models y\ \mathtt{r}\ \mathbb{1}\star$, that is

$$\nu' \models \forall X^{+}.\forall u.u\ \mathtt{r}\ X\star \to yu\ \mathtt{r}\ X\star$$

in particular if $X^{+} := \lambda u_1, u_2.u_2 = u_1 u \wedge \mathbb{1}u_1$ we have that

$$\nu' \models \forall u.u = \star u \wedge \mathbb{1}\star \to yu = \star u \wedge \mathbb{1}\star.$$

It is clear that the antecedent holds, therefore from the succedent we get in particular $\nu' \models \forall u.yu = \star u$ which by the interpretation of $\star$ leads to $\nu' \models \forall u.yu = u$, this implies that $rs =_{\beta\eta} s$ for all terms s which leads to $\nu'(y) = r =_{\beta\eta} \lambda zz$. Hence $\nu' \models y = \star$ and the proof is finished.

The above proof also shows that the predicate $\mathbb{1}^{\mathtt{r}} := \lambda y, z.z\ \mathtt{r}\ \mathbb{1}y$ only holds for $y, z := \star, \star$.

The Booleans

The Predicate

$$\mathbb{B} := \mu X\big(\langle \mathbb{1}, \mathsf{true_g}\rangle, \langle \mathbb{1}, \mathsf{false_g}\rangle\big)$$

representing booleans is a data type if we interpret $\mathsf{true}_{\mathsf{g}}^{\mathfrak{M}} := \mathbb{C}^2_1, \mathsf{false}_{\mathsf{g}}^{\mathfrak{M}} := \mathbb{C}^2_2$.

We will show

$$\mathfrak{M} \models \forall x \forall y.y\ \mathtt{r}\ \mathbb{B}x \leftrightarrow y = x \wedge \mathbb{B}x$$

Take r, s arbitrary terms and a valuation ν, set $\nu' := \nu[r, s/x, y]$. Assume $\nu' \models y = x \wedge \mathbb{B}x$. We will show $\nu' \models x\ \mathtt{r}\ \mathbb{B}x$. By hypothesis we have $\nu' \models \mathbb{B}x$ which by definition is the same as

$$\nu' \models \forall X.\mathbb{1} \subseteq X^{\mathsf{true_g}}, \mathbb{1} \subseteq X^{\mathsf{false_g}} \to Xx.$$

which, as $\mathbb{1}$ holds only for $\star$, simplifies to

$$\nu' \models \forall X.X(\mathsf{true}_\mathsf{g}\star), X(\mathsf{false}_\mathsf{g}\star) \to Xx,$$

in particular if $X := \lambda z.z \mathrel{\mathtt{r}} \mathbb{B}z$ we have

$$\nu' \models \mathsf{true}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{true}_\mathsf{g}\star), \mathsf{false}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{false}_\mathsf{g}\star) \to x \mathrel{\mathtt{r}} \mathbb{B}x.$$

Therefore it suffices to show $\nu' \models \mathsf{true}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{true}_\mathsf{g}\star)$ and $\nu' \models \mathsf{false}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{false}_\mathsf{g}\star)$.

$$\begin{array}{ll} \nu' \models & \mathsf{true}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{true}_\mathsf{g}\star) \Leftrightarrow \\ \nu' \models & \mu X^+(\langle \mathbb{1}^\mathtt{r}, \mathsf{true}, \mathbb{C}^2_1\rangle, \langle \mathbb{1}^\mathtt{r}, \mathsf{false}, \mathbb{C}^2_2\rangle)(\mathsf{true}_\mathsf{g}\star)(\mathsf{true}_\mathsf{g}\star) \Leftrightarrow \\ \nu' \models & \forall X^+.X^+(\mathsf{true}_\mathsf{g}\star)(\mathbb{C}^2_1\star), X^+(\mathsf{false}_\mathsf{g}\star)(\mathbb{C}^2_2\star) \to X^+(\mathsf{true}_\mathsf{g}\star)(\mathsf{true}_\mathsf{g}\star) \end{array}$$

But as $\mathsf{true}_\mathsf{g}^\mathfrak{M} := \mathbb{C}^2_1$ it suffices to show

$$\nu' \models \forall X^+.X^+(\mathsf{true}_\mathsf{g}\star)(\mathbb{C}^2_1\star), X^+(\mathsf{false}_\mathsf{g}\star)(\mathbb{C}^2_2\star) \to X^+(\mathsf{true}_\mathsf{g}\star)(\mathbb{C}^2_1\star)$$

which obviously holds. Similarly we conclude $\nu' \models \mathsf{false}_\mathsf{g}\star \mathrel{\mathtt{r}} \mathbb{B}(\mathsf{false}_\mathsf{g}\star)$.

Now assume $\nu' \models y \mathrel{\mathtt{r}} \mathbb{B}x$. We will prove $\nu' \models y = x \wedge \mathbb{B}x$. The assumption is equivalent to

$$\nu' \models \forall X^+.X^+(\mathsf{true}_\mathsf{g}\star)(\mathbb{C}^2_1\star), X^+(\mathsf{false}_\mathsf{g}\star)(\mathbb{C}^2_2\star) \to X^+xy$$

which in particular with $X^+ := \lambda u_1, u_2.u_2 = u_1 \wedge \mathbb{B}u_1$ implies

$$\nu' \models \mathbb{C}^2_1\star = \mathsf{true}_\mathsf{g}\star \wedge \mathbb{B}(\mathsf{true}_\mathsf{g}\star), \mathbb{C}^2_2\star = \mathsf{false}_\mathsf{g}\star \wedge \mathbb{B}(\mathsf{false}_\mathsf{g}\star) \to y = x \wedge \mathbb{B}x.$$

Next observe that $\mathbb{B}(\mathsf{true}_\mathsf{g}\star), \mathbb{B}(\mathsf{false}_\mathsf{g}\star)$ are trivially satisfied and by the interpretations of $\mathsf{true}_\mathsf{g}, \mathsf{false}_\mathsf{g}$ also $\nu' \models \mathbb{C}^2_1\star = (\mathsf{true}_\mathsf{g}\star)$ and $\nu' \models \mathbb{C}^2_2\star = (\mathsf{false}_\mathsf{g}\star)$ hold. Therefore the antecedents of the implication are satisfied and we can conclude $\nu' \models y = x \wedge \mathbb{B}x$.

The Natural Numbers

If $0_\mathsf{g}^\mathfrak{M} := \mathbb{C}^2_1, s^\mathfrak{M} := \mathbb{C}^2_2$ then

$$\mathbb{N} := \mu X\big(\langle \mathbb{1}, 0_\mathsf{g}\rangle, \langle X, s\rangle\big)$$

is a data type representing natural numbers.

Take $r, t \in |\mathfrak{M}|$ and a valuation ν. Set $\nu' := [x, y/r, t]$.
Assume $\nu' \models y \mathrel{\mathtt{r}} \mathbb{N}x$. Our goal is to show $\nu' \models y = x \wedge \mathbb{N}x$. We have

$$\begin{array}{ll} \nu' \models & y \mathrel{\mathtt{r}} \mathbb{N}x \Leftrightarrow \\ \nu' \models & \mu X^+\big(\langle \mathbb{1}^\mathtt{r}, 0_\mathsf{g}, \mathbb{C}^2_1\rangle, \langle X^+, s, \mathbb{C}^2_2\rangle\big)xy \Leftrightarrow \\ \nu' \models & \forall X^+.\mathbb{1}^\mathtt{r} \subseteq {X^+}^{0_\mathsf{g}, \mathbb{C}^2_1}, X^+ \subseteq {X^+}^{s, \mathbb{C}^2_2} \to X^+xy \end{array}$$

which as $\mathbb{1}^{\mathtt{r}}$ only holds for $\star, \star$ simplifies to

$$\nu' \models \forall X^{+}.X^{+}(0_{\mathtt{g}}\star)(\mathbb{C}^2_1\star), X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}xy$$

In particular setting $X^{+} := \lambda u_1, u_2.u_2 = u_1 \wedge \mathbb{N}u_1$, we have

$$\begin{array}{ll} \nu' \models & \mathbb{C}^2_1\star = 0_{\mathtt{g}}\star \wedge \mathbb{N}(0_{\mathtt{g}}\star), (\lambda u_1, u_2.u_2 = u_1 \wedge \mathbb{N}u_1) \subseteq (\lambda u_1, u_2.u_2 = u_1 \wedge \mathbb{N}u_1)^{s,\mathbb{C}^2_2} \\ & \to y = x \wedge \mathbb{N}x \end{array}$$

That is,

$$\begin{array}{ll} \nu' \models & \mathbb{C}^2_1\star = 0_{\mathtt{g}}\star \wedge \mathbb{N}(0_{\mathtt{g}}\star), \\ & \forall uv.v = u \wedge \mathbb{N}u \to \mathbb{C}^2_2 v = su \wedge \mathbb{N}su \\ & \to y = x \wedge \mathbb{N}x \end{array} \tag{5.32}$$

$\nu' \models \mathbb{N}(0_{\mathtt{g}}\star)$ holds trivially and $\nu' \models \mathbb{C}^2_1\star = 0_{\mathtt{g}}\star$ holds, because $0^{\mathfrak{M}}_{\mathtt{g}} = \mathbb{C}^2_1$. Take $p, q \in |\mathfrak{M}|$ and set $\nu'' := \nu'[u, v/p, q]$ and assume $\nu'' \models v = u \wedge \mathbb{N}u$. As $\models \mathbb{N} \subseteq \mathbb{N}^s$ and $\nu'' \models \mathbb{N}u$ we get $\nu'' \models \mathbb{N}su$. Moreover as $\nu'' \models v = u$ then $\nu'' \models \mathbb{C}^2_2 v = \mathbb{C}^2_2 u$, which as $s^{\mathfrak{M}} := \mathbb{C}^2_2$ yields $\nu'' \models \mathbb{C}^2_2 v = su$. Therefore $\nu'' \models \mathbb{C}^2_2 v = su \wedge \mathbb{N}su$, the anteccedents of (5.32) hold and we get $\nu' \models y = x \wedge \mathbb{N}x$.

Now assume $\nu' \models y = x \wedge \mathbb{N}x$. The goal is to show $\nu' \models y \;\mathtt{r}\; \mathbb{N}x$. As $\nu' \models y = x$ it suffices to show $\nu' \models x \;\mathtt{r}\; \mathbb{N}x$.

$$\begin{array}{ll} \nu' \models & \mathbb{N}x \Leftrightarrow \\ \nu' \models & \forall X.\mathbb{1} \subseteq X^{0_{\mathtt{g}}}, X \subseteq X^s \to Xx \Leftrightarrow \\ \nu' \models & \forall X.X(0_{\mathtt{g}}\star), (\forall z.Xz \to Xsz) \to Xx \end{array}$$

This implies in particular for $X := \lambda z.z \;\mathtt{r}\; \mathbb{N}z$

$$\nu' \models 0_{\mathtt{g}}\star \;\mathtt{r}\; \mathbb{N}(0_{\mathtt{g}}\star), (\forall z.z \;\mathtt{r}\; \mathbb{N}z \to sz \;\mathtt{r}\; \mathbb{N}sz) \to x \;\mathtt{r}\; \mathbb{N}x$$

We prove the anteccedents of this implication

- ○ $\nu' \models 0_{\mathtt{g}}\star \;\mathtt{r}\; \mathbb{N}0_{\mathtt{g}}\star$. We have

  $$\begin{array}{ll} \nu' \models & 0_{\mathtt{g}}\star \;\mathtt{r}\; \mathbb{N}0_{\mathtt{g}}\star \Leftrightarrow \\ \nu' \models & \mu X^{+}\big(\langle \mathbb{1}^{\mathtt{r}}, 0_{\mathtt{g}}, \mathbb{C}^2_1\rangle, \langle X^{+}, s, \mathbb{C}^2_2\rangle\big)(0_{\mathtt{g}}\star)(0_{\mathtt{g}}\star) \Leftrightarrow \\ \nu' \models & \forall X^{+}.\mathbb{1}^{\mathtt{r}} \subseteq X^{+0_{\mathtt{g}},\mathbb{C}^2_1}, X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}(0_{\mathtt{g}}\star)(0_{\mathtt{g}}\star) \Leftrightarrow \\ \nu' \models & \forall X^{+}.X^{+}(0_{\mathtt{g}}\star)(\mathbb{C}^2_1\star), X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}(0_{\mathtt{g}}\star)(0_{\mathtt{g}}\star) \underset{0^{\mathfrak{M}}_{\mathtt{g}} \equiv \mathbb{C}^2_1}{\Leftrightarrow} \\ \nu' \models & \forall X^{+}.X^{+}(0_{\mathtt{g}}\star)(0_{\mathtt{g}}\star), X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}(0_{\mathtt{g}}\star)(0_{\mathtt{g}}\star) \end{array}$$

  and the last claim is trivial. Therefore we are done.

- ○ $\nu' \models \forall z.z \;\mathtt{r}\; \mathbb{N}z \to sz \;\mathtt{r}\; \mathbb{N}sz$. Set $\nu'' := \nu[z/t]$ with $t \in |\mathfrak{M}|$, and assume $\nu'' \models z \;\mathtt{r}\; \mathbb{N}z$ i.e.

  $$\nu'' \models \forall X^{+}.X^{+}(0_{\mathtt{g}}\star)(\mathbb{C}^2_1\star), X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}zz \tag{5.33}$$

The goal is to show $\nu'' \models sz \ \mathtt{r} \ \mathbb{N}sz$, i.e.

$$\nu'' \models \forall X^{+}.X^{+}(0_{\mathtt{g}}\star)(\mathbb{C}^2_1\star), X^{+} \subseteq X^{+s,\mathbb{C}^2_2} \to X^{+}(sz)(sz)$$

Take $\nu^* := \nu''[X^{+}/\mathcal{R}]$ with $\mathcal{R} \subseteq |\mathfrak{M}|^2$ and assume $\nu^* \models X^{+}(0_{\mathtt{g}}\star)(\mathbb{C}^2_1\star)$ and $\nu^* \models X^{+} \subseteq X^{+s,\mathbb{C}^2_2}$. These assumptions together with (5.33) yield $\nu^* \models X^{+}zz$ which, by the second assumption, implies $\nu^\star \models X^{+}(sz)(\mathbb{C}^2_2 z)$. Finally as $s^{\mathfrak{M}} \equiv \mathbb{C}^2_2$ we get $\nu^* \models X^{+}(sz)(sz)$ and we are done.

We leave to the reader the verification of the remaining examples.

Sum of Data Types

If $\mathcal{A}, \mathcal{B}$ are data types then their sum (disjoint union)

$$\mathcal{A} + \mathcal{B} := \mu X\big(\langle \mathcal{A}, \mathtt{inl}\rangle, \langle \mathbb{B}, \mathtt{inr}\rangle\big)$$

is a data type if we set $\mathtt{inl}^{\mathfrak{M}} := \mathbb{C}^2_1, \mathtt{inr}^{\mathfrak{M}} := \mathbb{C}^2_2$.

Product of Data Types

If $\mathcal{A}, \mathcal{B}$ are data types then their product

$$\mathcal{A} \times \mathcal{B} := \nu X\big(\langle \mathcal{A}, \pi_1\rangle, \langle \mathbb{B}, \pi_2\rangle\big)$$

is a data type if we set ${\pi_1}^{\mathfrak{M}} := \mathbb{D}^2_1, {\pi_2}^{\mathfrak{M}} := \mathbb{D}^2_2$. The proof relies on the following consequence of the extensionality property $(M\eta_{\,\mathsf{Inv}})$ of $\mathfrak{M}$: If $\nu \models \pi_1\, v = \pi_1\, u$, $\nu \models \pi_2\, v = \pi_2\, u$ then $\nu \models v = u$.
For another concept of product data type which do not need this extensional property see page 170

Function Space of Data Types

If $\mathcal{A}, \mathcal{B}$ are data types then their function space

$$\mathcal{A} \to \mathcal{B} := \lambda f.\forall z.\mathcal{A}z \to \mathcal{B}fz$$

is a data type. Observe that this predicate is not (co)inductive.

Lists

Given a data type $\mathcal{A}$ we set

$$\mathcal{L}_{\mathcal{A}} := \mu X\big(\langle \mathbb{1}, \mathsf{nil}_{\mathtt{g}}\rangle, \langle A \times X, \mathsf{cons}\rangle\big).$$

$\mathcal{L}_{\mathcal{A}}$ defines the set of lists of elements of the data type A, which is again a data type if $\mathsf{nil}^{\mathfrak{M}}_{\mathtt{g}} := \mathbb{C}^2_1, \mathsf{cons}^{\mathfrak{M}} := \mathbb{C}^2_2$.

Streams

Given a data type $\mathcal{A}$ we would like the predicate of $\mathcal{A}$-streams

$$\mathcal{S}_{\mathcal{A}} := \nu X\big(\langle \mathcal{A}, \mathsf{head}\rangle, \langle X, \mathsf{tail}\rangle\big)$$

to be again a data type if we interpret $\mathsf{head} := \mathbb{D}^2_1, \mathsf{tail} := \mathbb{D}^2_2$. However even if $\mathcal{A}$ is a data type we cannot prove that $\mathcal{S}_{\mathcal{A}}$ is a data type. When trying to prove

$$\mathfrak{M} \models \forall xy.y \;\mathbf{r}\; \mathcal{S}_A[x] \leftrightarrow y = x \wedge \mathcal{S}_A[x]$$

in the direction from left to right we cannot get $y = x \wedge \mathcal{S}_A[x]$ but only

$$\mathcal{A}\,\mathsf{head}\,\mathsf{tail}^k\, x \quad \text{for every } k \in \mathbb{N}$$

and Leibniz' equality is to weak to conclude $\mathcal{S}_A[x]$ from this fact.

A solution to this problem will be given in section 6.5.1.

5.3 Programming with Proofs in MCICD

Now that we have data types at hand we can program some functions on them following the programming with proofs method of [KrPa90, Par92].

We proceed as follows to program a function

$$f : \mathcal{D}_1, \ldots, \mathcal{D}_n \to \mathcal{E}$$

between data types in $\mathfrak{M}$:

1. The specification of the function f is given by some equations $\mathbb{E}(f)$, semantically defining it.
2. Prove that $\vdash_{\mathbb{E}(f)} t : \forall \vec{x}.\mathcal{D}_1 x_1, \ldots, \mathcal{D}_n x_n \to \mathcal{E}(f\vec{x})$
3. If $\mathcal{W}(t)$ contains only canonical witnesses and $\mathcal{M} \models \mathbb{E}(f)$ then the correctness lemma (p. 143) guarantees that t is a program for f.

We will denote with $\overline{f}$ the program t for f extracted from the proof in the step 2 above.
Observe that the program $\overline{f}$ is obtained from the proof of the formula expressing the fact that the function f has the intended type. According to the step 3 above, to guarantee the correctness of the program $\overline{f}$ we need to prove the satisfiability of the specification set $\mathbb{E}(f)$, the usual way to do this is to obtain first the program $\overline{f}$ and then check that setting $f^{\mathfrak{M}} := \overline{f}$ the set $\mathbb{E}(f)$ is satisfied. Due to proposition 5.7 it suffices to check satisfiability in the intended domain only.

This method differs from the usual program extraction methodology, as mentioned in [Par92], in that we consider the specification of an algorithm as primitive. Instead of getting a program directly from the specification of the task to be programmed, which usually lead us to extract programs from proofs of the form $\forall\vec{x}\,\exists\vec{y}.\Phi(\vec{x},\vec{y})$, involving unpleasant existential formulas, we extract a program from the specification of an algorithm solving the original task, in our case the algorithms are specified by equations. To construct a program we give a equational specification of an algorithm, which defines a function, and then write a proof of the fact that the function has the intended type. The program is automatically generated from the proof, so that we do not have to work within the programming language (i.e. within the lambda calculus), in particular with this approach we do not need to calculate explicitly a single realizer, a big advantage in comparison to the method in [Tat93], for example.

