Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 123 (2005) 179–193
www.elsevier.com/locate/entcs

Realizability for Monotone and Clausular (Co)inductive Definitions

Favio E. Miranda-Perea [1,2]

Institut für Informatik
Ludwig-Maximillians-Universität
Oettingenstr. 67, D-80538
München, Germany

Abstract

We develop an extension of second order logic (AF2) with monotone, and not only positive, (co)inductive definitions and a clausular feature which simplifies considerably the defining mechanism. A sound realizability interpretation, where the extracted programs are untyped, but typable, terms of a strongly normalizing Curry-style system of monotone (co)inductive types makes our logic into a logical framework suitable for programming with proofs.

Keywords: second-order logic AF2, realizability, monotone (co)inductive definitions, system F, monotone (co)inductive types, programming with proofs.

1 Introduction

During the last decade several authors have studied logical systems of (co)inductive definitions and given realizability interpretations with purposes of program extraction ([9],[11],[10],[12]). We continue this line of research extending second-order logic and focusing on simplicity in the mechanism for defining (co)inductive objects, achieved by the use of clauses, as well as in a deep study of the term system of realizers. For sake of generality we consider monotone, and not only positive, inductive definitions —after all positivity is only used to ensure monotonicity. This choice allows to obtain shorter and simpler proofs

[1] This research is being supported by grant 154186 of the CONACYT-DAAD agreement
[2] Email: miranda@informatik.uni-muenchen.de

doi:10.1016/j.entcs.2004.04.048

as we do not have to calculate specific monotonicity witnesses, we only require the presence of a proof of monotonicity which might even be only an extra assumption. Moreover the study of full monotonicity shows what is needed to obtain a realizability interpretation where both, source and target logics, are structurally the same, the essential difference being the need of some extra equations involving monotonicity witnesses, namely those expressing the first functor law. Apart from the definition of a logic with clauses and full monotonicity the main contribution of this paper is the formulation of a sound realizability interpretation where the realizability of (co)inductive definitions is not reductive, like those in [9] and [10], but again a (co)inductive definition. In section 2 we extend second order logic (AF2) with monotone and clausular (co)inductive predicates and state two important properties of the system, namely subject reduction and strong normalization. The realizability interpretation and its soundness theorem are developed in section 3. Related work as well as conclusions and further work are provided in sections 4 and 5.

2 (Co)inductive Definitions with Tags

We extend second order logic (precisely Krivine's and Leivant's AF2 [4]) with (co)inductive predicates. For simplicity we include $\wedge, \vee$ as primitives in AF2. Recall that AF2 works with Leibniz' equality defined as $r = s := \forall X.Xr \to Xs$. Therefore the derivation relation $\vdash$ depends not only on a context of formulas but also on a given context of equalities $\mathbb{E}$, the judgements of the logic are then of the form $\Gamma \vdash_{\mathbb{E}} r : A$. For comprehension predicates we use the notation $\lambda \boldsymbol{y} F$ where F is a formula, the arity of $\lambda \boldsymbol{y} F$ is the length of the vector $\boldsymbol{y}$ and $(\lambda \boldsymbol{y} F)\boldsymbol{t}$ means $F[\boldsymbol{y} := \boldsymbol{t}\,]$.

The concept of clause will allow to simplify the way of defining (co)inductive predicates.

Definition 2.1 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 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 $\mathcal{C}_i, \mathcal{D}_j$. If $\mathcal{C}_i = \langle \mathcal{F}_i, \mathbb{c}_1^i, \ldots, \mathbb{c}_m^i \rangle$ we set $\boldsymbol{c_i} := \mathbb{c}_1^i, \ldots, \mathbb{c}_m^i$, and if $\boldsymbol{t} := t_1, \ldots, t_m$, we define $\boldsymbol{c_i t} := \mathbb{c}_1^i t_1, \ldots, \mathbb{c}_m^i t_m$.