Let us see some examples.

5.3.1 Programming Functions with Iteration or Recursion

The Negation on Booleans

We define a unary function $\mathsf{not} : \mathbb{B} \to \mathbb{B}$ such that:

- ○ $\mathsf{not}\ \mathsf{true_g} x = \mathsf{false_g} x$
- ○ $\mathsf{not}\ \mathsf{false_g} x = \mathsf{true_g} x$

Let $\mathbb{E}(\mathsf{not})$ the set containing these two equations.
We have

$$\vdash_{\mathbb{E}(\mathsf{not})} \lambda y.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, \overline{\mathsf{false_g}}, \overline{\mathsf{true_g}}, y) : \forall x.\mathbb{B}x \to \mathbb{B}\mathsf{not}\,x$$

Therefore $\overline{\mathsf{not}} := \lambda y.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, \overline{\mathsf{false_g}}, \overline{\mathsf{true_g}}, y)$ is a function computing the negation.

Even test function

We define a unary function $\mathsf{even?}$ such that:

- ○ $\mathsf{even?}\ 0_{\mathsf{g}} x = \mathsf{true_g} x$
- ○ $\mathsf{even?}\ sx = \mathsf{not}(\mathsf{even?} x)$

Let $\mathbb{E}(\mathsf{even?})$ the set containing these two equations.
We have

$$\vdash_{\mathbb{E}(\mathsf{even?})} \lambda y.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{Id}}, \overline{\mathsf{true_g}}, \overline{\mathsf{not}}, y) : \forall x.\mathbb{N}x \to \mathbb{B}\mathsf{even?}x$$

Addition of Natural Numbers

We need to define a binary function ad such that

- $\mathsf{ad}(x, 0_{\mathsf{g}}y) = x$
- $\mathsf{ad}(x, sy) = s(\mathsf{ad}(x, y))$

Let $\mathbb{E}(\mathsf{ad})$ be the set containing the two equations above. We start by proving that

$$\vdash_{\mathbb{E}(\mathsf{ad})} \forall x.\forall y.\mathbb{N}x, \mathbb{N}y \to \mathbb{N}\mathsf{ad}(x, y).$$

We will prove $u : \mathbb{N}x, v : \mathbb{N}y \vdash_{\mathbb{E}(\mathsf{ad})} \mathbb{N}\mathsf{ad}(x, y)$, using the rule (μE) with $\mathcal{K} := \lambda y.\mathbb{N}\mathsf{ad}(x, y)$.

- $u : \mathbb{N}x, v : \mathbb{N}y \vdash_{\mathbb{E}(\mathsf{ad})} \mathbb{1} \subseteq \mathcal{K}^{0_{\mathsf{g}}}$ we have

$$\begin{array}{rcl}
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathbb{1}z & \vdash_{\mathbb{E}(\mathsf{ad})} & u : \mathbb{N}x \\
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathbb{1}z & \vdash_{\mathbb{E}(\mathsf{ad})} & u : \mathbb{N}\mathsf{ad}(x, 0_{\mathsf{g}}z) \\
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathbb{1}z & \vdash_{\mathbb{E}(\mathsf{ad})} & u : \mathcal{K}0_{\mathsf{g}}z \\
u : \mathbb{N}x, v : \mathbb{N}y & \vdash_{\mathbb{E}(\mathsf{ad})} & \lambda w.u : \mathbb{1} \subseteq \mathcal{K}^{0_{\mathsf{g}}}
\end{array}$$

- $u : \mathbb{N}x, v : \mathbb{N}y \vdash_{\mathbb{E}(\mathsf{ad})} \mathcal{K} \subseteq \mathcal{K}^{s}$ we have

$$\begin{array}{rcl}
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathcal{K}z & \vdash_{\mathbb{E}(\mathsf{ad})} & w : \mathbb{N}\mathsf{ad}(x, z) \\
 & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathbb{C}_2^2 : \mathbb{N} \subseteq \mathbb{N}^{s} \\
 & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathbb{C}_2^2 : \mathbb{N}\mathsf{ad}(x, z) \to \mathbb{N}s(\mathsf{ad}(x, z)) \\
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathcal{K}z & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathsf{in}_{2,2}\, w : \mathbb{N}s(\mathsf{ad}(x, z)) \\
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathcal{K}z & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathsf{in}_{2,2}\, w : \mathbb{N}\mathsf{ad}(x, sz) \\
u : \mathbb{N}x, v : \mathbb{N}y, w : \mathcal{K}z & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathsf{in}_{2,2}\, w : \mathcal{K}sz \\
u : \mathbb{N}x, v : \mathbb{N}y & \vdash_{\mathbb{E}(\mathsf{ad})} & \lambda w.\, \mathsf{in}_{2,2}\, w : \mathcal{K} \subseteq \mathcal{K}^{s} \\
u : \mathbb{N}x, v : \mathbb{N}y & \vdash_{\mathbb{E}(\mathsf{ad})} & \mathbb{C}_2^2 : \mathcal{K} \subseteq \mathcal{K}^{s}
\end{array}$$

Therefore by (μE), we have $u : \mathbb{N}x, v : \mathbb{N}y \vdash_{\mathbb{E}(\mathsf{ad})} \mathsf{It}_2(\lambda w.u, \mathbb{C}_2^2, v) : \mathcal{K}y$, and finally

$$\vdash_{\mathbb{E}(\mathsf{ad})} \lambda u.\lambda v.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda w.u, \mathbb{C}_2^2, v) : \forall x \forall y.\mathbb{N}x, \mathbb{N}y \to \mathbb{N}\mathsf{ad}(x, y)$$

Now we can simplify the first equation defining $0 := 0_{\mathsf{g}}\star$, in this way the program $\mathsf{ad}^{\mathcal{M}} := \overline{\mathsf{ad}} := \lambda u.\lambda v.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda w.u, \mathbb{C}_2^2, v)$ computes the sum of two naturals given by:

$\mathsf{ad}(x, 0) = x$

$\mathsf{ad}(x, sy) = s(\mathsf{ad}(x, y))$

We have for any terms t, r:

$\overline{\mathsf{ad}}\, t\, 0^{\mathfrak{M}} \to_{\beta}^{+} t.$

$\overline{\mathsf{ad}}\, t\, (s^{\mathfrak{M}} r) \to_{\beta}^{+} s^{\mathfrak{M}}(\overline{\mathsf{ad}}\, t\, r)$

Multiplication of natural numbers

We need to define a binary function pd such that

- $\mathsf{pd}(x, 0_{\mathsf{g}}y) = 0_{\mathsf{g}}y$
- $\mathsf{pd}(x, sy) = \mathsf{ad}(\mathsf{pd}(x, y), x)$

We can prove that:

$$\vdash_{\mathbb{E}(\mathsf{pd})} \lambda u.\lambda v.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \overline{0_{\mathsf{g}}}, \lambda w.\overline{\mathsf{ad}}wu, v) : \forall x \forall y.\mathbb{N}x, \mathbb{N}y \to \mathbb{N}\mathsf{pd}(x, y)$$

Therefore the term $\overline{\mathsf{pd}} := \lambda u.\lambda v.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \overline{0_{\mathsf{g}}}, \lambda w.\overline{\mathsf{ad}}wu, v)$ is a program for the product.

The Predecessor

We need to define a unary function pred such that

- $\mathsf{pred}(0_{\mathsf{g}}y) = 0_{\mathsf{g}}y$
- $\mathsf{pred}(sy) = y$

The program is obtained from:

$$\vdash_{\mathbb{E}(\mathsf{p})} \lambda u.\mathsf{Rec}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda v.\overline{0_{\mathsf{g}}}, \lambda v.\pi_1 v, u) : \forall x.\mathbb{N}x \to \mathbb{N}\mathsf{p}x$$

The Factorial

We need to define a unary function fac such that

- $\mathsf{fac}(0_{\mathsf{g}}y) = s(0_{\mathsf{g}}y)$
- $\mathsf{fac}(sy) = \mathsf{pd}(sy, \mathsf{fac}(y))$

The program is obtained from:

$$\vdash_{\mathbb{E}(\mathsf{fac})} \lambda u.\mathsf{Rec}_2\Big(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{id}}, \lambda v.\overline{s}\overline{0_{\mathsf{g}}}, \lambda w.\overline{\mathsf{pd}}(\overline{s}(\pi_1 w))(\pi_2 w), u\Big) : \forall x.\mathbb{N}x \to \mathbb{N}\mathsf{fac}x$$

The Length of a List

We need to define a unary function len such that

- $\mathsf{len}(\mathsf{nil}_{\mathsf{g}}z) = 0_{\mathsf{g}}z$
- $\mathsf{len}(\mathsf{cons}z) = s(\mathsf{len}(\mathbb{\pi}_2\, z))$

We obtain:

$$\vdash_{\mathbb{E}(\mathsf{len})} \lambda x.\mathsf{It}_2(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{A \times X}, \overline{0_{\mathsf{g}}}, \lambda z.\overline{s}(\overline{\mathbb{\pi}_2}z), x) : \forall x.\mathcal{L}_{\mathcal{A}}x \to \mathbb{N}\mathsf{len}x$$

where $\mathbb{M}_{A \times X} := \lambda f \lambda x.\langle\!|\, \mathsf{out}_{2,1}\, x, f(\mathsf{out}_{2,2}\, x)|\!\rangle$, with $\langle\!| r, s |\!\rangle := \mathsf{out}_2^{-1}\left(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathsf{triv}}, r, s\right)$ and $\vdash^{\mathsf{can}} \mathbb{M}_{A \times X} : (A \times X)\,\mathsf{mon}\,X$.

Append of Lists

We need to define a binary function app such that

- $\circ$ $\mathsf{append}\langle \mathsf{nil}_{\mathsf{g}} z, y\rangle = y$
- $\circ$ $\mathsf{append}\langle \mathsf{cons}\, z, y\rangle = \mathsf{cons}\langle \pi_1\, z, \mathsf{append}\langle \pi_2\, z, y\rangle\rangle$

We cannot program this function directly because it has neither an inductive domain nor a coinductive codomain. Instead we will program the curried version:

- $\circ$ $\mathsf{append}(\mathsf{nil}_{\mathsf{g}} z)\; y = y$
- $\circ$ $\mathsf{append}\,\mathsf{cons}\, z\; y = \mathsf{cons}\langle \pi_1\, z, \mathsf{append}(\pi_2\, z)\; y\rangle$

We can prove that:

$$\vdash_{\mathbb{E}(\mathsf{app})} \quad \begin{array}{l} \lambda x.\mathsf{It}_2\Big(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{A\times X}, \lambda u\lambda z.z, \lambda u\lambda z.\overline{\mathsf{cons}}\llangle \overline{\pi_1} u, (\overline{\pi_2} u) y\rrangle, x\Big) : \\ \forall x\forall y.\mathcal{L}_{\mathcal{A}} x, \mathcal{L}_{\mathcal{A}} y \to \mathcal{L}_{\mathcal{A}}\,\mathsf{append}\; x\, y \end{array}$$

Reverse of a List

We need to define a unary function rev such that

- $\circ$ $\mathsf{rev}\,\mathsf{nil}_{\mathsf{g}} z = \mathsf{nil}_{\mathsf{g}} z$
- $\circ$ $\mathsf{rev}\,\mathsf{cons}\, z = \mathsf{append}(\mathsf{rev}\,\pi_2\, z)\;\mathsf{cons}\langle \pi_1\, z, \mathsf{nil}\rangle$

We can prove that:

$$\vdash_{\mathbb{E}(\mathsf{rev})} \quad \begin{array}{l} \lambda x.\mathsf{It}_2\Big(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{A\times X}, \mathsf{nil}_{\mathsf{g}}, \lambda z.\overline{\mathsf{append}}(\pi_2\, z)\;\overline{\mathsf{cons}}\llangle\, \pi_1\, z, \mathsf{nil}_{\mathsf{g}}\rrangle, x\Big) : \\ \forall x.\mathcal{L}_{\mathcal{A}} x \to \mathcal{L}_{\mathcal{A}}\mathsf{rev}\, x \end{array}$$

Filter

Given a unary predicate $\mathcal{P}$ over a data type $\mathcal{A}$ the function $\mathsf{filter}_{\mathcal{P}}$ from $\mathcal{L}_{\mathcal{A}}$ to $\mathcal{L}_{\mathcal{A}}$ such that

$$\begin{array}{rcl} \mathsf{filter}_{\mathcal{P}}\,\mathsf{nil} & := & \mathsf{nil} \\ \mathsf{filter}_{\mathcal{P}}\,\mathsf{cons}(x, l) & := & \mathsf{if}\ \mathcal{P}x\ \mathsf{then}\ \mathsf{cons}(x, \mathsf{filter}_{\mathcal{P}}\, l)\ \mathsf{else}\ \mathsf{filter}_{\mathcal{P}}\, l \end{array}$$

We represent a predicate $\mathcal{P}$ as a function $p : \mathcal{A} \to \mathbb{B}$ and define a curried version:

$$\mathsf{filter} : (\mathcal{A} \to \mathbb{B}) \to \mathcal{L}_{\mathcal{A}} \to \mathcal{L}_{\mathcal{A}}$$

such that

$$\begin{aligned}\mathsf{filter}\, p\ \mathsf{nil}_{\mathsf{g}} w &= \mathsf{nil}_{\mathsf{g}} w\\[1em] \mathsf{filter}\, p\ \ \mathsf{cons}\, w &= \mathsf{if}\ p(\pi_1\, w)\ \mathsf{then}\ \ \mathsf{cons}\langle \pi_1\, w, \mathsf{filter}\, p(\pi_2\, w)\rangle\ \mathsf{else}\ \ \mathsf{filter}\, p(\pi_2\, w)\end{aligned}$$

The following holds:

$$\begin{aligned}\vdash\quad & \lambda x\lambda y.\mathsf{It}_2\big(\mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{A\times X}, \mathsf{nil}_{\mathsf{g}}, \lambda z.\overline{\mathsf{if}}\big(x(\overline{\pi_1}z), \overline{\mathsf{cons}}\llparenthesis\, \pi_1\, z, \pi_2\, z\rrparenthesis, \pi_2\, z\big), y\big) :\\ & \forall p\forall y.(\mathcal{A}\rightarrow \mathbb{B})p, \mathcal{L}_{\mathcal{A}}, y \rightarrow \mathcal{L}_{\mathcal{A}}\, \mathsf{filter}\, p\, y\end{aligned}$$

Quicksort

Given an order $\leq$ over a data type $\mathcal{A}$ we define the quicksort operation from $\mathcal{L}_{\mathcal{A}}$ to $\mathcal{L}_{\mathcal{A}}$ as follows:

$$\begin{aligned}\mathsf{quicksort}\,\mathsf{nil} &= \mathsf{nil}\\[1em] \mathsf{quicksort}\,\mathsf{cons}(x,l) &= \mathsf{append}\,\Big(\mathsf{append}\,\big((\mathsf{filter}\ \leq_x\ \ \mathsf{quicksort}\, l)\ [x]\big),\\ &\qquad\qquad (\mathsf{filter}\ >_x\ \ \mathsf{quicksort}\, l)\Big)\end{aligned}$$

We program a curried version as follows: we represent the relation $\leq$ as a function $F_{\leq}$ such that $x\leq y$ holds if and only if $F_{\leq}xy = \mathsf{true}$. Analogously we also have a function $F_{>}$. Now given a x such that $\mathcal{A}x$ hold we define the functions $F_{\leq x} := \lambda y.F_{\leq}yx$ and $F_{>x} := \lambda y.F_{>}yx$. The quicksort operation is defined as follows:

$$\begin{aligned}\mathsf{quicksort}\,\mathsf{nil}_{\mathsf{g}} w &= \mathsf{nil}_{\mathsf{g}} w\\[1em] \mathsf{quicksort}\,\mathsf{cons}\, w &= \mathsf{append}\,\Big(\mathsf{append}\,\big((\mathsf{filter}\, F_{\leq\pi_1\, w}\ \ \mathsf{quicksort}\,\pi_2\, w)\ [\pi_1\, w]\big),\\ &\qquad\qquad (\mathsf{filter}\, F_{>\pi_1\, w}\ \ \mathsf{quicksort}\,\pi_2\, w)\Big)\end{aligned}$$

where $[\pi_1\, w] := \mathsf{cons}\langle\pi_1\, w, \mathsf{nil}\rangle$ is the list which unique element is $\pi_1\, w$

To get a program for $\mathsf{quicksort}$ we assume that the functions $F_{\leq}, F_{<}$ are computable by programs $\overline{F}_{\leq}, \overline{F}_{>}$, such that $\vdash\ \overline{F}_{\leq}\ :\ \forall x\forall y.\mathcal{A}x, \mathcal{A}y\ \rightarrow\ \mathbb{B}\, F_{\leq}xy$ and $\vdash \overline{F}_{>} : \forall x\forall y.\mathcal{A}x, \mathcal{A}y \rightarrow \mathbb{B}\, F_x > y$. From these programs we get the needed programs $\overline{F}_{\leq x} := \lambda y.\overline{F}_{\leq}yx$ and $\overline{F}_{>x} := \lambda y.\overline{F}_{>}yx$ such that

$$\vdash \overline{F}_{\leq x} : (\mathcal{A}\rightarrow\mathbb{B})F_{\leq x}\qquad \vdash \overline{F}_{>x} : (\mathcal{A}\rightarrow\mathbb{B})F_{>x}$$

Finally a program for $\mathsf{quicksort}$ is:

$$\begin{aligned}\overline{\mathsf{quicksort}} := \lambda x.\mathsf{It}_2\Big(\quad & \mathbb{M}_{\mathsf{triv}}, \mathbb{M}_{\mathcal{A}\times X}, \overline{\mathsf{nil}}_{\mathsf{g}},\\ & \lambda y.\overline{\mathsf{append}}\ \Big(\overline{\mathsf{append}}\ (\overline{\mathsf{filter}}\ \overline{F}_{\leq(\overline{\pi_1}y)}\ \overline{\pi_2}y)\ \overline{\mathsf{cons}}\ \llparenthesis\overline{\pi_1}y, \overline{\mathsf{nil}}\rrparenthesis\Big)\\ & \qquad\qquad\quad \big(\overline{\mathsf{filter}}\ \overline{F}_{>\overline{\pi_1}y}\ \overline{\pi_2}y\big)\\ & x\Big)\end{aligned}$$

5.3.2 Programming Functions with Coiteration or Corecursion

As we have seen in some of the examples presented in the previous section, the programs obtained via the derivations in the logic coincide with those obtained in the type system using the categorical point of view. This does not seem surprising as the logic was designed having in mind the Curry-Howard correspondence.

Let us analize the particular case for programming unary functions (other cases can be solved by currying). The goal is to obtain derivations of the form

$$\vdash t : \forall x.\mathcal{D}x \to \mathcal{E}fx$$

The cases where $\mathcal{D}$ is an inductive data type, say $\mathcal{D} := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$, have not possesed a problem. The obvious choice is to use iteration or recursion with the predicate $\mathcal{K} := \lambda x.\mathcal{E}fx$.
For iteration, the goal is to obtain the following:

$$\frac{\begin{array}{c} x : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \vdash x : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \\ x : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \vdash m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ x : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \vdash s_i : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\vec{\mathbb{c}_i}},\ \ 1 \leq i \leq k \end{array}}{x : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \vdash \mathsf{It}_k(\vec{m}, \vec{s}, x) : \mathcal{K}\vec{x}}$$

which yields

$$\vdash \lambda x.\mathsf{It}_k(\vec{m}, \vec{s}, x) : \forall x.\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)x \to \mathcal{E}fx$$

This kind of proofs are quite easily achieved if $\mathcal{E}$ is again an inductive data type and although it could work in some cases where $\mathcal{E}$ is coinductive the natural thing for that case would be to use coiteration/corecursion. For these cases the goal is to get derivations of the form:

$$\vdash t : \forall x.\mathcal{D}x \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)fx$$

Using coiteration/corecursion, the obvious choice will be to get:

$$\frac{\begin{array}{l} x : \mathcal{D}x \vdash x : \mathcal{K}(fx) \\ x : \mathcal{D}x \vdash m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ x : \mathcal{D}x \vdash s_i : \mathcal{K} \subseteq \mathcal{F}_i[X := \mathcal{K}]^{\vec{\mathbb{c}_i}},\ \ 1 \leq i \leq k \end{array}}{x : \mathcal{D}x \vdash \mathsf{CoIt}_k(\vec{m}, \vec{s}, x) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)(fx)}$$

which allows to conclude

$$\vdash \lambda x.\mathsf{CoIt}_k(\vec{m}, \vec{s}, x) : \forall x.\mathcal{D}x \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)fx$$

But in this case there is no obvious choice for $\mathcal{K}$ such that the derivation $x : \mathcal{D}x \vdash x : \mathcal{K}(fx)$ holds. Moreover the restriction given by the proof-term x leave us with very few possibilities.

We could also add some features to the logic, like restricted formulas, that would help to obtain some examples but we do not see a general pattern to obtain the desired programs.

The example which led us to find this problem was the from function which takes a natural number and returns the stream of naturals starting in the given number. This function is destructed as

$$\begin{array}{rcl}\mathsf{head}\,\mathsf{from}\,x & = & x\\ \mathsf{tail}\,\mathsf{from}\,x & = & \mathsf{from}\,sx\end{array}$$

This function can be easily programmed within the term system MCICT (see page 73), which give us a clue about the proof-term we are looking for.

Let us forget for a moment that we were not able to prove that the streams are a formal data type and try to program the function from. The goal is to get a term $\overline{\mathsf{from}}$ such that $\vdash \overline{\mathsf{from}} : \forall x.\mathbb{N}x \to \mathcal{S}_{\mathbb{N}}\,\mathsf{from}\,x$. We would like to derive the premisses of the following instance of (νI):

$$\frac{\begin{array}{c}x : \mathbb{N}x \vdash x : \mathcal{K}\,\mathsf{from}\,x\\ x : \mathbb{N}x \vdash m_1 : \mathbb{N}\mathsf{mon}X\\ x : \mathbb{N}x \vdash m_2 : X\;\mathsf{mon}\,X\\ x : \mathbb{N}x \vdash s_1 : \mathcal{K} \subseteq \mathbb{N}^{\mathsf{head}}\\ x : \mathbb{N}x \vdash s_2 : \mathcal{K} \subseteq \mathcal{K}^{\mathsf{tail}}\end{array}}{x : \mathbb{N}x \vdash \mathsf{Colt}_k(\vec{m}, \vec{s}, x) : \mathcal{S}_{\mathbb{N}}\,\mathsf{from}\,x}$$

The first premisse restricts us to take $\mathcal{K} := \lambda x.\mathbb{N}\;\mathsf{head}\,x$, so that $\mathcal{K}\,\mathsf{from}\,x \equiv \mathbb{N}\;\mathsf{head}\,\mathsf{from}\,x \equiv \mathbb{N}x$. The monotonicity proofs are trivially accomplishable, as well as the first step contention for which we have $x : \mathbb{N}x \vdash \lambda zz : \mathcal{K} \subseteq \mathbb{N}^{\mathsf{head}}$. The problem arises when trying to prove the last premisse:

$$x : \mathbb{N}x \vdash ? : \forall y.\mathbb{N}\;\mathsf{head}\,y \to \mathbb{N}\;\mathsf{head}\,\mathsf{tail}\,y$$

The proof in the term system indicates us that the proof-term should be a program for the succesor function $\overline{s}$, which obviously lead us to get

$$x : \mathbb{N}x \vdash \overline{s} : \forall y.\mathbb{N}\;\mathsf{head}\,y \to \mathbb{N}\;\mathsf{head}\,\mathsf{tail}\,\mathsf{from}\,\mathsf{head}\,y$$

from the equations $s(\mathsf{head}\,y) = \mathsf{head}\,\mathsf{from}\,s(\mathsf{head}\,y) = \mathsf{head}\,\mathsf{tail}\,\mathsf{from}\,\mathsf{head}\,y$.

This is the best we can do as to get $\mathbb{N}\;\mathsf{head}\,\mathsf{tail}\,y$ the obvious way would be to get $\mathcal{S}_{\mathbb{N}}\;\mathsf{tail}\,y$ and then apply a coclosure axiom, but the only fact we now about y is that $\mathbb{N}\;\mathsf{head}\,y$ and from this we will never get the required $\mathcal{S}_{\mathbb{N}}\;\mathsf{tail}\,y$. For to get $\mathcal{S}_{\mathbb{N}}\;\mathsf{tail}\,y$ either we get $\mathcal{S}_{\mathbb{N}}y$ and apply a coclosure axiom or we try to prove $\mathbb{N}\;\mathsf{head}\,\mathsf{tail}\,y$ and $\mathcal{S}_{\mathbb{N}}\;\mathsf{tail}\,\mathsf{tail}\,y$ and use inversion, both tasks are not derivable from the only premisse $\mathbb{N}\;\mathsf{head}\,y$.

If we see how easy was to obtain the program directly in the type system, we realize that the problem here is caused by the first-order objects. In that case the second step function is only $\rhd s : \mathsf{nat} \to \mathsf{nat}$, the type is the same on both sides of the arrow. On the other hand in the logic the first order-objects cause to have different predicates on both sides of the inclusion symbol, namely $\mathcal{K}$ and $\mathcal{K}^{\mathsf{tail}}$. I like to refer to this kind of problems as the evilness of first-order objects for conventional coinduction.

A similar problem was detected when trying to prove the streams of succesors of a stream of natural numbers in [Tat93], where a tailor-made quite complex solution was provided.

Fortunately we can get rid of this evilness by defining an alternative system which allows to do some easy programming with coinductive data types, namely a system including coiteration and corecursion principles in the sense of Mendler. Next chapter is devoted to this question.

-Hace calor aquí -dije.
-Sí, y esto no es nada me contestó el otro-. Cálmese. Ya lo sentirá más fuerte cuando lleguemos a Comala. Aquello está sobre las brasas de la tierra, en la mera boca del infierno. Con decirle que muchos de los que allí se mueren, al llegar al infierno regresan por su cobija.

Juan Rulfo, Pedro Paramo.

6 A System with Mendler-style Coinduction

At the end of the last chapter we have seen that the principles of coiteration/corecursion in MCICD are not useful to program functions into coinductive predicates. In this chapter we give a solution based on Mendler's approach.

6.1 Fixed-Point Theory

As we will see later, the Mendler-style coinduction principles are related to the construction of the greatest-fixed point by means of transfinite induction, process that we recall here.

Definition 6.1 *Given a monotone operator $\Gamma : \mathcal{P}(A) \to \mathcal{P}(A)$. The downward or greatest fixed point hierarchy of Γ consists of the sequence of sets $\Gamma^{\downarrow}_{\alpha}$ defined by transfinite recursion as follows:*

$$\begin{array}{rcl} \Gamma^{\downarrow}_{0} & := & A \\ \Gamma^{\downarrow}_{\alpha+1} & := & \Gamma(\Gamma^{\downarrow}_{\alpha}) \\ \Gamma^{\downarrow}_{\lambda} & := & \bigcap_{\alpha\leq\lambda}\Gamma^{\downarrow}_{\alpha} \ \textit{with } \lambda \textit{ a limit ordinal} \end{array}$$

Analogously the upward or least fixed point hierarchy of Γ is the sequence $\left(\Gamma^{\uparrow}_{\alpha}\right)_{\alpha\in\mathsf{On}}$

defined as:

$$\begin{aligned} \Gamma^{\uparrow}_{0} &:= \varnothing \\ \Gamma^{\uparrow}_{\alpha+1} &:= \Gamma(\Gamma^{\uparrow}_{\alpha}) \\ \Gamma^{\uparrow}_{\lambda} &:= \textstyle\bigcup_{\alpha\leq\lambda}\Gamma^{\uparrow}_{\alpha} \ \textit{with } \lambda \textit{ a limit ordinal} \end{aligned}$$

The following fact is easy to prove.

Proposition 6.1 *Let* $\Gamma : \mathcal{P}(A) \to \mathcal{P}(A)$ *be a monotone operator. Set* $\Gamma^{\downarrow} := \bigcap_{\alpha\in\mathsf{On}}\Gamma^{\downarrow}_{\alpha}$ *and* $\Gamma^{\uparrow} := \bigcup_{\alpha\in\mathsf{On}}\Gamma^{\uparrow}_{\alpha}$. *Then* $\Gamma^{\uparrow}$ *is the least fixed point of* Γ *and* $\Gamma^{\downarrow}$ *is the greatest fixed point of* Γ.

6.2 The Logic $\mathsf{MCICD}_{\mu M\nu}$

This system is obtained by eliminating the conventional coinduction principles of MCICD and introducing instead Mendler-style coinduction principles. Monotonicity is not needed to formulate these principles, however our choice of semantics will require a syntactical restriction to build coinductive predicates. The necessity for this condition, which we call admissibility, is made clear in the proof of lemma 6.1

Definition 6.2 *Given a formula* F *and a second order variable* X *we define the relation "*X *is admissible in* F*" denoted* $X\,\mathsf{admis}\,F$ *as follows:*

- $X\,\mathsf{admis}\,X\vec{t}$
- *If* $X \notin FV(F)$ *then* $X\,\mathsf{admis}\,F$.
- *If* $X \notin FV(G)$ *and* $X\,\mathsf{admis}\,H$ *then* $X\,\mathsf{admis}\,G \to H$.
- *If* $X\,\mathsf{admis}\,F$ *then* $X\,\mathsf{admis}\,\forall xF$.
- *If* $X\,\mathsf{admis}\,F$ *then* $X\,\mathsf{admis}\,\forall YF$.
- *If* $X\,\mathsf{admis}\,G$ *and* $X\,\mathsf{admis}\,H$ *then* $X\,\mathsf{admis}\,G \wedge H$.
- *If* $X\,\mathsf{admis}\,F_i$ *and* $Z\,\mathsf{admis}\,F_i$ *then* $X\,\mathsf{admis}\,\mu Z(\mathcal{C}_1,\ldots,\mathcal{C}_k)\vec{t}$
- *If* $X\,\mathsf{admis}\,G_i$ *and* $Z\,\mathsf{admis}\,G_i$ *then* $X\,\mathsf{admis}\,\nu Z(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{t}$

where in the last two cases $\mathcal{C}_i := \langle\lambda\vec{y}.F_i, \vec{\mathfrak{c}_i}\rangle$, $\mathcal{D}_i := \langle\lambda\vec{y}.G_i, \vec{\mathfrak{c}_i}\rangle$.
If $\mathcal{F} := \lambda\vec{y}.F$ *then we define* $X\,\mathsf{admis}\,\mathcal{F} := X\,\mathsf{admis}\,F$.

Observe that the essential difference with the definition of strict positivity is the case for (co)inductive predicates.

Proposition 6.2 *If* $X\,\mathsf{admis}\,\mathcal{F}$ *then* $\vdash \mathcal{F}\,\mathsf{mon}\,X$.
Proof. Induction on F, where $\mathcal{F} := \lambda\vec{y}F$. ⊣

Proposition 6.3 *If* $X \,\mathsf{admis}\, \mathcal{F}$ *then* $\models \mathcal{F} \,\mathsf{mon}\, X$.
Proof. Analogous to the previous proposition. ⊣

We define the new system by eliminating disjunctions from MCICD and replacing two introduction rules for coinductive predicates (those for coiteration and corecursion) with the following rules:

$$\frac{\Gamma \vdash s_i : \forall X.\big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{t}\big) \to \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}}_i}\vec{t}\big),\ \ 1 \le i \le k}{\Gamma \vdash \mathsf{MCoIt}_k\vec{s} : \forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{t}} \ (M\nu I)$$

$$\frac{\begin{array}{ll}\Gamma \vdash s_i : & \forall X.\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k) \subseteq X \to \\ & \quad \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{t}\big) \to \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}}_i}\vec{t}\big),\ \ 1 \le i \le k\end{array}}{\Gamma \vdash \mathsf{MCoRec}_k\vec{s} : \forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{t}} \ (M\nu I^+)$$

Both rules with the proviso $X \notin FV(\Gamma, \mathcal{K})$ and $X \,\mathsf{admis}\, \mathcal{F}_i$ for $1 \le i \le k$. These rules express Mendler-style coiteration and corecursion respectively (see [Men87, Men91]).
The intuition behind the rule for coiteration can be explained as follows: looking at the construction of the greatest fixed point by transfinite recursion given in section 6.1 we see that the coinductive predicate $\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)$ can be intuitively "defined" as the infinite intersection (conjunction)

$$\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k) := \bigwedge_{\alpha \in \mathsf{On}} \mathcal{G}_\alpha,$$

where $\mathcal{G}_\alpha := \mathcal{G}^\alpha(\top)$ (with $\top$ the true predicate) and $\mathcal{G} := \mathcal{F}_1^{\vec{\mathbb{c}}_1} \wedge \ldots \wedge \mathcal{F}_n^{\vec{\mathbb{c}}_n}$. In this way a formula of the form $\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{t}$ can be obtained by constructing "approximations" $\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{G}_\alpha\vec{t}$ for every $\alpha \in \mathsf{On}$.
What the premisses of the rule $(M\nu I)$ ensure is the construction of a formula

$$(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{G}_\alpha\vec{t}\,) \to (\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{G}_{\alpha+1}\vec{t}\,).$$

Now observe that the case for $\alpha = 0$ is trivially provable as $\mathcal{G}_0 = \top$. Therefore the last formula guarantees the existence of every approximation $\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{G}_\alpha\vec{t}$. As this process cannot be justified syntactically, we do it semantically in lemma 6.2 below. A justification for the rule $(M\nu I^+)$ is similar but this time the premisses guarantee the construction of approximations only if we start with a set which already includes the coinductively defined set.

By dualizing we can get similar rules for inductive predicates. For explanations on Mendler-style induction/recursion from this point of view see [Urz99].

The proof-reduction behaviour is given by:

$$\begin{array}{rcl}\mathsf{out}_{k,i}\big(\mathsf{MCoIt}_k\vec{s}\,r\big) & \mapsto_\beta & s_i\big(\mathsf{MCoIt}_k\vec{s}\big)r \\ \mathsf{out}_{k,i}\big(\mathsf{MCoRec}_k\vec{s}\,r\big) & \mapsto_\beta & s_i(\lambda y.y)\big(\mathsf{MCoRec}_k\vec{s}\big)r\end{array}$$

The just described logical system will be called $\mathsf{MCICD}_{\boldsymbol{\mu M \nu}}$.

The rules for elimination of coinductive predicates generate the following axioms.

Definition 6.3 *The Mendler-style coinduction axioms are:*

$$\begin{aligned}
\mathsf{MCoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} := \forall X \forall \vec{z}.\quad & \Big(\big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\,\big) \to \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{z}\,\big)\Big) \\
& \to \Big(\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{z}\,\Big)
\end{aligned}$$

$$\begin{aligned}
\mathsf{MCoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} := \forall X \forall \vec{z}.\quad & \Big(\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \subseteq X \to \\
& \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\,\big) \to \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{z}\,\big)\Big) \\
& \to \Big(\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{z}\,\Big)
\end{aligned}$$

Subject Reduction and Strong Normalization

Subject reduction can be proved by the method of section 4.1.3, indeed the proof is simpler.
By forgetting first-order objects we get an embedding of this logic into $\mathsf{MCICT}_{\boldsymbol{\mu M \nu}}$ (see page 77), which proves strong normalization.

6.3 Realizability for $\mathsf{MCICD}_{\boldsymbol{\mu M \nu}}$

As in section 4 the realizability interpretation of $\mathsf{MCICD}_{\boldsymbol{\mu M \nu}}$ will be given into an extended system $\mathsf{MCICD}^{\star}_{\boldsymbol{\mu M \nu}}$ which is the same system but over the term system $\mathsf{MCICT}_{\boldsymbol{\mu M \nu}}$. This time we do not need an extension with existential and restricted formulas, because the system does not have disjunctions.
The definition of realizability, which gives $\mathsf{MCICD}^{\star}_{\boldsymbol{\mu M \nu}}$-formulas $t \;\mathtt{r}\; A$ where t is a $\mathsf{MCICT}_{\boldsymbol{\mu M \nu}}$-term and A is a $\mathsf{MCICD}_{\boldsymbol{\mu M \nu}}$-formula, is just definition 4.6 leaving out the case for disjunctions which simplifies the target logic considerably, we do not need neither existential nor restricted formulas. Therefore the only difference now between source and target logics is the underlying type system.