Definition 2.2 An expression of the form $\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$, where $\mathcal{C}_i := \langle \mathcal{F}_i, \boldsymbol{c_i} \rangle$ 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 coinduc-

tive 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 closure operator generated by $\mathcal{F}_1 \vee \ldots \vee \mathcal{F}_k$ via the constructors $\boldsymbol{c_1}, \ldots, \boldsymbol{c_k}$. Analogously, $\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ represents the greatest fixed point of the support (coclosure) operator generated by $\mathcal{F}_1 \wedge \ldots \wedge \mathcal{F}_k$ via the destructors $\boldsymbol{c_1}, \ldots, \boldsymbol{c_k}$. The below inference rules will formalize this intuitive meaning.

The formula $\mathcal{F}\,\mathsf{mon}\,X := \forall X \forall Y. X \subseteq Y \rightarrow \mathcal{F} \subseteq \mathcal{F}[X := Y]$ [3] expresses the fact that the predicate $\mathcal{F}$ is monotone w.r.t the variable X, we use this full-monotonicity instead of the usual syntactical restriction of positivity following [5]. There are no syntactical restrictions to form a (co)inductive predicate. However there will be some restrictions to eliminate an inductive predicate or to introduce a coinductive predicate, namely we will require some monotonicity proofs.

The (co)inductive definitions are ruled by:

- Folding of the Least Fixed Point: for $1 \leq j \leq k$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : \mathcal{F}_j[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)]\boldsymbol{t}}{\Gamma \vdash_{\mathbb{E}} \mathsf{in}_{k,j}\, r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\boldsymbol{c_j t}} \ (\mu I)$$

- Iteration:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)\boldsymbol{r} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{F}_i[X := \mathcal{K}] \subseteq \mathcal{K}^{\boldsymbol{c_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{It}_k(\boldsymbol{m}, \boldsymbol{s}, r) : \mathcal{K}\boldsymbol{r}} \ (\mu E)$$

  where $\mathcal{K}^{\boldsymbol{c_i}} := \lambda \boldsymbol{y}.\mathcal{K}(\boldsymbol{c_i y})$.

- Primitive Recursion:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mu X.(\mathcal{C}_1, \ldots, \mathcal{C}_k)\boldsymbol{r} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge \mathcal{K}] \subseteq \mathcal{K}^{\boldsymbol{c_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{Rec}_k(\boldsymbol{m}, \boldsymbol{s}, r) : \mathcal{K}\boldsymbol{r}} \ (\mu E^+)$$

[3] We set, as usual, $\mathcal{F} \subseteq \mathcal{G} := \forall \boldsymbol{x}.\mathcal{F}\boldsymbol{x} \rightarrow \mathcal{G}\boldsymbol{x}$

- Coiteration:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mathcal{K}\boldsymbol{t} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{K} \subseteq \mathcal{F}_i[X := \mathcal{K}]^{\boldsymbol{c_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{Colt}_k(\boldsymbol{m}, \boldsymbol{s}, r) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\boldsymbol{t}} \ (\nu I)$$

- Primitive Corecursion:

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r : \mathcal{K}\boldsymbol{t} \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \mathsf{mon} X,\ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} s_i : \mathcal{K} \subseteq \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k) \vee \mathcal{K}]^{\boldsymbol{c_i}},\ \ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{CoRec}_k(\boldsymbol{m}, \boldsymbol{s}, r) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\boldsymbol{t}} \ (\nu I^+)$$

- Folding of the Greatest Fixed Point (Inversion):

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathbb{E}} r_i : \mathcal{F}_i[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)])\boldsymbol{c_i t},\ \ 1 \leq i \leq k \\ \Gamma \vdash_{\mathbb{E}} m_i : \mathcal{F}_i \,\mathsf{mon}\, X,\ 1 \leq i \leq k\end{array}}{\Gamma \vdash_{\mathbb{E}} \mathsf{out}_k^{-1}(\boldsymbol{m}, \boldsymbol{r}) : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\boldsymbol{t}} \ (\nu I^i)$$

- Unfolding of the Greatest Fixed Point: for $1 \leq j \leq k$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)\boldsymbol{t}}{\Gamma \vdash_{\mathbb{E}} \mathsf{out}_{k,j}\, r : \mathcal{F}_j[X := \nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)]\boldsymbol{c_j t}} \ (\nu E)$$

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), 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.