6.3.1 Realizing the Axioms

Proposition 6.4 *Let* $\mathbb{K} := \lambda\vec{z}.\mathsf{MCoIt}_k\vec{z}$ *and* $\mathbb{Q} := \lambda\vec{z}.\mathsf{MCoRec}_k\vec{z}$. *Then*

(i) $\vdash \lambda\vec{y}.\mathsf{MCoIt}_k\vec{y} : \mathbb{K} \;\mathtt{r}\; \mathsf{MCoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

(ii) $\vdash \lambda\vec{y}.\mathsf{MCoRec}_k\vec{y} : \mathbb{Q} \;\mathtt{r}\; \mathsf{MCoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

Proof. We prove part (ii), the first part is easier.
We need to proof $\mathbb{Q}\ \mathtt{r}\ \mathsf{MCoInd}^{+}_{\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)}$, that is

$$\begin{aligned}\mathbb{Q}\ \mathtt{r}\ \forall X \forall \vec{z}.\quad & \Big(\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k) \subseteq X \to \\ & \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\,\big) \to \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{z}\,\big)\Big) \\ & \to \Big(\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{z}\,\Big)\end{aligned}$$

which unfolds to

$$\begin{aligned}\forall X^{+} \forall \vec{z} \forall \vec{f}.\quad & \Big(\forall y \forall w. y\ \mathtt{r}\ \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k) \subseteq X \to w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\big) \\ & \to f_i y w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{z}\,\big)\Big) \\ & \to \mathbb{Q}\vec{f}\ \mathtt{r}\ \Big(\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{z}\,\Big)\end{aligned}$$

Set

$$\begin{aligned}\Gamma := \Big\{ y_i : \forall y \forall w.\quad & y\ \mathtt{r}\ \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k) \subseteq X \to w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\big) \\ & \to f_i y w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{z}\,\big)\Big\}\end{aligned}$$

The goal is

$$\Gamma \vdash \mathsf{MCoRec}_k \vec{y} : \mathbb{Q}\vec{f}\ \mathtt{r}\ \Big(\forall \vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{z}\,\Big) \tag{6.1}$$

Now observe that

$$\begin{aligned} y\ \mathtt{r}\ \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k) \subseteq X \equiv\ & \forall \vec{x} \forall z. z\ \mathtt{r}\ \nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)\vec{x} \to yz\ \mathtt{r}\ X\vec{x} \\ \equiv\ & \forall \vec{x} \forall z. \nu X^{+}(\mathcal{D}_1^{\mathtt{r}},\dots,\mathcal{D}_k^{\mathtt{r}})\vec{x}z \to X^{+}\vec{x}(yz) \\[1ex] w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{z}\big) \equiv\ & \forall \vec{x} \forall z. z\ \mathtt{r}\ \mathcal{K}\vec{x} \to wz\ \mathtt{r}\ X\vec{z} \\ \equiv\ & \forall \vec{x} \forall z. \mathcal{K}^{\mathtt{r}}\vec{x}z \to X^{+}\vec{z}(wz) \\[1ex] f_i y w\ \mathtt{r}\ \big(\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{z}\,\big) \equiv\ & \forall \vec{x} \forall z. z\ \mathtt{r}\ \mathcal{K}\vec{x} \to f_i y w z\ \mathtt{r}\ \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{z} \\ \equiv\ & \forall \vec{x} \forall z. \mathcal{K}^{\mathtt{r}}\vec{x}z \to \mathcal{F}_i^{\mathtt{r}}(\vec{\mathbb{c}_i}\vec{z})(f_i y w z)\end{aligned}$$

Therefore instantiating $y := \lambda uu,\ w := \mathsf{MCoRec}_k \vec{f}$ we get

$$\begin{aligned}\Gamma \vdash y_i :\quad & \forall \vec{x} \forall z. \nu X^{+}(\mathcal{D}_1^{\mathtt{r}},\dots,\mathcal{D}_k^{\mathtt{r}})\vec{x}z \to X^{+}\vec{x}\big((\lambda uu)z\big) \to \\ & \forall \vec{x} \forall z. \mathcal{K}^{\mathtt{r}}\vec{x}z \to X^{+}\vec{z}\big((\mathsf{MCoRec}_k \vec{f})z\big) \to \\ & \forall \vec{x} \forall z. \mathcal{K}^{\mathtt{r}}\vec{x}z \to \mathcal{F}_i^{\mathtt{r}}(\vec{\mathbb{c}_i}\vec{z})\big(f_i(\lambda uu)(\mathsf{MCoRec}_k \vec{f})z\big)\end{aligned}$$

but we have both $(\lambda uu)z = z \in \mathbb{E}_\beta$ and

$$f_i(\lambda uu)(\mathsf{MCoRec}_k \vec{f})z = \mathsf{out}_{k,i},\big(\mathsf{MCoRec}_k \vec{f} z\big) = \mathbb{D}_i^k(\mathsf{MCoRec}_k \vec{f} z) \in \mathbb{E}_\beta$$

and simplifying we get

$$\begin{array}{ll}\Gamma \vdash y_i : & \forall\vec{x}\forall z.\nu X^{+}(\mathcal{D}^{\mathtt{r}}_1,\ldots,\mathcal{D}^{\mathtt{r}}_k)\vec{x}z \to X^{+}\vec{x}z \to \\ & \forall\vec{x}\forall z.\mathcal{K}^{\mathtt{r}}\vec{x}z \to X^{+}\vec{z}\big((\mathsf{MCoRec}_k\vec{f})z\big) \to \\ & \forall\vec{x}\forall z.\mathcal{K}^{\mathtt{r}}\vec{x}z \to \mathcal{F}^{\mathtt{r}}_i(\vec{\mathbb{c}_i}\vec{z})\big(\mathbb{D}^k_i(\mathsf{MCoRec}_k\vec{f}z)\big),\end{array}$$

that is,

$$\begin{array}{ll}\Gamma \vdash y_i : & \nu X^{+}(\mathcal{D}^{\mathtt{r}}_1,\ldots,\mathcal{D}^{\mathtt{r}}_k) \subseteq X^{+} \to \\ & \forall\vec{x}\forall z.\mathcal{K}^{\mathtt{r}}\vec{x}z \to X^{+}\vec{z}\big((\mathsf{MCoRec}_k\vec{f})z\big) \to \\ & \forall\vec{x}\forall z.\mathcal{K}^{\mathtt{r}}\vec{x}z \to \mathcal{F}^{\mathtt{r}}_i(\vec{\mathbb{c}_i}\vec{z})\big(\mathbb{D}^k_i(\mathsf{MCoRec}_k\vec{f}z)\big),\end{array}$$

Therefore the rule $(M\nu I^{+})$ yields

$$\Gamma \vdash \mathsf{MCoRec}_k\vec{y} : \forall\vec{x}\forall z.\mathcal{K}^{\mathtt{r}}\vec{x}z \to \nu X^{+}(\mathcal{D}^{\mathtt{r}}_1,\ldots,\mathcal{D}^{\mathtt{r}}_k)\vec{z}\big(\mathsf{MCoRec}_k\vec{f}z\big)$$

Now observing that $\mathbb{Q}\vec{f} = \mathsf{MCoRec}_k\vec{f} \in \mathbb{E}_\beta$ and using the definition of realizability we obtain

$$\Gamma \vdash \mathsf{MCoRec}_k\vec{y} : \forall\vec{x}\forall z.z\ \mathtt{r}\ \mathcal{K}\vec{x} \to \mathbb{Q}\vec{f}z\ \mathtt{r}\ \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{z}$$

But this is exactly derivation (6.1).

$\dashv$

Observe that in comparison to proposition 4.5 the proofs of realizability for the Mendler-style coinduction axioms are quite simple.

6.3.2 The Soundness Theorem

Definition 6.4 *Given an* $\mathsf{MCICD}_{\boldsymbol{\mu M\nu}}$*-proof-term r we define the* $\mathsf{MCICD}^{\star}_{\boldsymbol{\mu M\nu}}$*-proof-term $\widetilde{r}$ as follows:*

$$\begin{array}{rclrcl}\widetilde{x} & := & x & \widetilde{\lambda x.r} & := & \lambda x.\widetilde{r} \\ \widetilde{rs} & := & \widetilde{r}\,\widetilde{s} & \widetilde{\langle r,s\rangle} & := & \langle\widetilde{r},\widetilde{s}\,\rangle \\ \widetilde{\pi_1 r} & := & \pi_1\,\widetilde{r} & \widetilde{\pi_2 r} & := & \pi_2\,\widetilde{r}\end{array}$$

$$\begin{array}{rcl}\widetilde{\mathsf{in}_{k,i}\,t} & := & \mathsf{in}_{k,i}\,\widetilde{t} \\ \widetilde{\mathsf{It}_k(\vec{m},\vec{s},t)} & := & \mathsf{It}_k(\widetilde{\vec{m}},\vec{\mathsf{s}}[\vec{m},\vec{s}\,],\widetilde{t}\ ) \\ \widetilde{\mathsf{Rec}_k(\vec{m},\vec{s},t)} & := & \mathsf{Rec}_k(\widetilde{\vec{m}},\vec{\mathsf{s}}[\vec{m},\vec{s}\,],\widetilde{t}\ ) \\ \widetilde{\mathsf{out}_{k,i}\,t} & := & \mathsf{out}_{k,i}\,\widetilde{t} \\ \widetilde{\mathsf{MCoIt}_k\,\vec{t}} & := & \mathsf{MCoIt}_k\,\widetilde{\vec{t}} \\ \widetilde{\mathsf{MCoRec}_k\,\vec{t}} & := & \mathsf{MCoRec}_k\,\widetilde{\vec{t}} \\ \widetilde{\mathsf{out}_k^{-1}(\vec{m},\vec{s})} & := & \mathsf{out}_k^{-1}(\widetilde{\vec{m}},\vec{\mathsf{t}}[\vec{m},\vec{s}]\ )\end{array}$$

where in the cases for iteration and recursion, we have:

$$\begin{array}{rcl}\mathsf{s}[x,y] & := & \lambda u.\widetilde{y}(\widetilde{x}(\lambda v.v)u) \\ \mathsf{t}[x,y] & := & \widetilde{x}(\lambda zz)\widetilde{y}\end{array}$$

and we define $\vec{\mathsf{s}}[\vec{x},\vec{y}\,] := \mathsf{s}[x_1,y_1],\ldots,\mathsf{s}[x_k,y_k]$ *(the same for* t*).*

The definition 4.8 of $\mathcal{W}(s)$ is adapted accordingly and the definition 4.9 of $\mathbb{E}^\star(\vec{s}\,)$ remains the same.

Given a context $\Gamma = \{x_1 : A_1, \ldots, x_k : A_k\}$ we set

$$\Gamma^{\mathtt{r}} := \{x_1 : x_1 \mathtt{\ r\ } A_1, \ldots, x_k : x_k \mathtt{\ r\ } A_k\},$$

where w.l.o.g. $x_i \notin FV(A_i)$.

Theorem 6.1 (Soundness of Realizability for $\mathsf{MCICD}_{\boldsymbol{\mu M\nu}}$) *If* $\Gamma \vdash_{\mathsf{MCICD}_{\mu M\nu}} s : A$ *then* $\Gamma^{\mathtt{r}} \vdash_{\mathsf{MCICD}^\star_{\mu M\nu},\mathbb{E}^\star(s)} \widetilde{s} : s \mathtt{\ r\ } A$

Proof. Induction on $\vdash_{\mathsf{MCICD}_{\mu M\nu}}$. The cases for rules $(M\nu I), (M\nu I^+)$ are solved with the help of proposition 6.4. ⊣

6.4 Semantics

The main goal of this section is to prove the soundness of the logic $\mathsf{MCICD}_{\boldsymbol{\mu M\nu}}$ with respect to the same tarskian semantics given for $\mathsf{MCICD}^\star$. We will see that to be able to prove the validity of the Mendler-style coinduction axioms, we need some continuity property guaranteed to hold by the syntactic condition of admissibility.

Lemma 6.1 (Continuity Lemma) *Let* $\mathcal{F}$ *be a predicate such that* $X \,\mathsf{admis}\, \mathcal{F}$ *and* $\big(P_\alpha\big)_{\alpha\in\mathsf{On}}$ *a family of sets with* $\bigcap_{\alpha\in\mathsf{On}} P_\alpha \neq \varnothing$*. If* $\nu[X/P_\alpha][\vec{x}/\vec{r}\,] \models \mathcal{F}\vec{x}$ *for all* $\alpha \in \mathsf{On}$*, then*

$$\nu[X/\bigcap_{\alpha\in\mathsf{On}} P_\alpha][\vec{x}/\vec{r}\,] \models \mathcal{F}\vec{x}.$$

Informally $\bigcap_{\alpha\in\mathsf{On}} \mathcal{F}(P_\alpha) \subseteq \mathcal{F}\big(\bigcap_{\alpha\in\mathsf{On}} P_\alpha\big)$.

Proof. Induction on F, with $\mathcal{F} := \lambda\vec{y}F$, for every family of sets $(P_\alpha)_{\alpha\in\mathsf{On}}$ with $\bigcap P_\alpha \neq \varnothing$. Set $\mathcal{P} := \bigcap_{\alpha\in\mathsf{On}} P_\alpha$.
Case $\mathcal{F} := \lambda\vec{y}.\nu Z(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{t}.$ with $\mathcal{D}_i := \langle\mathcal{G}_i, \vec{\mathfrak{c}_i}\rangle$. As $X \,\mathsf{admis}\, \mathcal{F}$ we have also that $X \,\mathsf{admis}\, \mathcal{G}_i, Z \,\mathsf{admis}\, \mathcal{G}_i$.

By assumption we have $\nu[X/P_\alpha][\vec{x}/\vec{r}\,] \models \nu Z(\mathcal{D}_1,\ldots,\mathcal{D}_k)\vec{t}$, i.e.,

$$\nu[X/P_\alpha][\vec{x}/\vec{r}\,] \models \mathcal{G}_i \,\mathsf{mon}\, Z \wedge \exists Z.Z \subseteq \mathcal{G}_i^{\vec{\mathfrak{c}_i}} \wedge Z\vec{t}\,[\vec{y} := \vec{x}\,],\ \text{for all } \alpha \in \mathsf{On}$$

This implies that for all α there exists a set $\mathcal{Q}_\alpha$ such that

$$\nu[X/P_\alpha][\vec{x}/\vec{r}\,][Z/\mathcal{Q}_\alpha] \models Z \subseteq \mathcal{G}_i^{\vec{\mathfrak{c}_i}} \wedge Z\vec{t}\,[\vec{y} := \vec{x}\,]$$

The goal is

$$\nu[X/\mathcal{P}][\vec{x}/\vec{r}\,] \models \mathcal{G}_i \,\mathsf{mon}\, Z \wedge \exists Z.Z \subseteq \mathcal{G}_i^{\vec{\mathfrak{c}_i}} \wedge Z\vec{t}\,[\vec{y} := \vec{x}\,]$$

Set $\mathcal{Q} := \bigcap \mathcal{Q}_\alpha$. Clearly $\mathcal{Q} \neq \varnothing$.

- $\nu[X/\mathcal{P}][\vec{x}/\vec{r}] \models \mathcal{G}_i \,\mathsf{mon}\, Z$. Clear.
- $\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}] \models Z \subseteq \mathcal{G}_i^{\vec{\mathfrak{c}_i}}$
  Assume $\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}][\vec{y}/\vec{s}] \models Z\vec{y}$. This implies

  $$\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}_\alpha][\vec{y}/\vec{s} \models Z\vec{y}, \text{ for all } \alpha,$$

  which by the previous assumption implies

  $$\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}_\alpha][\vec{y}/\vec{s}] \models \mathcal{G}_i^{\vec{\mathfrak{c}_i}}\vec{y},$$

  therefore by IH as $Z \,\mathsf{admis}\, \mathcal{G}_i$ we get

  $$\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}][\vec{y}/\vec{s}] \models \mathcal{G}_i^{\vec{\mathfrak{c}_i}}\vec{y}.$$

- $\nu[X/\mathcal{P}][\vec{x}/\vec{r}][Z/\mathcal{Q}] \models Z\vec{t}[\vec{y} := \vec{x}]$. This is clear from the fact that

  $$\nu[X/P_\alpha][\vec{x}/\vec{r}][Z/\mathcal{Q}_\alpha] \models Z\vec{t}[\vec{y} := \vec{x}] \text{ for all } \alpha \in \mathsf{On}.$$

⊣

Now we relate the definition of the greatest fixed point in section 6.1 with the Mendler-style coinduction principles. We will see that the rule for Mendler-style induction is semantically justified by the construction of the greatest fixed point by transfinite induction

Definition 6.5 *Given a coinductive predicate $\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)$ with $\mathcal{D}_i := \langle \mathcal{F}_i, \vec{\mathfrak{c}_i}\rangle$ and a valuation ν we define the semantical downward hierarchy $\big(P_\alpha\big)_{\alpha\in\mathsf{On}}$ of $\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)$ with respect to ν as follows:*

$$\begin{array}{rcl} P_0 & := & |\mathcal{M}|^n \\ P_{\alpha+1} & := & \bigcap_{i=i}^k Sat(\mathcal{F}_i^{\vec{\mathfrak{c}_i}}, \nu[X/P_\alpha]) \\ P_\lambda & := & \bigcap_{\alpha<\lambda} P_\alpha \;\; \textit{with } \lambda \;\; \textit{a limit ordinal} \end{array}$$

where

$$Sat(\mathcal{F},\nu) := \{\vec{r} \in |\mathcal{M}|^n \mid \nu[\vec{z}/\vec{r}] \models \mathcal{F}\vec{z}\}.$$

Lemma 6.2 *Let $\big(P_\alpha\big)_{\alpha\in\mathsf{On}}$ be the semantical downward hierarchy of the predicate $\nu X(\mathcal{D}_1,\dots,\mathcal{D}_k)$ with respect to ν. If*

$$\nu \models \forall X.(\forall \vec{x}.\mathcal{K}\vec{x} \to X\vec{t}) \to (\forall \vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{t}),\ 1 \le i \le k$$

then the following holds:

1. *For all* $\alpha \in \mathsf{On}$*, if* $\nu[X/P_\alpha] \models \forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}$ *then*
$$\nu[X/P_{\alpha+1}] \models \forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}.$$
2. *For all* $\alpha \in \mathsf{On}$, $\nu[X/P_\alpha] \models \forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}.$

Proof. The second part follows from the first part by observing that as $P_0 = |\mathcal{M}|^n$ we have $\nu[X/P_0] \models \forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}.$
We prove the first part, assume $\nu[X/P_\alpha] \models \forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}$ which, using the main assumption, implies
$$\nu[X/P_\alpha] \models \forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{t}. \tag{6.2}$$
Next assume
$$\nu[X/P_{\alpha+1}][\vec{x}/\vec{r}\,] \models \mathcal{K}\vec{x},$$
so that the goal becomes
$$\nu[X/P_{\alpha+1}][\vec{x}/\vec{r}\,] \models X\vec{t}. \tag{6.3}$$
As $X \notin FV(\mathcal{K})$, from this assumption we get $\nu[X/P_\alpha][\vec{x}/\vec{r}\,] \models \mathcal{K}\vec{x}$. Therefore, using (6.2) we get $\nu[X/P_\alpha][\vec{x}/\vec{r}\,] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{t}$ which by substitution properties is the same as
$$\nu[X/P_\alpha][\vec{z}/\vec{t}^{\mathcal{M}}[\nu[\vec{x}/\vec{r}\,]\,]\,] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{z}.$$
But this fact means that $\vec{t}^{\mathcal{M}}[\nu[\vec{x}/\vec{r}\,]\,] \in Sat(\mathcal{F}_i^{\vec{\mathfrak{c}_i}}, \nu[X/P_\alpha])$ and as this happens for $1 \le i \le k$ we conclude $\vec{t}^{\mathcal{M}}[\nu[\vec{x}/\vec{r}\,]\,] \in P_{\alpha+1}$, which is the same as our goal (6.3). ⊣

Proposition 6.5 *If* $\mathcal{D}_i := \langle \mathcal{F}_i, \vec{\mathfrak{c}_i} \rangle$ *for* $1 \le i \le n$, $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$ *and*
$$\nu \models \forall X.(\forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}) \to (\forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{t}) \;\; 1 \le i \le k,$$
then
$$\nu \models \forall\vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}.$$

Proof. Assume
$$\nu \models \mathcal{F}_i \,\mathsf{mon}\, X \tag{6.4}$$
and
$$\nu \models \forall X.(\forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}) \to (\forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{t}) \tag{6.5}$$
We need to show
$$\nu \models \forall\vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}.$$
Therefore set $\nu' := \nu[\vec{x}/\vec{r}\,]$ and assume
$$\nu' \models \mathcal{K}\vec{x} \tag{6.6}$$

Our goal becomes $\nu' \models \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{t}$, i.e.,

$$\nu' \models \exists X.\mathcal{F}_i \,\mathsf{mon}\, X \wedge X \subseteq \mathcal{F}_i^{\vec{\mathfrak{c}_i}} \wedge X\vec{t} \tag{6.7}$$

Let $\mathcal{P} := \bigcap_{\alpha \in \mathsf{On}} P_\alpha$ be the intersection of the semantical downward hierarchy of $\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)$ w.r.t. ν. We prove:

1. $\nu'[X/\mathcal{P}] \models \mathcal{F}_i \,\mathsf{mon}\, X$.
   Clear by (6.4), because $X, \vec{x} \notin FV(\mathcal{F}_i \,\mathsf{mon}\, X)$.

2. $\nu'[X/\mathcal{P}] \models X\vec{t}$.
   Using the second part of lemma 6.2 and (6.6) as $X \notin FV(\mathcal{K})$ we get $\nu[X/P_\alpha][\vec{x}/\vec{r}] \models X\vec{t}$ for all α, i.e. $\vec{t}^{\mathcal{M}}\Big[\nu[\vec{x}/\vec{r}]\Big] \in P_\alpha$, for all α. Therefore $\vec{t}^{\mathcal{M}}\Big[\nu[\vec{x}/\vec{r}]\Big] \in \bigcap_\alpha P_\alpha$ which is the same as $\nu'[X/\mathcal{P}] \models X\vec{t}$.

3. $\nu'[X/\mathcal{P}] \models X \subseteq \mathcal{F}_i^{\vec{\mathfrak{c}_i}}$.
   Suffices to prove $\nu[X/\mathcal{P}] \models X \subseteq \mathcal{F}_i^{\vec{\mathfrak{c}_i}}$. Assume $\nu[X/\mathcal{P}][\vec{y}/\vec{s}] \models X\vec{y}$, i.e., $\vec{s} \in \bigcap_\alpha P_\alpha$. This implies $\vec{s} \in P_{\alpha+1}$ for all α, which by definition of $P_{\alpha+1}$ leads to $\nu[X/P_\alpha][\vec{x}/\vec{s}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{x}$ for all α, which is the same as $\nu[X/P_\alpha][\vec{y}/\vec{s}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{y}$ for all α. Therefore by the continuity lemma 6.1, as $\mathcal{P} \neq \varnothing$, we get $\nu[X/\mathcal{P}][\vec{y}/\vec{s}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{y}$ and we are done.

Therefore (6.7) is proved. ⊣

Lemma 6.3 *Let $\big(P_\alpha\big)_{\alpha \in \mathsf{On}}$ be the semantical downward hierarchy of the predicate $\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)$ with respect to ν and $\mathcal{D}_i := \langle \mathcal{F}_i, \vec{\mathfrak{c}_i} \rangle$. If $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$ $1 \leq i \leq k$ then for all $\alpha \in \mathsf{On}$,*

$$\nu[X/P_\alpha] \models \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k) \subseteq X.$$

Proof. Induction on α. The case $\alpha = 0$ is obvious as $P_0 = |\mathcal{M}|^n$.
The case for $\alpha = \lambda$ a limit ordinal follows directly from the IH by definition of P_λ. We detail the case for a succesor. Assume $\nu[X/P_{\alpha+1}][\vec{x}/\vec{r}] \models \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{x}$, which, as X is not free in $\nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{x}$, is the same as

$$\nu[X/P_\alpha][\vec{x}/\vec{r}] \models \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)\vec{x}.$$

Using this and the obvious $\models \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k) \subseteq \mathcal{F}_i^{\vec{\mathfrak{c}_i}}[X := \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)]$ we get

$$\nu[X/P_\alpha][\vec{x}/\vec{r}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}[X := \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)]\vec{x}.$$

On the other hand, using IH and the fact that $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$ we conclude

$$\nu[X/P_\alpha][\vec{x}/\vec{r}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}[X := \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)] \subseteq \mathcal{F}_i^{\vec{\mathfrak{c}_i}}.$$

Therefore we get

$$\nu[X/P_\alpha][\vec{x}/\vec{r}] \models \mathcal{F}_i^{\vec{\mathfrak{c}_i}}\vec{x},$$

but this happens for $1 \leq i \leq k$, which implies $\vec{r} \in P_{\alpha+1}$, i.e.,

$$\nu[X/P_{\alpha+1}][\vec{x}/\vec{r}] \models X\vec{x}$$

and we are done.

$\dashv$

Proposition 6.6 *If* $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$ *and*

$$\nu \models \forall X.\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq X \to (\forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}) \to (\forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{t}) \;\; 1 \leq i \leq k,$$

then

$$\nu \models \forall\vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}.$$

Proof. Immediate from the proof of proposition 6.5 and lemma 6.3. $\dashv$

Theorem 6.2 (Soundness of the Logic $\mathsf{MCICD}^\star_{\mu M\nu}$) *If* $\Gamma \vdash_{\mathsf{MCICD}^\star_{\mu M\nu},\mathbb{E}} s : A$ *then* $\Gamma, \mathbb{E} \models A$.
Proof. Induction on $\vdash_{\mathsf{MCICD}^\star_{\mu M\nu},\mathbb{E}}$.
Case $(M\nu I)$. Assume $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \forall X.\big(\forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}\big) \to \big(\forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{t}\big), \;\; 1 \leq i \leq k$$

As the rule $(M\nu I)$ requires $X \,\mathsf{admis}\, \mathcal{F}_i$, proposition 6.3 implies $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$. Therefore by proposition 6.5 we have

$$\nu \models \forall\vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$$

Case $(M\nu I^+)$. Assume $\nu \models \Gamma, \mathbb{E}$. By IH we have

$$\nu \models \forall X.\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \subseteq X \to (\forall\vec{x}.\mathcal{K}\vec{x} \to X\vec{t}) \to (\forall\vec{x}.\mathcal{K}\vec{x} \to \mathcal{F}_i^{\vec{\mathbb{c}_i}}\vec{t}) \;\; 1 \leq i \leq k$$

As the rule $(M\nu I^+)$ requires $X \,\mathsf{admis}\, \mathcal{F}_i$, proposition 6.3 implies $\nu \models \mathcal{F}_i \,\mathsf{mon}\, X$. Therefore by proposition 6.6 we have

$$\nu \models \forall\vec{x}.\mathcal{K}\vec{x} \to \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\vec{t}$$

$\dashv$

6.5 Programming with Proofs in $\mathsf{MCICD}_{\mu M\nu}$

To finalize the chapter we extend the programming with proofs paradigm to our new logic. We start by observing that as both realizability and logic soundness hold for the logic $\mathsf{MCICD}_{\mu M\nu}$, (theorems 6.1 and 6.2) the semantical soundness theorem (proposition 5.5) and the conservation lemma (corollary 5.1) hold also for the new logic, therefore we obtain again as a corollary the correctness lemma which is the cornerstone of the programming method. Here we state it again:

Corollary 6.1 (Correctness Lemma for $\mathsf{MCICD}_{\mu M\nu}$) *Let f be a function symbol, $\mathcal{D}_i$, $\mathcal{E}$ data types in $\mathcal{M}$ and s_i an inhabitant of $\mathcal{D}_i$ (i.e. $\mathcal{M} \models \mathcal{D}_i$). If $\mathcal{W}(t)$ comprises only canonical witnesses, $\mathcal{M}$ satisfies $\mathbb{E}$ and*

$$\vdash_{\mathsf{MCICD}_{\mu M\nu},\mathbb{E}} t : \forall x_1 \dots \forall x_n.\mathcal{D}_1 x_1, \dots, \mathcal{D}_n x_n \to \mathcal{E} f(x_1, \dots, x_n),$$

then

$$\mathcal{M} \models t s_1 \dots s_n = f(s_1, \dots, s_n).$$

Therefore the $\mathsf{MCICT}_{\mu M\nu}$-term t is a program to compute the function $f^{\mathcal{M}}$.

However although the correctness lemma holds and we have different coinduction principles our new logic is still not useful to program functions involving coinductive predicates as we have not solved the problem of getting formal data types from coinductive predicates. This question will be addressed in the next section.

6.5.1 Data types with Equality

As mentioned in page 150 when trying to prove that if $\mathcal{A}$ is a data type then

$$\mathcal{S}_{\mathcal{A}} := \nu X \Big(\langle \mathcal{A}, \mathsf{head} \rangle, \langle X, \mathsf{tail} \rangle \Big)$$

is again a data type we are not able to do it due to the fact that Leibniz' equality is not good for infinite data types. In this section we give a solution to this problem by generalizing the concept of formal data type to defined equalities. Similar solutions can be found in chapter eight of [Raf94].

Definition 6.6 *Given a unary predicate $\mathcal{A} := \lambda x.A[x]$ with $FV(A) = \{x\}$ we say that the binary predicate $\approx_{\mathcal{A}}$ defines an equality for $\mathcal{A}$ if the following holds:*

- ∘ $FV(x \approx_{\mathcal{A}} y) = \{x, y\}$.
- ∘ $\vdash \forall x \forall y. x \approx_{\mathcal{A}} y \to \mathcal{A}x \wedge \mathcal{A}y$.
- ∘ $\vdash \forall x. \mathcal{A}x \to x \approx_{\mathcal{A}} x$.
- ∘ $\vdash \forall x \forall y. x \approx_{\mathcal{A}} y \to y \approx_{\mathcal{A}} x$.
- ∘ $\vdash \forall x \forall y \forall z. x \approx_{\mathcal{A}} y, y \approx_{\mathcal{A}} z \to x \approx_{\mathcal{A}} z$.

Given a predicate $\mathcal{A} := \lambda x.A[x]$ there is a trivial way of defining an equality for $\mathcal{A}$, just take the Leibniz Equality restricted to $\mathcal{A}$, i.e.,

$$\approx_{\mathcal{A}} := \lambda xy.\mathcal{A}x \wedge \mathcal{A}y \wedge x = y$$

An equality for the product predicate can be defined as follows

Proposition 6.7 *If $\approx_{\mathcal{A}}, \approx_{\mathcal{B}}$ are equalities for $\mathcal{A}, \mathcal{B}$ respectively then*

$$\approx_{\times} := \nu Y\Big(\langle \lambda x, y.x \approx_{\mathcal{A}} y, \pi_1, \pi_1\rangle, \langle \lambda x, y.x \approx_{\mathcal{B}} y, \pi_2, \pi_2\rangle\Big)$$

is an equality for their product $\mathcal{A} \times \mathcal{B}$, defined as

$$\mathcal{A} \times \mathcal{B} := \nu X\left(\langle \mathcal{A}, \pi_1\rangle, \langle \mathcal{B}, \pi_2\rangle\right)$$

Proof. Straightforward ⊣

With this definition we do not need extensionality to get an equality for the product predicate.

The problem of an adequate equality for the streams predicate is solved in the following

Proposition 6.8 *Given the coinductive predicate defining $\mathcal{A}$-streams*

$$\mathcal{S}_{\mathcal{A}} := \nu X\Big(\langle \mathcal{A}, \mathsf{head}\rangle, \langle X, \mathsf{tail}\rangle\Big)$$

and $\approx_{\mathcal{A}}$ an equality for $A[x]$, the binary predicate

$$\approx_{\mathcal{S}_A} := \nu Y\Big(\langle \lambda x, y.x \approx_{\mathcal{A}} y, \mathsf{head}, \mathsf{head}\rangle, \langle Y, \mathsf{tail}, \mathsf{tail}\rangle\Big)$$

defines an equality for $\mathcal{S}_A$ in the system $\mathsf{MCICD}_{\mu M\nu}$.

Proof. We prove the five properties of an equality:

- ○ $FV(x \approx_{\mathcal{S}_{\mathcal{A}}} y) = \{x, y\}$. Is clear.
- ○ $\vdash_{\mathsf{MCICD}_{\mu M\nu}} \forall xy.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to \mathcal{S}_A x \wedge \mathcal{S}_A y$.
  We prove $\vdash \forall xy.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to \mathcal{S}_A x$, using $(M\nu I)$. We need to show

  (A). $\vdash \forall X.(\forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to X\,\mathsf{tail}\,x) \to \forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to A\,\mathsf{head}\,\mathsf{tail}\,x$
  (B). $\vdash \forall X.(\forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to X\,\mathsf{tail}\,x) \to \forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to X\,\mathsf{tail}\,\mathsf{tail}\,x$

  Set $\Gamma := \{(\forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to X\,\mathsf{tail}\,x), x \approx_{\mathcal{S}_{\mathcal{A}}} y\}$. For (A) we have

  $\Gamma \vdash \mathsf{tail}\,x \approx_{\mathcal{S}_{\mathcal{A}}} \mathsf{tail}\,y$ (Coclosure Ax.)
  $\Gamma \vdash \mathsf{head}\,\mathsf{tail}\,x \approx_{\mathcal{A}} \mathsf{head}\,\mathsf{tail}\,y$ (Coclosure Ax.)
  $\Gamma \vdash A\,\mathsf{head}\,\mathsf{tail}\,x \wedge A\,\mathsf{head}\,\mathsf{tail}\,y$ ($\approx_{\mathcal{A}}$ is an equality)
  $\Gamma \vdash A\,\mathsf{head}\,\mathsf{tail}\,x$

  For (B) we have

  $\Gamma \vdash \mathsf{tail}\,x \approx_{\mathcal{S}_{\mathcal{A}}} \mathsf{tail}\,y$ (Coclosure Ax.)
  $\Gamma \vdash \mathsf{tail}\,x \approx_{\mathcal{S}_{\mathcal{A}}} \mathsf{tail}\,y \to X\,\mathsf{tail}\,\mathsf{tail}\,x$
  $\Gamma \vdash X\,\mathsf{tail}\,\mathsf{tail}\,x$

  Analogously we get $\vdash \forall x, y.x \approx_{\mathcal{S}_{\mathcal{A}}} y \to \mathcal{S}_A y$ and we are done.

- $\forall x.\mathcal{S}_A x \to x \approx_{\mathcal{S}_\mathcal{A}} x$. We use $(M\nu I)$ with $\mathcal{K} := \lambda x, y.\mathcal{S}_A x$, so we need to prove

  (A). $\vdash \forall Y.(\forall x, y.\mathcal{S}_A x \to Y\,\mathsf{tail}\,x\,\mathsf{tail}\,x) \to \forall x, y.\mathcal{S}_A x \to \mathsf{head}\,\mathsf{tail}\,x \approx_\mathcal{A} \mathsf{head}\,\mathsf{tail}\,x$

  (B). $\vdash \forall Y.(\forall x, y.x\mathcal{S}_A x \to Y\,\mathsf{tail}\,x\,\mathsf{tail}\,x) \to \forall x, y.\mathcal{S}_A x \to Y\,\mathsf{tail}\,x\,\mathsf{tail}\,x$

  Set $\Gamma := \{(\forall x, y.\mathcal{S}_A x \to Y\,\mathsf{tail}\,x\,\mathsf{tail}\,x), \mathcal{S}_A x\}$.
  For (A) we have

  $$\begin{array}{l} \Gamma \vdash \mathcal{S}_A\,\mathsf{tail}\,x \quad \text{(Coclosure Ax.)} \\ \Gamma \vdash A\,\mathsf{head}\,\mathsf{tail}\,x \quad \text{(Coclosure Ax.)} \\ \Gamma \vdash \mathsf{head}\,\mathsf{tail}\,x \approx_\mathcal{A} \mathsf{head}\,\mathsf{tail}\,x \quad (\approx_\mathcal{A} \text{ is an equality}) \end{array}$$

  For (B),

  $$\begin{array}{l} \Gamma \vdash \mathcal{S}_A\,\mathsf{tail}\,x \quad \text{(Coclosure Ax.)} \\ \Gamma \vdash \mathcal{S}_A\,\mathsf{tail}\,x \to Y\,\mathsf{tail}\,\mathsf{tail}\,x\,\mathsf{tail}\,\mathsf{tail}\,x \\ \Gamma \vdash Y\,\mathsf{tail}\,\mathsf{tail}\,x\,\mathsf{tail}\,\mathsf{tail}\,x \end{array}$$

- $\forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to y \approx_{\mathcal{S}_\mathcal{A}} x$. Again we use $(M\nu I)$ with $\mathcal{K} := \lambda xy.x \approx_{\mathcal{S}_\mathcal{A}} y$, so it suffices to prove

  (A). $\vdash \forall Y.(\forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to Yyx) \to \forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to \mathsf{head}\,y \approx_\mathcal{A} \mathsf{head}\,x$

  (B). $\vdash \forall Y.(\forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to Yyx) \to \forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to Y\,\mathsf{tail}\,y\,\mathsf{tail}\,x$

  Set $\Gamma := \{\forall x, y.x \approx_{\mathcal{S}_\mathcal{A}} y \to Yyx, x \approx_{\mathcal{S}_\mathcal{A}} y\}$.
  For (A) we have

  $$\begin{array}{l} \Gamma \vdash x \approx_{\mathcal{S}_\mathcal{A}} y \\ \Gamma \vdash \mathsf{head}\,x \approx_\mathcal{A} \mathsf{head}\,y \quad \text{(Coclosure Ax.)} \\ \Gamma \vdash \mathsf{head}\,y \approx_\mathcal{A} \mathsf{head}\,x \quad (\approx_\mathcal{A} \text{ is an equality}) \end{array}$$

  For (B),

  $$\begin{array}{l} \Gamma \vdash \mathsf{tail}\,x \approx_{\mathcal{S}_\mathcal{A}} \mathsf{tail}\,y \quad \text{(Coclosure Ax.)} \\ \Gamma \vdash \mathsf{tail}\,x \approx_{\mathcal{S}_\mathcal{A}} \mathsf{tail}\,y \to Y\,\mathsf{tail}\,y\,\mathsf{tail}\,x \\ \Gamma \vdash Y\,\mathsf{tail}\,y\,\mathsf{tail}\,x \end{array}$$

- $\forall x, y, z.x \approx_{\mathcal{S}_\mathcal{A}} y \wedge y \approx_{\mathcal{S}_\mathcal{A}} z \to x \approx_{\mathcal{S}_\mathcal{A}} z$.
  Again we use $(M\nu I)$ with $\mathcal{K} := \lambda xz.x \approx_{\mathcal{S}_\mathcal{A}} y \wedge y \approx_{\mathcal{S}_\mathcal{A}} z$.

$\dashv$

The concept of formal data type is generalized as follows:

Definition 6.7 *Given a predicate $\mathcal{D} := \lambda x.D[x]$ with $FV(D) = \{x\}$, an equality $\approx_{\mathcal{D}}$ for $\mathcal{D}$ and a model $\mathcal{M}$, we say that $\langle \mathcal{D}, \approx_{\mathcal{D}} \rangle$ is a data type with equality in $\mathcal{M}$ if and only if:*

$$\mathcal{M} \models \forall x \forall y. y \ \mathtt{r}\ D[x] \leftrightarrow x \approx_{\mathcal{D}} y.$$

Observe that we do not require now $D[x]$ on the right hand side of the above equivalence because this is derivable from $x \approx_{\mathcal{D}} y$.

Proposition 6.9 *If $\langle \mathcal{A}, \approx_{\mathcal{A}} \rangle, \langle \mathcal{B}, \approx_{\mathcal{B}} \rangle$ are data types with equality and ${\mathbb{\pi}_1}^{\mathcal{M}} := \mathbb{D}^2_1, {\mathbb{\pi}_2}^{\mathcal{M}} := \mathbb{D}^2_2$ then their product $\langle \mathcal{A} \times \mathcal{B}, \approx_{\times} \rangle$ is a data type with equality.*

Proof. Straightforward ⊣

Proposition 6.10 *If $\langle \mathcal{A}, \approx_{\mathcal{A}} \rangle$ is a data type with equality, $\mathsf{head}^{\mathcal{M}} := \mathbb{D}^2_1$ and $\mathsf{tail}^{\mathcal{M}} := \mathbb{D}^2_2$ then $\langle \mathcal{S}_{\mathcal{A}}, \approx_{\mathcal{S}_{\mathcal{A}}} \rangle$ is a data type with equality.*
Proof. Assume that $\langle \mathcal{A}, \approx_{\mathcal{A}} \rangle$ is a data type with equality. The goal is to prove

$$\mathcal{M} \models \forall xy. y \ \mathtt{r}\ \mathcal{S}_A x \leftrightarrow y \approx_{\mathcal{S}_{\mathcal{A}}} x$$

Take a valuation ν and $r, s \in |\mathcal{M}|$ and set $\nu' := \nu[x, y/r, s]$.
$\Rightarrow$) Assume $\nu' \models y \ \mathtt{r}\ \mathcal{S}_A x$. That is there exists a $\mathcal{Q} \subseteq |\mathcal{M}|^2$ such that

$$\begin{aligned} \nu'[X^+/\mathcal{Q}] \models \quad & \mathcal{A}^{\mathtt{r}}\,\mathsf{mon}\,X^+ \wedge X^+\,\mathsf{mon}\,X^+ \wedge X^+ \subseteq \mathcal{A}^{\mathtt{r}\,\mathsf{head},\mathbb{D}^2_1} \\ & \wedge\ X^+ \subseteq {X^+}^{\mathsf{tail},\mathbb{D}^2_2} \wedge X^+ xy \end{aligned}$$

The goal is to prove $\nu' \models y \approx_{\mathcal{S}_{\mathcal{A}}} x$.