Some examples of (co)inductive predicates in our logic are the natural numbers $\mathbb{N} := \mu X\big(\langle \mathbb{1}, 0_{\mathsf{g}}\rangle, \langle X, s\rangle\big)$, where $\mathbb{1}$ is the unit predicate, inhabited by only one element, and the constructors $0_{\mathsf{g}}, s$ represent a global zero and the succesor function respectively, analogously $\mathcal{L}_{\mathcal{A}} := \mu X\big(\langle \mathbb{1}, \mathsf{nil}_{\mathsf{g}}\rangle, \langle \mathcal{A} \times X, \mathsf{cons}\rangle\big)$

represents lists of elements of $\mathcal{A}$, where the product predicate is coinductively defined as $\mathcal{P}\times\mathcal{Q} := \nu X\big(\langle\mathcal{P},\pi_1\rangle,\langle\mathcal{Q},\pi_2\rangle\big)$. The predicate of streams of elements of $\mathcal{A}$ is defined as $\mathcal{S}_{\mathcal{A}} := \nu X\big(\langle\mathcal{A},\mathsf{head}\rangle,\langle X,\mathsf{tail}\rangle\big)$. In comparison with previous systems ours allows to define (co)inductive objects in a very simply way due to the clausular feature, as the previous examples let it show.

The proof-reduction for (co)inductive definitions is given by the following β-reduction rules between proof-terms:

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

where $\langle\cdot,\cdot\rangle,[\cdot,\cdot]$ denote the usual pair and copair of functions respectively. 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 cannot be discussed here due to lack of space. If we consider in detail the rule for iteration, for example, we observe that the function $f := \lambda x.\mathsf{It}_k(\boldsymbol{m},\boldsymbol{s},x)$, which is the function being defined by iteration, when applied to the argument $\mathsf{in}_{k,i}\,t$, reduces to a term with access to the entire f. However, we get strong normalization because the typing of m controls the access to f —an instance of type-based termination. The described logical system will be called MCICD, a system of Monotone and Clausular Inductive and Coinductive Definitions.

2.1 Properties of MCICD

Our logic possess two important properties, namely subject-reduction (type preservation) and strong normalization. The type preservation: If $\Gamma\vdash r:A$ and $r\to_\beta s$ then $\Gamma\vdash s:A$, being MCICD a system in Curry-style, is not trivial and can be proven, for example, adapting Krivine's method for AF2 (see [4]). To prove the strong normalization of MCICD we embed it into a type system via a first-order forgetful map (and using $+,\times$ instead of $\vee,\wedge$, as usual). The resulting system will be called MCICT and it is an extension of system F with (co)inductive types of the form $\mu\alpha(\rho_1,\ldots,\rho_k),\nu\alpha(\rho_1,\ldots,\rho_k)$. A related system with only positive (co)inductive types of this shape and without polymorphism was developed in [2]. The above examples of (co)inductive predicates become (co)inductive types, for example, $\mathsf{nat} := \mu\alpha(1,\alpha),\mathsf{list}(\rho) :=$

$\mu\alpha(1, \rho \times \alpha), \mathsf{stream}(\rho) := \nu\alpha(\rho, \alpha)$. The strong normalization of MCICT is proved by embedding it into a system without clauses, like those of [6], so that the coinductive type $\nu\alpha(\rho_1, \ldots, \rho_k)$ embeds into the type $\nu\alpha.\rho_1 \times \ldots \times \rho_k$ and the inductive type $\mu\alpha(\rho_1, \ldots, \rho_k)$ is mapped to $\mu\alpha.\rho_1 + \ldots + \rho_k$. The strong normalization of this final system is proved directly via saturated sets.

3 Realizability

Realizability is a powerful mathematical tool to synthetize programs from, usually constructive, proofs. If a logical system is based on contructive logic and has a sound realizability interpretation we can extract a program and its verification proof effectively from a proof of the specification of the program using such interpretation.

We give a realizability interpretation of MCICD-formulas into the system MCICD*, which is MCICD over the term system MCICT and extended with first order existential formulas and restricted formulas.