- ∘ $\nu'[Y/Q] \models (\lambda x, y.x \approx_{\mathcal{A}} y)\,\mathsf{mon}\,Y$. Is clear.
- ∘ $\nu'[Y/Q] \models Y\,\mathsf{mon}\,Y$. Is clear.
- ∘ $\nu'[Y/Q] \models Y \subseteq (\lambda x, y.x \approx_{\mathcal{A}} y)^{\mathsf{head},\mathsf{head}}$. From the assumption we get $\nu'[X^+/\mathcal{Q}] \models X^+ \subseteq \mathcal{A}^{\mathtt{r}\,\mathsf{head},\mathbb{D}^2_1}$ which by $(Fsp6)$ is the same as $\nu'[Y/\mathcal{Q}] \models Y \subseteq \mathcal{A}^{\mathtt{r}\,\mathsf{head},\mathbb{D}^2_1}$ which as $\mathsf{head}^{\mathcal{M}} := \mathbb{D}^2_1$ equals $\nu'[Y/\mathcal{Q}] \models Y \subseteq \mathcal{A}^{\mathtt{r}\,\mathsf{head},\mathsf{head}}$. Now assume $\nu'[Y/\mathcal{Q}] \models Yuv$, this implies $\nu'[Y/\mathcal{Q}] \models \mathcal{A}^{\mathtt{r}\,\mathsf{head},\mathsf{head}}uv$. But as $\langle \mathcal{A}, \approx_{\mathcal{A}} \rangle$ is a data type with equality the last fact yields $\nu'[Y/\mathcal{Q}] \models \mathsf{head}\,u \approx_{\mathcal{A}} \mathsf{head}\,v$, that is, $\nu'[Y/\mathcal{Q}] \models (\lambda x, y.x \approx_{\mathcal{A}} y)^{\mathsf{head},\mathsf{head}}uv$.
- ∘ $\nu'[Y/Q] \models Y \subseteq Y^{\mathsf{tail},\mathsf{tail}}$. By assumption we have $\nu'[X^+/\mathcal{Q}] \models X^+ \subseteq {X^+}^{\mathsf{tail},\mathbb{D}^2_2}$ which by $(Fsp6)$ and as $\mathsf{tail}^{\mathcal{M}} := \mathbb{D}^2_2$ is the same as $\nu'[Y/Q] \models Y \subseteq Y^{\mathsf{tail},\mathsf{tail}}$ and we are done.
- ∘ $\nu'[Y/Q] \models Yyx$. Analogously from the assumption $\nu'[X^+/\mathcal{Q}] \models X^+xy$ and $(Fsp6)$.

The previous five facts prove $\nu' \models x \approx_{\mathcal{S}_{\mathcal{A}}} y$. Finally as $\approx_{\mathcal{S}_{\mathcal{A}}}$ is an equality we conclude $\nu' \models y \approx_{\mathcal{S}_{\mathcal{A}}} x$.

$\Leftarrow$) Assume $\nu' \models y \approx_{\mathcal{S}_\mathcal{A}} x$, this implies $\nu' \models x \approx_{\mathcal{S}_\mathcal{A}} y$, i.e. there is a $\mathcal{Q} \subseteq |\mathcal{M}|^2$ such that

$$\begin{aligned}\nu'[Y/\mathcal{Q}] \models \quad & (\lambda x, y.x \approx_\mathcal{A} y)\,\mathsf{mon}\,Y \wedge Y\,\mathsf{mon}\,Y \wedge Y \subseteq (\lambda x, y.x \approx_\mathcal{A} y)^{\mathsf{head},\mathsf{head}} \\ & \wedge\ Y \subseteq Y^{\mathsf{tail},\mathsf{tail}} \wedge Yxy\end{aligned}$$

Goal is $\nu' \models y \;\mathtt{r}\; \mathcal{S}_A x$, i.e.

$$\nu' \models X^+\Big(\langle A^{\mathrm{r}}, \mathsf{head}, \mathbb{D}^2_1\rangle, \langle X^+, \mathsf{tail}, \mathbb{D}^2_2\rangle\Big)xy.$$

We prove now:

- ○ $\nu'[Y/\mathcal{Q}] \models \mathcal{A}^{\mathrm{r}}\,\mathsf{mon}\,Y$. Is clear.
- ○ $\nu'[Y/\mathcal{Q}] \models Y\,\mathsf{mon}\,Y$. Is clear.
- ○ $\nu'[Y/\mathcal{Q}] \models Y \subseteq \mathcal{A}^{\mathrm{r}\,\mathsf{head},\mathbb{D}^2_1}$. Assume $\nu'[Y/\mathcal{Q}] \models Yuv$, the main assumption yields $\nu'[Y/\mathcal{Q}] \models \mathsf{head}\,u \approx_\mathcal{A} \mathsf{head}\,v$. But $\langle \mathcal{A}, \approx_\mathcal{A}\rangle$ is a data type with equality, therefore we get $\nu'[Y/\mathcal{Q}] \models \mathsf{head}\,v \;\mathtt{r}\; \mathcal{A}[\mathsf{head}\,u]$, i.e., $\nu'[Y/\mathcal{Q}] \models \mathcal{A}^{\mathrm{r}\,\mathsf{head},\mathsf{head}}uv$ and we are done as $\mathsf{head}^\mathcal{M} := \mathbb{D}^2_1$.
- ○ $\nu'[Y/\mathcal{Q}] \models Y \subseteq Y^{\mathsf{tail},\mathbb{D}^2_2}$. Immediate from the assumption $\nu'[Y/\mathcal{Q}] \models Y \subseteq Y^{\mathsf{tail},\mathsf{tail}}$, as $\mathsf{tail}^\mathcal{M} := \mathbb{D}^2_2$.
- ○ $\nu'[Y/\mathcal{Q}] \models Yxy$. Is part of the main assumption.

These facts prove $\nu' \models \nu Y\big(\langle \mathcal{A}^{\mathrm{r}}, \mathsf{head}, \mathbb{D}^2_1\rangle, \langle Y, \mathsf{tail}, \mathbb{D}^2_2\rangle\big)xy$. Therefore we are done.

⊣

Now that we have solved the problem of equality in streams with the concept of data type with equality, we would like to program with these kind of data types. The following generalization of the correctness lemma allow us to do it.

Proposition 6.11 (Correctness Lemma for Data Types with Equality) *Let f be a function symbol, $\langle \mathcal{D}_i, \approx_i\rangle$, $\langle \mathcal{E}, \approx_\mathcal{E}\rangle$ data types with equality in $\mathcal{M}$ and s_i an inhabitant of $\mathcal{D}_i$ (i.e. $\mathcal{M} \models \mathcal{D}_i s_i$).*
If $\mathcal{W}(t)$ comprises only canonical witnesses, $\mathcal{M}$ satisfies $\mathbb{E}$ and

$$\vdash_{\mathsf{MCICD}_{\mu M\nu},\mathbb{E}} t : \forall x_1 \ldots \forall x_n.\mathcal{D}_1 x_1, \ldots, \mathcal{D}_n x_n \to \mathcal{E} f(x_1, \ldots, x_n),$$

then

$$\mathcal{M} \models ts_1 \ldots s_n \approx_\mathcal{E} f(s_1, \ldots, s_n).$$

Therefore the $\mathsf{MCICT}_{\mu M\nu}$-term t is a program to compute the function $f^\mathcal{M}$.

Moreover, f is compatible with respect to $\approx_i, \approx_\mathcal{E}$, i.e.,

$$\mathcal{M} \models r_i \approx_i s_i \to f\vec{r} \approx_\mathcal{E} f\vec{s}$$

6.5.2 Programming with Mendler-style Coiteration or Corecursion

Observe that the goal for programming functions into a coinductive predicate

$$\vdash t : \forall x.D[x] \to \nu X(\mathcal{D}_1, \dots, \mathcal{D}_k)(fx)$$

is now achieved very easily using Mendler-style coiteration or corecursion, the obvious choice for the predicate $\mathcal{K}$ is $\mathcal{K} := \lambda x.D[x]$.

Let us develop some examples.

A Stream of Constants

We want to program a function cst from a data type $\mathcal{D}$ into the data type $\mathcal{S}_{\mathcal{D}}$ of streams of elements of D, such that $\mathsf{cst}(a) \approx_{\mathcal{S}_{\mathcal{D}}} \langle a, a, a, \dots \rangle$. The function is destructed as follows:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{cst}\,a) & = & a \\ \mathsf{tail}(\mathsf{cst}\,a) & = & \mathsf{cst}\,a \end{array}$$

The goal is to obtain $\vdash t : \forall x.\mathcal{D}x \to \mathcal{S}_{\mathcal{D}}[\mathsf{cst}\,x]$.

We need to derive the premises of the Mendler-style coiteration rule for $\Gamma = \varnothing, \mathcal{K} := \mathcal{D},\ t := \mathsf{cst}\,x$.

$$\begin{array}{rll} x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash & ? : \mathcal{D}\,\mathsf{head}(\mathsf{cst}\,x) \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash & y : \mathcal{D}x \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash_{\mathbb{E}(\mathsf{cst})} & y : \mathcal{D}\,\mathsf{head}(\mathsf{cst}\,x) \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x & \vdash_{\mathbb{E}(\mathsf{cst})} & \lambda yy : \forall x.\mathcal{D}x \to \mathcal{D}\,\mathsf{head}(\mathsf{cst}\,x) \end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{cst})} \lambda x\lambda yy : \forall X.(\forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x) \to \forall x.\mathcal{D}x \to \mathcal{D}^{\mathsf{head}}(\mathsf{cst}\,x)$$

$$\begin{array}{rll} x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash & ? : X\,\mathsf{tail}(\mathsf{cst}\,x) \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash & xy : X\,\mathsf{cst}\,x \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x, y : \mathcal{D}x & \vdash_{\mathbb{E}(\mathsf{cst})} & xy : X\,\mathsf{tail}(\mathsf{cst}\,x) \\ x : \forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x & \vdash_{\mathbb{E}(\mathsf{cst})} & \lambda y.xy : \forall x.\mathcal{D}x \to X\,\mathsf{tail}(\mathsf{cst}\,x) \end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{cst})} \lambda x\lambda y.xy : \forall X.(\forall x.\mathcal{D}x \to X\,\mathsf{cst}\,x) \to \forall x.\mathcal{D}x \to X^{\mathsf{tail}}(\mathsf{cst}\,x)$$

Both derivations yield

$$\vdash_{\mathbb{E}(\mathsf{cst})} \mathsf{MCoIt}_2(\lambda x\lambda yy)(\lambda x\lambda y.xy) : \forall x.\mathcal{D}x \to \mathcal{S}_{\mathcal{D}}\,\mathsf{cst}\,x$$

Now if we set $\overline{\mathsf{cst}} := \mathsf{MCoIt}_2(\lambda x\lambda yy)(\lambda x\lambda y.xy)$ we get

$$\overline{\mathsf{head}}\,\overline{\mathsf{cst}}x \to_\beta \underset{2,1}{\mathsf{out}}\,\overline{\mathsf{cst}}x \to_\beta (\lambda x\lambda yy)(\overline{\mathsf{cst}})x \to_\beta x.$$

$$\overline{\mathsf{tail}}\,\overline{\mathsf{cst}}x \to_\beta \mathsf{out}_{2,2}\overline{\mathsf{cst}}x \to_\beta (\lambda x\lambda y.xy)(\overline{\mathsf{cst}})x \to_\beta \overline{\mathsf{cst}}x.$$

Stream of natural numbers from a given one

The from function can now be programmed very easily by means of Mendler-style coiteration. from is a function from $\mathbb{N}$ into $\mathcal{S}_{\mathbb{N}}$ such that

$$\mathsf{from}\ n \approx_{\mathcal{S}_{\mathbb{N}}} \langle n, n+1, n+2, \ldots \rangle.$$

This function is destructed as:

$$\begin{array}{rcl}\mathsf{head}(\mathsf{from}\ n) & = & n \\ \mathsf{tail}(\mathsf{from}\ n) & = & \mathsf{from}\ s(n)\end{array}$$

The goal is to obtain $\vdash t : \forall x.\mathbb{N}x \rightarrow \mathcal{S}_{\mathbb{N}}\,\mathsf{from}\,x$.

$$\begin{array}{rll}
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash & ? : \mathbb{N}\,\mathsf{head}(\mathsf{from}\ x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash & y : \mathbb{N}x \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash_{\mathbb{E}(\mathsf{from}\,)} & y : \mathbb{N}\,\mathsf{head}(\mathsf{from}\ x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x & \vdash_{\mathbb{E}(\mathsf{from}\,)} & \lambda yy : \forall x.\mathbb{N}x \rightarrow \mathbb{N}\,\mathsf{head}(\mathsf{from}\ x)
\end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{from}\,)} \lambda x \lambda yy : \forall X.(\forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x) \rightarrow \forall x.\mathbb{N}x \rightarrow \mathbb{N}^{\mathsf{head}}(\mathsf{from}\ x)$$

$$\begin{array}{rll}
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash & ? : X\,\mathsf{tail}(\mathsf{from}\ x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash & \overline{s}\,y : \mathbb{N}s(x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash & x(\overline{s}\,y) : X\,\mathsf{from}\ s(x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x, y : \mathbb{N}x & \vdash_{\mathbb{E}(\mathsf{from}\,)} & x(\overline{s}\,y) : X\,\mathsf{tail}(\mathsf{from}\ x) \\
x : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x & \vdash_{\mathbb{E}(\mathsf{from}\,)} & \lambda y.x(\overline{s}\,y) : \forall x.\mathbb{N}x \rightarrow X\,\mathsf{tail}(\mathsf{from}\ x)
\end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{from}\,)} \lambda x \lambda y.x(\overline{s}\,y) : \forall X.(\forall x.\mathbb{N}x \rightarrow X\,\mathsf{from}\ x) \rightarrow \forall x.\mathbb{N}x \rightarrow X^{\mathsf{tail}}(\mathsf{from}\ x)$$

Both derivations yield

$$\vdash_{\mathbb{E}(\mathsf{from}\,)} \mathsf{MCoIt}_2(\lambda x \lambda yy)(\lambda x \lambda y.x(\overline{s}\,y)) : \forall x.\mathbb{N}x \rightarrow \mathcal{S}_{\mathbb{N}}\,\mathsf{from}\ x$$

Now if we set $\overline{\mathsf{from}\,} := \mathsf{MCoIt}_2(\lambda x \lambda yy)(\lambda x \lambda y.x(\overline{s}\,y))$ we get

$$\overline{\mathsf{head}}\,\overline{\mathsf{from}}\,x \rightarrow_\beta \mathsf{out}_{2,1}\overline{\mathsf{from}}\,x \rightarrow_\beta (\lambda x \lambda yy)(\overline{\mathsf{from}\,})x \rightarrow_\beta x.$$

$$\overline{\mathsf{tail}}\,\overline{\mathsf{from}}\,x \rightarrow_\beta \mathsf{out}_{2,2}\overline{\mathsf{from}}\,x \rightarrow_\beta (\lambda x \lambda y.x(\overline{s}\,y))(\overline{\mathsf{from}\,})x \rightarrow_\beta \overline{\mathsf{from}\,}(\overline{s}\,x).$$

Stream of Succesors

The function ss from $\mathcal{S}_{\mathbb{N}}$ into $\mathcal{S}_{\mathbb{N}}$ such that

$$\mathsf{ss}\langle a_1, \ldots, a_n, \ldots\rangle \approx_{\mathcal{S}_{\mathbb{N}}} \langle sa_1, \ldots, sa_n, \ldots\rangle,$$

is destructed as:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{ss}\,x) & = & s(\mathsf{head}\ x) \\ \mathsf{tail}(\mathsf{ss}\,x) & = & \mathsf{ss}(\mathsf{tail}\ x) \end{array}$$

This apparently simple example causes important problems in the formalism of [Tat93] and forced the author to develop a complex tailor-made system for extracting program from streams. In contrast the stream of successors can easily be programmed with Mendler-style coiteration:

$$\begin{array}{rll} x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & ? : \mathbb{N}\,\mathsf{head}(\mathsf{ss}\,x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & \overline{\mathsf{head}}y : \mathbb{N}\,\mathsf{head}\ x \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & \overline{s}\,\overline{\mathsf{head}}y : \mathbb{N}s(\mathsf{head}\ x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash_{\mathbb{E}(\mathsf{ss})} & \overline{s}\ \overline{\mathsf{head}}y : \mathbb{N}\,\mathsf{head}(\mathsf{ss}\,x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x & \vdash_{\mathbb{E}(\mathsf{ss})} & \lambda y.\overline{s}\ \overline{\mathsf{head}}y : \forall x.\mathcal{S}_{\mathbb{N}}x \to \mathbb{N}\,\mathsf{head}(\mathsf{ss}\,x) \end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{ss})} \lambda x\lambda y.\overline{s}\ \overline{\mathsf{head}}y : \forall X.(\forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}x) \to \forall x.\mathcal{S}_{\mathbb{N}}x \to \mathbb{N}^{\mathsf{head}}(\mathsf{ss}x)$$

$$\begin{array}{rll} x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & ? : X\,\mathsf{tail}(\mathsf{ss}\,x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & \overline{\mathsf{tail}}\,y : \mathcal{S}_{\mathbb{N}}\,\mathsf{tail}\ x \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash & x(\overline{\mathsf{tail}}\,y) : X\mathsf{ss}(\mathsf{tail}\ x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x, y : \mathcal{S}_{\mathbb{N}}x & \vdash_{\mathbb{E}(\mathsf{ss})} & x(\overline{\mathsf{tail}}\,y) : X\,\mathsf{tail}(\mathsf{ss}\,x) \\ x : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}\,x & \vdash_{\mathbb{E}(\mathsf{ss})} & \lambda y.x(\overline{\mathsf{tail}}\,y) : \forall x.\mathcal{S}_{\mathbb{N}}x \to X\,\mathsf{tail}(\mathsf{ss}\,x) \end{array}$$

Therefore

$$\vdash_{\mathbb{E}(\mathsf{ss})} \lambda x\lambda y.x(\overline{\mathsf{tail}}\,y) : \forall X.(\forall x.\mathcal{S}_{\mathbb{N}}x \to X\mathsf{ss}x) \to \forall x.\mathcal{S}_{\mathbb{N}}x \to X^{\mathsf{tail}}(\mathsf{ss}\,x)$$

Both derivations yield

$$\vdash_{\mathbb{E}(\mathsf{ss})} \mathsf{MCoIt}_2\Big(\lambda x\lambda y.\overline{s}\ \overline{\mathsf{head}}y\Big)\Big(\lambda x\lambda y.x(\overline{\mathsf{tail}}\,y)\Big) : \forall x.\mathcal{S}_{\mathbb{N}}x \to \mathcal{S}_{\mathbb{N}}\mathsf{ss}\,x$$

Now if we set $\overline{\mathsf{ss}} := \mathsf{MCoIt}_2\Big(\lambda x\lambda y.\overline{s}\ \overline{\mathsf{head}}y\Big)\Big(\lambda x\lambda y.x(\overline{\mathsf{tail}}\,y)\Big)$ we get

$$\overline{\mathsf{head}}\ \overline{\mathsf{ss}}x \to_\beta \Big(\lambda x\lambda y.\overline{s}\ \overline{\mathsf{head}}y\Big)(\overline{\mathsf{ss}})x \to_\beta \overline{s}\ \overline{\mathsf{head}}x$$

$$\overline{\mathsf{tail}}\ \overline{\mathsf{ss}}x \to_\beta \Big(\lambda x\lambda y.x(\overline{\mathsf{tail}}\,y)\Big)(\overline{\mathsf{ss}})x \to_\beta \overline{\mathsf{ss}}(\overline{\mathsf{tail}}\,x)$$

The Map Head Function

As an example of programming with Mendler-style corecursion we program the map head function (see page 73 for a program with conventional corecursion). Given a function $h : \mathcal{A} \to \mathcal{A}$ the map head function $\mathsf{maphd}_h : \mathcal{S}_\mathcal{A} \to \mathcal{S}_\mathcal{A}$ is destructed as follows:

$$\begin{array}{rcl} \mathsf{head}(\mathsf{maphd}_h\, x) & = & h(\mathsf{head}\ x) \\ \mathsf{tail}(\mathsf{maphd}_h\, x) & = & \mathsf{tail}\ x \end{array}$$

Of course we need to assume that the function h is computable by a program $\overline{h}$ such that $\vdash \overline{h} : \forall x.\mathcal{A}x \to \mathcal{A}hx$.