3.1 The Logic MCICD*

MCICD* is an extension of MCICD defined as follows: we add first-order existential and restricted formulas (defined below), we extend the term system to MCICT. Finally, tags in clauses can be either function symbols (considered as constants added to MCICT) or closed terms of MCICT. Subject reduction and strong normalization hold also for this extended system.

3.1.1 Existential Formulas

Existential formulas are ruled by:

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A[x := s]}{\Gamma \vdash_{\mathbb{E}} \mathsf{pack}\, t : \exists x A}\ (\exists I) \qquad \frac{\Gamma \vdash_{\mathbb{E}} t : \exists x A \quad \Gamma, z : A[x := u] \vdash_{\mathbb{E}} r : B}{\Gamma \vdash_{\mathbb{E}} \mathsf{open}(t, z.r) : B}\ (\exists E)$$

where in the $(\exists E)$ rule, $u \notin FV(\Gamma, B, \exists x A)$. Proof reduction is given by the following β-reduction rule: $\mathsf{open}(\mathsf{pack}\, t, z.r) \ \mapsto_\beta \ r[z := t]$

The rules for existential formulas are given only in partial Curry-style, due to the fact that the rules in full Curry-style (non-traceable rules) would cause the subject reduction property to fail.

3.1.2 Restricted Formulas

We will also need Parigot's restriction to be able to formulate realizability for disjunctions. Expressions of the form $A\restriction s_1 = t_1, \ldots, s_k = t_k$ are called restricted formulas. A restricted formula represents a conjunction $A \wedge s_1 =$

$t_1 \wedge \ldots \wedge s_k = t_k$ where the equations do not have computational content, i.e. are not traced by the proof-term system.
Restriction behaves according to the following rules, where we abbreviate a sequence of equations as $\boldsymbol{s} = \boldsymbol{t}$. :

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A \quad \Gamma \vdash_{\mathbb{E}} \boldsymbol{s} = \boldsymbol{t}}{\Gamma \vdash_{\mathbb{E}} r : A\restriction \boldsymbol{s} = \boldsymbol{t}} \ (\restriction I)$$

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A\restriction \boldsymbol{s} = \boldsymbol{t}}{\Gamma \vdash_{\mathbb{E}} r : A} \ (\restriction E)$$

The notation $\Gamma \vdash_{\mathbb{E}} s = t$ means a derivation obtained within $\mathsf{MCICD}^\star$ and possibly using the following rule:

$$\frac{\Gamma \vdash_{\mathbb{E}} r : A\restriction \boldsymbol{s} = \boldsymbol{t}}{\Gamma \vdash_{\mathbb{E}} s_i = t_i} \ (\restriction E_R)$$

We have a basic context of equalities $\mathbb{E}_\beta := \{t = r \mid t \to_\beta r \text{ or } r \to_\beta t\}$, which represents β-equality but only for one-step reduction. Unless stated otherwise, while working in $\mathsf{MCICD}^\star$, we will write $\vdash$ for $\vdash_{\mathbb{E}_\beta}$.

3.2 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: $\mathbb{C}_i^k := \lambda x.\, \mathsf{in}_{k,i}\, x$ and $\mathbb{D}_i^k := \lambda x.\, \mathsf{out}_{k,i}\, x$

Definition 3.1 Given an MCICT-term t and an $\mathsf{MCICD}-$formula A we define the $\mathsf{MCICD}^\star-$formula $t\ \mathtt{r}\ A$ as follows:

$$\begin{aligned}
t\ \mathtt{r}\ X\boldsymbol{s} &:= X^+\boldsymbol{s}\,t \\
t\ \mathtt{r}\ A \to B &:= \forall z.z\ \mathtt{r}\ A \to tz\ \mathtt{r}\ B \\
t\ \mathtt{r}\ \forall x A &:= \forall x.t\ \mathtt{r}\ A \\
t\ \mathtt{r}\ \forall X A &:= \forall X^+.t\ \mathtt{r}\ A \\
t\ \mathtt{r}\ A \wedge B &:= (\pi_1 t\ \mathtt{r}\ A) \wedge (\pi_2 t\ \mathtt{r}\ B) \\
t\ \mathtt{r}\ A \vee B &:= \exists z.(z\ \mathtt{r}\ A\restriction t = \mathsf{inl}\, z) \vee (z\ \mathtt{r}\ B\restriction t = \mathsf{inr}\, z) \\
t\ \mathtt{r}\ \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)\boldsymbol{s} &:= \mu X^+(\mathcal{C}_1^{\mathtt{r}},\ldots,\mathcal{C}_k^{\mathtt{r}})\boldsymbol{s}\,t \\
t\ \mathtt{r}\ \ \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\boldsymbol{s} &:= \nu X^+(\mathcal{D}_1^{\mathtt{r}},\ldots,\mathcal{D}_k^{\mathtt{r}})\boldsymbol{s}\,t
\end{aligned}$$