The following derivations are easy to obtain:

$$\vdash \lambda x\lambda y\lambda z.\overline{h}(\overline{\mathsf{head}}z) : \quad \forall X.\mathcal{S}_\mathcal{A} \subseteq X \to (\forall x.\mathcal{S}_\mathcal{A}x \to X\mathsf{maphd}_h x) \to$$

$$(\forall x.\mathcal{S}_\mathcal{A}x \to \mathcal{A}^{\mathsf{head}}\mathsf{maphd}_h x)$$

$$\vdash \lambda x\lambda y\lambda z.x(\overline{\mathsf{tail}}z) : \quad \forall X.\mathcal{S}_\mathcal{A} \subseteq X \to (\forall x.\mathcal{S}_\mathcal{A}x \to X\mathsf{maphd}_h x) \to$$

$$(\forall x.\mathcal{S}_\mathcal{A}x \to X^{\mathsf{tail}}\mathsf{maphd}_h x)$$

Therefore by $(M\nu I^+)$ we get

$$\vdash \mathsf{MCoRec}_2\big(\lambda x\lambda y\lambda z.\overline{h}(\overline{\mathsf{head}}z)\big)\big(\lambda x\lambda y\lambda z.x(\overline{\mathsf{tail}}z)\big) : \forall x.\mathcal{S}_\mathcal{A}x \to \mathcal{S}_\mathcal{A}\mathsf{maphd}_h x$$

and

$$\overline{\mathsf{maphd}_h} := \mathsf{MCoRec}_2\big(\lambda x\lambda y\lambda z.\overline{h}(\overline{\mathsf{head}}z)\big)\big(\lambda x\lambda y\lambda z.x(\overline{\mathsf{tail}}z)\big)$$

is a program for maphd_h.

Wir behalten von unseren Studien am Ende doch nur das, was wir praktisch anwenden.

Johann Wolfgang von Goethe (1749-1832)

7 Conclusions and Future Work

7.1 Conclusions

The initial goal of this project was to solve the following problem left open in [Mat98] (p. 178), I quote:

"One should work out modified realizability for monotone inductive definitions using the term language of systems of monotone inductive types and prove soundness of this interpretation. (For interleaving positive inductive definitions without "extended induction" and without second-order universal quantification this is sketched in [Ber95].)"

When doing the initial research I was pointed to the work by Krivine and Parigot ([KrPa90, Par92]). After reading these papers I was fascinated with the programming with proofs paradigm and decided to pursuit something in this direction too. The result is this thesis, which contributions are now stated with details:

- ◦ Inspired by [Mat98, Mat99] and [Hag87a] we formulate the following extensions of system F including the following (co)inductive types and principles:
  - MICT. Traditional (co)inductive types, conventional (co)iteration, conventional (co)recursion and (co)inductive inversion.
  - MCICT. Clausular (co)inductive types, conventional (co)iteration, conventional (co)recursion and (co)inductive inversion.

  - $\mathsf{MCICT}_{\boldsymbol{M}}$ Clausular (co)inductive types, Mendler-style (co)iteration, Mendler-style (co)recursion and coinductive inversion.
  - $\mathsf{MCICT}_{\boldsymbol{\mu M\nu}}$. Clausular (co)inductive types, conventional iteration, Mendler-style coiteration, conventional recursion, Mendler-style corecursion and coinductive inversion principles.

  all systems use full-monotonicity witnesses and are type-preserving and strongly normalizing.

- I introduce a concept of monotone and clausular inductive definition which syntactically simplifies the way to define predicates and the monotonicity witnesses required. This concept was initially inspired by Berger's unpublished draft [Ber95] and by Hagino's categorical type system [Hag87a]. Due to the clausular feature coinductive definitions are easily obtained by dualizing.

- Using the Curry-Howard correspondence I introduce a logic MCICD, corresponding to the type system MCICT, which extends the second-order logic AF2 with monotone and clausular (co)inductive definitions. The duality between inductive and coinductive definitions allows to get coinductive predicates by means of its destructors. In my opinion this is the most natural way to define sets coinductively, an important difference with [Raf94] where coinductive definitions are obtained via constructors.

- Based on the semantic notion of type in [KrPa90, Par92] I define a syntactical notion of realizability where first-order universal formulas do not have a computational content. Moreover, based on [Ber95] and [Tat93], I extend the realizability interpretation to (co)inductive definitions in a non-reductive way, i.e., the definition of realizability for (co)inductive predicates is again (co)inductive, using as target language the system MCICT of clausular (co)inductive types.

- Although the use of η-rules destructs the subject-reduction of the type systems I still study this kind of rules, which in the case of (co)inductive types guarantee the uniqueness of the initial algebra and final coalgebra, as far as the computational aspect is concerned.
  This additional study of the type systems pays off allowing to obtain the first functor law for canonical monotonicity witnesses via $\beta\eta$-reductions. As this fact guarantees the validity of the first functor law in the canonical model of the logic, using some instances of the first functor law as equations in the logic I was able to obtain a realizability soundness theorem where both source and target logics differ essentially only on the underlying object-term system and on the equational theory. This is an important improvement with respect to the system in [Tat94].

- With respect to data types, the use of clauses allows to prove in an easy way that some usual inductive predicates are formal data types, and therefore are suitable to program with them. On the other hand the weakness of

Leibniz' equality forbids to prove that the coinductive predicate of streams is a formal data type. This problem of equality for infinite datatypes is solved by means of a concept of datatype with equality, solution inspired by [Raf94].

- ◦ The problems arised while trying to program with coinductive predicates in MCICD using conventional coiteration are solved by means of a new logic $\mathsf{MCICD}_{\boldsymbol{\mu M\nu}}$, corresponding to the type system $\mathsf{MCICT}_{\boldsymbol{\mu M\nu}}$, which includes conventional induction principles and Mendler-style coinduction principles. As the only reason to include disjunctions and existentials as primitive formula constructors in the target logic for the realizability interpretation was to be able to define the conventional corecursion principles, we eliminate these conflictive formula constructors and therefore obtain a simpler realizability interpretation in comparison to the original interpretation for MCICD.

Although it was not, the original goal can be completely achieved with the tools developed in this thesis. However I think the realizability interpretation presented here offers more advantages than modified realizability, in particular the programming with proofs paradigm allows to extract programs from proofs without calculate a single realizer, an important improvement in comparison with [Tat93], for example. Moreover the extracted program is exactly the code for the original proof of the specification.

7.2 Related Work

Logical systems related to MCICD are presented in [Par92, Raf94, Tat93, Tat94, Uus98]. The system TTR sketched in [Par92] is an extension of AF2 with positive inductive definitions (called there "recursive types"), the associated proof-term system uses a fixed-point operator and is therefore non strongly normalizing, although some weak normalization results are stated. This paper also mentions some additional rules between them the ones for Mendler-style iteration and recursion. Based on [Par92], Raffalli presents in [Raf94] another extension of AF2, which includes not only inductive but also positive coinductive definitions but does not include primitive (co)recursion and we have again a fixed point operator within the proof-term system. On the other hand it includes a Läuchli-style realizability semantics based on the untyped lambda calculus.
In [Tat93] Tatsuta develops several extensions of Beeson's EON with positive inductive, monotone inductive and positive coinductive definitions, all three independent, there is no treatment of proof-terms and only partial terms of combinatory logic are used in a q-realizability interpretation which needs a fixed point operator to realize the induction axiom. In particular the system for coinductive definitions is not suitable for extracting programs about streams, which obliges the author to formulate a tailor-made solution.
[Uus98] presents several extensions of first order intuitionistic logic with positive inductive and coinductive definitions. The system MCICD could be seen as a

monotone version of a fusion of the systems $\mathcal{NI}_p(\mu, \nu)$ and $\mathcal{NI}_p(\mu^{\shortparallel}, \nu^{\shortparallel})$, but it uses minimal second-order logic.
The use of clauses to get inductive definitions is already present in [Ber95], this paper also sketches a modified realizability interpretation which inspires my definition of realizability for the case of (co)inductive predicates although mine is based in the semantic notion of type of [KrPa90, Par92], the main difference being that the first order universal quantifier does not have a computational content.
With respect to the type systems, MCICT is essentially a monotone version of the system developed in [Hag87a], but includes polymorphism and primitive (co)recursion and uses a natural deduction approach, following [Mat98, Mat99] very closely. This allows to establish a direct Curry-Howard correspondence between MCICT and MCICD. On the other hand systems of higher-order polymorphism including (co)iteration principles, useful for programming with nested data types have been developed in [AM03, AMU04].

7.3 Future Work

To finish this thesis I mention some problems left open, some of them are easily achieved by adapting the work done here, other are interesting open questions.

More Logics

With the results developed in this work we can formulate different versions of logics with (co)inductive definitions corresponding, for instance, to some of the logic systems of [Uus98] or to the type systems in [Mat98]. In particular positive versions of all systems presented here are easily definable. The immediate work is to define logics for the type systems $\mathsf{MCICT}_{\boldsymbol{M}}$ using only Mendler-style principles and MICT, involving traditional, i.e. non-clausular, (co)inductive definitions, a starting point for this last system is my paper [Mir02].

Subtyping and η-rules

It is well-known that η-rules cause the subject-reduction property to fail already in system F. However with some notion of subtyping this property is recovered (see [Mit88, Raf98, Raf99]). The goal is to formulate an adequate notion of subtyping such that $\mathsf{MCICT}\eta$ preserves the subject-reduction property. For a subtyping notion including (co)inductive types with approximations see [Ab03]. Of more interest is a notion of "subtyping" for the logic $\mathsf{MCICD}^{\exists}$ such that the rules in full Curry-style for existential formulas (see page 94) preserve the subject-reduction. The advantages of having a notion of subtyping in a logic can be seen, for example, in [Raf03].

On restricted formulas

Restricted formulas were introduced by Parigot in [Par92] with the purpose of hiding the computational content of some parts of a proof. I used them in this work only to define realizability for disjunctions (see page 104) mainly to avoid the occurrence of projections in the proof-terms. However later when proving the realizability of coinduction axioms the proof-terms obtained are anyway quite complicated due to the use of existential formulas in our framework. The goal is to find more useful and interesting applications of restricted formulas, like those described in appendix C of [Raf94].

The Inductive Inversion Rule

When developing the original version of MCICD I came out with the following rule for inductive inversion:

$$\dfrac{\begin{array}{l}\Gamma \vdash r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\vec{t} \\ \Gamma \vdash m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k\end{array}}{\Gamma \vdash \mathsf{in}_k^{-1}(\vec{m}, r) : \exists \vec{u}. \bigvee_{i=1}^{k} \Big(\mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)]\vec{u} \restriction \vec{t} = \vec{\mathbb{c}_i}\vec{u}\Big)}\ (\mu E^i)$$

This rule was left out later because it causes more problems than advantages, for example we would need to have existentials and restrictions on the source logic and it is not compatible (avoids the generation of neccesary redexes) with the existential rules. On the other hand its main application – to define inductive destructors – can be achieved with primitive recursion.
The goal is to define a better rule for inductive inversion. Observe that the rule would work better if the existential rules in full Curry-style, given in page 94, were available.

Improvements on the Definition of Clause

An unpleasant technicality in this work is the presence of global constructors, inherited from the category theory intuition. It would be nice to generalize the defining mechanism to allow constructors of different arity in each clause of a (co)inductive definition, in this way we could get rid of global constructors and have, for example, 0 as constructor of arity 0 and s of arity 1 in the definition of natural numbers.
In other direction, what advantages would bring a generalization of the concept of clause with several defining predicates ? that is clauses with the form $\langle \mathcal{F}_1, \ldots, \mathcal{F}_n, \mathbb{c}_1, \ldots, \mathbb{c}_m \rangle$. An application of this kind of clause would be an inductive definition of the product of predicates as: $\mathcal{A} \times \mathcal{B} := \mu X^{(2)}.\Big(\langle \mathcal{A}, \mathcal{B}, \mathsf{pair} \rangle\Big)$, with pair a binary constructor. The immediate condition here would be that the sum of the arities of all constructors in a clause has to coincide with the arity of the variable X.

Conservativity of Subject Reduction

Every system in this work posses the subject-reduction property. However I only developed the direct proof for the most complex system, namely $\mathsf{MCICD}^{\star}$, and argue that the proofs for simpler systems can be obtained by simplifying that proof. The goal is to find a general method to guarantee the inheritance of subject-reduction, something similar to the embeddings to prove strong normalization. In particular the method should guarantee that a subsystem of a given system inherits the subject-reduction property.

Semantics

I have used a classical tarskian semantics for the systems in this work. Indeed the satisfiability definition for (co)inductive predicates is a reductive one. The goal is to analyze the advantages of an intuitionist semantics, given directly by the realizability interpretation, i.e. a semantics in Läuchli style as the one presented in [Raf94]

Simultaneously Defined Predicates

The goal here is to extend our defining mechanism to include simultaneous definitions, for example trees $\mathcal{T}$ and tree lists $\mathcal{L}_{\mathcal{T}}$, informally defined with the following closure axioms, where $\mathsf{leaf}, \mathsf{nil}$ are 0-ary constructors, branch is unary and tcons is binary:

$$\mathcal{T}(\mathsf{leaf})$$

$$\mathcal{L}_{\mathcal{T}}(\mathsf{nil})$$

$$\forall x.\mathcal{L}_{\mathcal{T}}x \to \mathcal{T}\mathsf{branch}\,x$$

$$\forall x \forall y.\mathcal{T}x, \mathcal{L}_{\mathcal{T}}y \to \mathcal{L}_{\mathcal{T}}\mathsf{tcons}\,x\,y$$

For inductive predicates in free-algebra style this has been done in chapter five of [Sch04].

Inductive Predicates as Free Algebras

In which way is related the approach to inductive definitions via free-algebras (see [Sch04], chapter 5), implemented in the MINLOG system (`http://www.minlog-system.de/`) with mine? Does there exist an approach to coinductive definitions from free-algebras?

Implementation

The goal is to implement the method of program extraction for the system $\mathsf{MCICD}_{\mu M\nu}$ (recall that MCICD has problems with coinductive programming). Pointers in this direction are the systems ProPre described in [MPS92], which implements Parigot's TTR and SKIL reported in [GaHe93], which implements only AF2. For an strategy for proving termination of functions defined by recursive equations implemented in ProPre see [MaSi95], whereas a proof search strategy for AF2 can be found in [GaHe96].

Proof-Theoretical Analysis

From the proof-theoretical point of view it is of interest to establish the strength of the theory MCICD as well as the relationships with traditional systems of inductive definitions. The standard reference is [BFPS81].

Extensions to Higher Order Logic

Would it be useful to extend my approach to (co)inductive definitions to higher-order logic? From the type-theoretical perspective, systems of higher-order polymorphism, extending F^{ω} with several (co)iteration schemes, are useful to handle nested data types (see [AM03, AMU04]) and would serve as systems of realizers.

Systems for Course of values (Co)induction

In [Uus98] Uustalu develops logics $\mathcal{NI}_p(\mu^\star, \nu^\star)$ and $\mathcal{NI}_p(\text{M}^\star, \text{N}^\star)$ for conventional and Mendler-style course of values (co)induction. The goal is to extend system F as well as AF2 with similar principles.

Systems with both conventional and Mendler-style (co)induction

The system $\mathsf{AF2}^{\mu\nu}$ developed in [Raf94] has inference rules for conventional as well as for Mendler-style (co)iteration, the former ones being non-traceable. The goal is to formulate such a system and to analyze the advantages it brings.

Por ahí pasa la escalera espiral, que se abisma y se eleva hacia lo remoto. En el zaguán hay un espejo, que fielmente duplica las apariencias. Los hombres suelen inferir de ese espejo que la Biblioteca no es infinita (si lo fuera realmente, ¿ a qué esa duplicación ilusoria ?); yo prefiero soñar que las superficies bruñidas figuran y prometen el infinito ...

Jorge Luis Borges, La Biblioteca de Babel.

Bibliography

[Ab03] A. Abel. Termination and Productivity Checking with Continuous Types. In M.Hofmann. Ed. *Typed Lambda Calculi and Applications, 6th International Conference, TLCA 2003,* Valencia, Spain. LNCS **2701** Springer Verlag 2003.

[AM03] A.Abel, R. Matthes. (Co-)iteration for higher-order nested datatypes. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs, International Workshop, TYPES 2002.* LNCS **2646**, pages 1-20, Berg en Dal, The Netherlands. Springer Verlag 2003.

[AMU04] A. Abel, R. Matthes, T. Uustalu. Iteration and Coiteration Schemes for Higher-Order Nested Datatypes. Accepted for publication in *Theoretical Computer Science.* Elsevier 2004.

[Bar93] H. Barendregt. Lambda Calculi with Types. In S. Abramski, D. M. Gabbay, T. S. E. Maibaum, editors.*Handbook of logic in Computer Science, Vol. 2 Background: Computational Structures.* Oxford University Press 1993.

[Bar97] H. Barendregt. The Impact of Lambda Calculus in Logic and Computer Science. *Bulletin of Symbolic Logic* **3**(2). pp 181-214. Association for Symbolic Logic 1997.

[Ben98] Holger Benl. Konstruktive Interpretation induktiver Definitionen. (Constructive Interpretation of Inductive Definitions) (In German). Diplomarbeit, Mathematisches Institut der LMU München. June 1996.

[Ber93] Ulrich Berger. Program Extraction from normalization proofs. In M. Bezem and J.F. Groote, editors. *Typed Lambda Calculus and Applications.* LNCS **664**, Springer Verlag. 1993.

[Ber95] Ulrich Berger. A constructive interpretation of positive inductive definitions. Unpublished Draft. March 1995.

[BBS02] U. Berger, W. Buchholz, H. Schwichtenberg. Refined Program Extraction from Classical Proofs. In *Annals of Pure and Applied Logic* **114**(1-3), pp. 3-25. Elsevier Science B.V. April 2002.

[Ber97] C. Berline. A presentation of the Curry-Howard Correspondence. Unpublished note available via `http://www.pps.jussieu.fr/~berline/Cur-How.ps`

[BFPS81] W. Buchholz, S. Feferman, W. Pohlers, W. Sieg. Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies. LNM **897**, Springer Verlag, 1981.

[Cro93] R.L. Crole. Categories for Types. Cambridge Mathematical Textbooks. Cambridge University Press, 1993.

[DM93] H. Dybkjær, A. Melton. Comparing Hagino's Categorical Programming Language and Typed Lambda-Calculi. *Theoretical Computer Science* **111** pp. 145-189. Elsevier 1991.

[GaHe93] D. Galmiche, O. Hermann. SKIL: A System for Programming with Proofs. In *LPAR'93, International Conference on Logic Programming and Automated Reasoning*, LNAI **698**. Springer Verlag 1993.

[GaHe96] Proof search and induction choices in AF2 system D. Galmiche and O. Hermann. Technical report, march 1996. A preprint is available in `http://ww.loria.fr/~galmiche/oldpapers.html`

[Geu92] H. Geuvers. Inductive and coinductive types with iteration and recursion. In B. Nordström, K. Petterson, G. Plotkin, Eds. *Proceedings of the 1992 Workshop on Types for Proofs and Programs* Båstad, Sweden June 1992, pp. 183-207. Available Via `http://www.cs.kun.nl/~herman/BRABasInf_RecTyp.ps.gz`.

[Gir72] J.Y. Girard. Interprétation fonctionelle et élimination des coupures dans l'arithmét ique d'ordre supérieur. Thèse de Doctorat d'État, Université de Paris VII. 1972.

[GLT89] J.Y. Girard, Y. Lafont, P. Taylor. Proofs and Types. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press 1989.

[Gre92] J. Greiner. Programming with Inductive and Co-Inductive Types. Technical Report CMU-CS-92-109, Carnegie-Mellon University. January 1992

[Hag87a] T. Hagino. A Typed Lambda Calculus with Categorical Type Constructors. In D.H. Pitt, A. Poigné, D.E. Rydeheard. *Category Theory and Computer Science*. LNCS **283** Springer Verlag 1987.

[Hag87b] T. Hagino. A Categorical Programming Language. Ph.D. Thesis CST-47-87 (also published as ECS-LFCS-87-38). Department of Computer Science, University of Edinburgh 1987.

[Ho92] B.T. Howard. Fixed Points and Extensionality in Typed Functional Programming Languages. Ph. D. Thesis, Stanford University 1992. Available via `http://www.cis.ksu.edu/~bhoward/ftp/sudiss.ps.Z`

[Ho80] W.A. Howard. The Formulae-as-Types Notion of Construction. In J.P. Seldin and J.R. Hindley, editors. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* pp. 479–490. Academic Press 1980.

[JaRu97] B. Jacobs, J. Rutten. A Tutorial on (Co)Algebras and (Co)Induction. EATCS Bulletin 62. p. 222-259. 1997.

[KrPa90] J.L. Krivine, M. Parigot. Programming with Proofs. In *Journal of Information Processing and Cybernetics EIK (Formerly Elektronische Informationsverarbeitung und Kybernetik)* **26**(3) pp. 149-167. 1990.

[Kri93] J.L. Krivine. Lambda-Calculus, Types and Models. Ellis Horwood Series in Computers and their Applications. Ellis Horwood, Masson 1993.

[Lei83] D. Leivant. Reasoning about Functional Programs and Complexity Classes associated with Type Disciplines. *Proceedings of 24th Annual Symposium on Foundations of Computer Science* pp.460-469 IEEE Computer Science Press. 1983.

[MPS92] P. Manoury, M. Parigot, M. Simonot. ProPre A Programming Language with Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning LPAR 92*. LNAI **624** Springer Verlag 1992.

[MaSi95] P. Manoury, M. Simonot. Automatizing Terminations Proofs of Recursively Defined Functions. In *Theoretical Computer Science* **135**, Elsevier 1995.

[Mac98] S. Mac Lane. Categories for the Working Mathematicioan. 2nd. Edition. Vol. 5. Graduate Texts in Mathematics, Springer Verlag 1998.

[Mat98] Ralph Matthes, Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types, Dissertation Universität München, 1999. Available via `http://www.tcs.informatik.uni-muenchen.de/~matthes/dissertation/matthesdiss.ps.gz`

[Mat99] Ralph Matthes. Monotone (co)inductive types and positive fixed-point types. In *Theoretical Informatics and Applications* 33(4-5) pp. 309-328. EDP Sciences. 1999.

[Mat01] Ralph Matthes. Parigot's second order lambda-mu-calculus and inductive types. In *Samson Abramsky, editor, Proceedings of TLCA 2001*, volume 2044 of Lecture Notes in Computer Science, pages 329-343. Springer Verlag, 2001.

[Men87] N.P. Mendler. Recursive Types and Type Constraints in Second-Order Lambda Calculus. In *Proceedings of the 2nd Annual Symposium on Locig in Computer Science,Ithaca N.Y.* pp. 30-36 IEEE Computer Society Press, Washington D.C. 1987.

[Men91] N.P. Mendler. Inductive Types and Type Constraints in the Second-Order Lambda Calculus. *Annals of Pure and Applied Logic* **51**(1-2) pp. 159-172. North-Holland 1991.

[Mir02] F.E. Miranda Perea. A Curry-Style Realizability Interpretation for Monotone Inductive Definitions. In Malvina Nissim, editor. *Proceedings of the 7th. ESSLLI Student Session.* Trento Italy 2002.

[Mit88] John C. Mitchell. Polymorphic Type Inference and Containment. Information and Computation **76** pp. 211-249. 1988.

[Par92] M. Parigot, Recursive programming with proofs. In *Theoretical Computer Science* **94**, pp.335-356. Elsevier. 1992.

[PZ01] E. Poll, J. Zwanenburg. From Algebras and Coalgebras to Dialgebras. In *Coalgebraic Methods in Computer Science (CMCS'2001).* Electronic Notes in Theoretical Computer Science **44**. Elsevier, 2001.

[Raf94] C. Raffalli. L' Arithmétique Fonctionnelle du Second Ordre avec Points Fixes, Thèse de l'Université Paris VII. 1994. Available via `http://www.lama.univ-savoie.fr/~RAFFALLI/`

[Raf98] C. Raffalli. Type Checking in System F^η. Unpublished draft. 1998. Available via `http://www.lama.univ-savoie.fr/~RAFFALLI/`

[Raf99] C. Raffalli. An Optimized Complete Semi-Algorithm for system F^η. Unpublished draft. Available via `http://www.lama.univ-savoie.fr/~RAFFALLI/`

[Raf03] C. Raffalli. System ST, Toward a Type System for Extraction and Proofs of Programs. Annals of Pure and Applied Logic **122**(1–3), pp. 107-130. Elsevier 2003.

[Rey74] J. C. Reynolds. Towards a Theory of Type Structure. In B. Robinet, editor. *Programming Symposium*, LNCS **19**. Springer Verlag 1974.

[Sch04] H. Schwichtenberg. Minimal Logic for Computable Funtionals. Unpublished notes from january 2004. Available via `http://www.mathematik.uni-muenchen.de/~minlog/minlog/mlcf.ps`

[Tat93] M. Tatsuta, Realizability of Inductive Definitions for Constructive Programming. PhD Thesis, University of Tokyo, 1993.

[Tat94] M. Tatsuta, Two Realizability Interpretations of Monotone Inductive Definitions. In *International Journal of Foundations of Computer Science* **5**(1), pp. 1-21. 1994.

[Tho91] S. Thompson. Type Theory and Functional Programming. Addison-Wesley International Computer Science Series. 1991.

[Tro98] A.S. Troelstra, Realizability. In S.R. Buss, editors. *Handbook of Proof Theory.* Elsevier, 1998.

[Urz99] P. Urzyczyn. The Curry-Howard Isomorphism: Remarks on Recursive Types. Lecture Notes for the EEF Trends School in Logic and Computation, Heriot-Watt University, Edinburgh, April 1999. Available via `ftp://ftp.mimuw.edu.pl/People/urzy/edynburg.ps.gz`

[UV99] T. Uustalu, V Vene. Mendler-style Inductive Types, categorically. In *Nordic Journal of Computing* **6**(3), pp. 343-361, 1999.

[UV00] T. Uustalu, V. Vene. Coding Recursion á la Mendler (extended abstract). In J. Jeuring, ed. *Proc. of 2nd Workshop on Generic Programming WGP 2000.* Technical Report UU-CS-2000-19, Dept. of Computer Science, Utrecht University pp. 69-85. 2000.

[Uus98] T Uustalu. Natural deduction for intuitionistic least and greatest fixedpoint logics, with an application to program construction (PhD thesis). Dissertation TRITA-IT AVH 98:03, Dept. of Teleinformatics, Royal Inst of Technology (KTH), Stockholm, 1998.

[Wra89] G.C. Wraith. A note on categorical datatypes. In D.Pitts et al, editors. Category Theory and Computer Science. LNCS **389**, Springer Verlag 1989.

Symbol Index

A°, 98
$\mathsf{AF2}$, 24
$\mathsf{AF2}^{\wedge,\vee}$, 29
$\mathcal{A} \to \mathcal{B}$, 149
$\mathcal{A} + \mathcal{B}$, 149
$\mathcal{A} \times \mathcal{B}$, 149
$A[\vec{X} := \vec{\mathcal{F}}\,]$, 25
$A[\vec{x} := \vec{s}\,]$, 24

$\mathcal{C}_{\Gamma,\mathbb{E}}(A)$, 97
$\mathbb{C}_i^k$, 68, 104
$\vec{\mathbb{c}_i}\vec{t}$, 80
$\mathsf{cl}(M)$, 17
$\mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),i}$, 83
$\mathsf{CoCl}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k),i}$, 84
$\mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, 84
$\mathsf{CoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, 84
$[f, g]$, 38
$\mathcal{C}_i^{\mathtt{r}}$, 104

$\vdash^{\mathsf{can}}$, 89
$\mathbb{D}_i^k$, 69, 104
$\mathcal{D}_i^{\mathtt{r}}$, 104
$D_{\mathcal{T}}$, 130

E, 16
e, 15
$\mathbb{E}_\beta$, 95
$\mathbb{E}^\star(s)$, 121
$\mathbb{E}$, 26
$=$, 27
$E[r]$, 16

F, 12
$\mathcal{F}\,\mathsf{mon}^-\,X$, 87
$\mathbb{FFL}(s)$, 121
$\mathcal{F} \subseteq \mathcal{G}$, 81
$\mathcal{F}\,\mathsf{mon}\,X$, 81
$\mathcal{F}^\nu$, 133

$\overline{f}$, 150
$\mathcal{F} \vee \mathcal{G}$, 81
$\mathcal{F}^{\mathtt{r}}$, 104
$\mathsf{F}^{+,\times}$, 15
$FV(A)$, 24
$FV(\sigma)$, 12
$FV(t)$, 13, 24
$\mathcal{F} \wedge \mathcal{G}$, 81

Γ, 26
$\Gamma_\alpha^\downarrow$, 159
$\Gamma \vdash_{\mathbb{E}} t : A$, 26
$\Gamma[\gamma := \chi]$, 96
$\Gamma^{\mathtt{r}}$, 121
$\Gamma_\alpha^\uparrow$, 160
$\Gamma[\vec{y}/\vec{x}\,]$, 96

$\mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$, 83
$\mathsf{Ind}^{+}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$, 83
$\mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, 84

$\mathcal{K}^{\vec{\mathbb{c}_i}}$, 81

$\lambda\alpha\rho$, 31
$\lambda\vec{y}F$, 25
$\mathcal{L}_{\mathcal{A}}$, 149

MCICD, 83
$\mathsf{MCICD}_{\boldsymbol{\mu M \nu}}$, 162
$\mathsf{MCICD}^\star$, 93
MCICT, 55
$\mathsf{MCICT}\eta$, 64
$\mathsf{MCICT}_{\boldsymbol{M}}$, 75
MCICT^-, 55
$\mathsf{MCICT}_{\boldsymbol{\mu M \nu}}$, 77
$\mathsf{MCoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, 162
$\mathsf{MCoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$, 162
$\mathfrak{M}$, 144
MICT, 37

$\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$, 81

$\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$, 81
$\nu \models_{\mathcal{M}} A$, 133

$\Pi[\gamma := \chi]$, 96
$\langle f, g \rangle$, 38

$\rho[\vec{\alpha} := \vec{\sigma}\,]$, 12
$\|\, r \,\|$, 130
$\mapsto_\beta$, 13
$\to_\beta$, 13
$\rhd$, 13
$\rho \, \mathsf{mon} \, \alpha$, 32
$\widetilde{r}$, 120, 164

SAT, 16
$\mathsf{SC}^\rho[\Gamma]$, 20
$s = t$, 27
$\mathbb{\Sigma}$, 13
$\mathbb{\Sigma} \rhd t : \rho$, 13
SN, 16
sn, 21
$\star$, 15
$\mathcal{S}_{\mathcal{A}}$, 150
$\mathsf{S}_x(\mathcal{M}, \mathcal{N})$, 17

$t^{\mathfrak{D}}[\nu]$, 131
$t \; \mathtt{r} \; A$, 93, 104
$t[\vec{x} := \vec{s}\,]$, 13, 24

$\mathcal{W}(t)$, 121

$X \, \mathsf{admis} \, \mathcal{F}$, 160
X^+, 104

Index

Algebra, 1
  initial, 2
  recursive, 3
Antimonotonicity, 65, 87
Antimonotonicity witness
  typing rules for, 66, 88
Applicative structure, 128
Axiom
  closure, 83
  coclosure, 84
  coinduction, 84
    Mendler-style, 162
    strong, 84
  induction, 83
    strong, 83
  Realizability for, 106

Candidate assignment, 19
Church numerals, 13
Clause, 52, 80
Coalgebra, 1
  final, 2
  corecursive, 4
Coinduction, 1
  á la Mendler, 159
Coinductive predicate, 81
Coinductive set, 80
Coiteration
  Mendler-style, 161
Comprehension
  predicate, 25
Conservation lemma, 142
Constructor, 81
Continuity lemma, 165
Corecursion, 35
  Mendler-style, 161
Correctness lemma, 143
  for $\mathsf{MCICD}_{\mu M\nu}$, 170
  for data types with equality, 174

Data types, 142
  examples, 145
  function space of, 149
  product of, 149
  sum of, 149
  with equality, 170, 173
Degenerated type, 56
Destructor, 81
Dialgebra, 9
Downward hierarchy, 159
  semantical, 166

Elimination, 15
  multiple, 16
Embedding, 23
Equation, 27
E-term, 100
Existential formula, 94
Extensionality, 129

First functor law, 67, 89
Fixed point, 79
  coinductive, 80
  greatest, 80, 160
  inductive, 80
  least, 80, 160
Formula
  existential, 94
  open, 98
  restricted, 94

Greatest fixed point, 80, 160
  hierarchy, 159

Induction, 1
Inductive predicate, 81
Inductive set, 80
Instances
  $\Gamma, \mathbb{E}$-, 97

Inversion
  coinductive, 35
  inductive, 33, 34
Iteration, 33
I-term, 100

Kan extension, 76

Least fixed point, 80, 160
  hierarchy, 160
Leibniz Equality, 27
Lemma
  coincidence, 20, 49
  conservation, 142
  continuity, 165
  correctness, 143, 170
  generation, 100
  main
    for strong normalization, 20, 50
    for subject reduction, 99
  substitution, 20, 50
Logic
  second order, 24

Model
  canonical, 144
  extensional, 129
  identity, 134
  intended, 144
  syntactical, 128
Modified realizability, 143
Monotone operator, 79
Monotonicity witness, 32
  canonical, 65, 89
  generic, 65, 87
  on-display, 121
  typing rules for, 66, 88
Morphism
  of F, G-dialgebras, 9
  of T-M-algebras, 6
  of T-M-coalgebras, 8
  of algebras, 1
  of coalgebras, 2
Multiple elimination, 16

Natural Numbers
  in AF2, 27
  in F, 13
  in MCICD, 85
  in MCICT, 55
  in MICT, 38

Open formula, 98
Operator, 79
  monotone, 79

Post-fixed point, 79
Pre-fixed point, 79
Predicate, 25
  coinductive, 81
  comprehension, 25
  inductive, 81
  of strong computability, 19
Principle
  of coinductive inversion, 2, 35, 54
    on dialgebras, 11
  of coiteration, 2, 34, 36
    in Mendler-style, 8, 12, 36
    on dialgebras, 10
  of inductive inversion, 2, 33, 34, 54
    on dialgebras, 11
  of iteration, 2, 33
    in Mendler-style, 6, 12, 35, 36
    on dialgebras, 11
  of primitive corecursion, 5, 35, 36
    in Mendler-style, 9, 12, 36
    on dialgebras, 11
  of primitive recursion, 5, 33
    in Mendler-style, 7, 12, 36
    on dialgebras, 11
Proof-term, 24, 26

Realizability, 93
  modified, 143
  semantical soundness, 142
  soundness theorem, 120
Recursion, 33
Reduction rules

β, 13, 38, 54, 75, 83, 94
η, 62
Restricted formula, 94
Rule
non-traceable, 14, 27, 99
traceable, 27
Rules
for monotonicity witnesses, 88

Satisfaction, 133
Saturated closure, 17
Saturated set, 16
for coinductive types, 44
for inductive types, 40
properties, 18, 48
Set
coinductive, 80
inductive, 80
saturated, 16
SN-method, 15
Soundness
of term interpretation, 130
of the logic MCICD*, 136
of the logic MCICD$^{\star}_{\mu M\nu}$, 169
Streams
in AF2, 27
in F, 14
in MCICD, 85
in MCICT, 55
in MICT, 39
Strong computability
predicate, 19
for coinductive types, 49
for inductive types, 49
Strong Normalization
for AF2, 29
for AF2$^{\wedge,\vee}$, 30
for F, 14
for F$^{\exists}$, 23
for F$^{+,\times}$, 15
for MCICD*, 95
for MCICD$_{\mu M\nu}$, 162
for MCICT, 57
for MCICT$_{M}$, 76
for MICT, 39
inheritance of, 23
Strong normalization, 14
Subject Reduction, 14
for AF2, 29
for AF2$^{\wedge,\vee}$, 30
for F, 14
for F$^{\exists}$, 22
for F$^{+,\times}$, 15
for MCICD, 86
for MCICD*, 104
for MCICD$_{\mu M\nu}$, 162
for MCICT, 55
for MICT, 38
Substitution
properties, 25, 104, 133
for derivations, 96
Surjective pairing, 63
System
AF2, 24
AF2$^{\wedge,\vee}$, 29
F, 12
F$^{\exists}$, 22
F$^{+,\times}$, 15
MCICD, 83
MCICD*, 93
MCICD$_{\mu M\nu}$, 162
MCICT$_{M}$, 75
MCICT$_{\mu M\nu}$, 77
MICT, 37

Tag, 80
T-algebra, 1
M-recursive, 7
recursive, 3
T-coalgebra, 1
M-corecursive, 8
corecursive, 4
Termination, 14
Theorem
Knaster-Tarski, 80
T-M-algebra, 5
T-M-coalgebra, 7
Type Preservation, 14

Upward hierarchy, 160

Valuation, 127, 132

Variable
  second order
    admissible, 160

Lebenslauf

Persönliche Daten

- Name: Favio Ezequiel Miranda Perea
- Geburtsdatum: 20.12.1972
- Geburtsort: Mexiko Stadt, Mexiko.
- Staatsangehörigkeit: mexikaner
- Familienstand: ledig

Schulbildung

- 1979-1984 staatliche Grundschule:
  Escuela Primaria "Ignacio Ramirez", Mexiko Stadt
- 1984-1987 staatliche Sekundarschule (Zwischenstufe):
  Escuela Secundaria Diurna No. 36 "Cuauhtemoc", Mexiko Stadt
- 1987-1990 staatliche Oberschule (Gymnasium):
  Escuela Nacional Preparatoria No. 6 "Antonio Caso", Mexiko Stadt

Studium

- 1990-1995 Studium der Mathematik an der Wissenschafts Fakultät der Nationale Autonom Universität Mexikos (UNAM).
  Abschluß: Licenciatura (Bachelor of Science).
- 1997-2000 Studium der Mathematik an der Wissenschafts Fakultät der Nationale Autonom Universität Mexikos (UNAM).
  Abschluß: Maestría en Ciencias (Master of Science)
- 2000-2004 Promotionsstudium der Mathematik an der Ludwig-Maximilians Universität München.

Beruf

- 1994-1997 Lehrauftrag an der Universität Mexiko als Assistent.
- 1997-2000 Lehrauftrag an der Universität Mexiko als Dozent.
- 2000-2004 assoziierter Mitglied des GKLI (Wissenschaftliche Mitarbeiter am LFE Theoretische Informatik, Institut für Informatik LMU München)