where for an n-ary predicate $\mathcal{F}$ we define the $(n+1)$-ary predicate $\mathcal{F}^{\mathtt{r}} := \lambda \boldsymbol{y}, z.z\ \mathtt{r}\ \mathcal{F}\boldsymbol{y}$ with $z \notin \boldsymbol{y} \cup FV(F)$, and if $\mathcal{C}_i := \langle \mathcal{F}_i, \boldsymbol{c_i}\rangle,\ \mathcal{D}_i := \langle \mathcal{G}_i, \boldsymbol{d_i}\rangle$ then

$\mathcal{C}_i^{\mathtt{r}} := \langle \mathcal{F}_i^{\mathtt{r}}, \boldsymbol{c_i}, \mathbb{C}_i^k \rangle, \mathcal{D}_i^{\mathtt{r}} := \langle \mathcal{G}_i^{\mathtt{r}}, \boldsymbol{d_i}, \mathbb{D}_i^k \rangle$.

Observe that the last tag in the clauses $\mathcal{C}_i^{\mathtt{r}}, \mathcal{D}_i^{\mathtt{r}}$ is a closed term, existential and restricted formulas are only needed to define realizability for disjunctions and that the realizability for a (co)inductive predicate is (co)inductively defined.

The essential difference with the more usual modified realizability interpretation is the case for first order universal formulas, in our case the realizer does not behave as a function, so that universal formulas do not have computational content. In the formula $t\ \mathtt{r}\ \forall x A$ the term t can be thinked of as a "potential" realizer of the formula $\forall x A$. In the cases of interest, involving some formal datatype, the actual realizer t will behave as a function. This choice of definition simplifies proofs, in particular it allows for the canonical iterator (recursor) to be the realizer of the induction (extended induction) axiom (see prop. 3.6).

Lemma 3.2 (Substitution Properties) *The following properties hold:*

(*i*) $(t\ \mathtt{r}\ A)[x := s] \equiv t[x := s]\ \mathtt{r}\ A[x := s]$.

(*ii*) $(t\ \mathtt{r}\ A)[X^+ := \mathcal{F}^{\mathtt{r}}] \equiv t\ \mathtt{r}\ A[X := \mathcal{F}]$.

Proof. Induction on A. □

3.3 Realizing the Axioms

The rules for (co)inductive predicates generate the following axioms.

Definition 3.3 Given an inductive predicate $\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ we define the closure, induction and strong induction axioms for $\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ as follows [4]:

$$\mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),i} := \mathcal{F}_i[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k)] \subseteq (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))^{\boldsymbol{c_i}}$$

$$\mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)} := \forall Z.\, \boldsymbol{\mathcal{F}}\ \mathsf{mon}\ X, \mathcal{F}_1[X := Z] \subseteq Z^{\boldsymbol{c_1}}, \ldots, \mathcal{F}_k[X := Z] \subseteq Z^{\boldsymbol{c_k}} \to \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z$$

$$\mathsf{Ind}^+_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)} := \forall Z.\, \boldsymbol{\mathcal{F}}\ \mathsf{mon}\ X, \mathcal{F}_1[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge Z] \subseteq Z^{\boldsymbol{c_1}}, \ldots, \mathcal{F}_k[X := \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \wedge Z] \subseteq Z^{\boldsymbol{c_k}} \to \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z$$

Analogously, given a coinductive predicate $\nu X(\mathcal{D}_1, \ldots, \mathcal{D}_k)$ we define its

[4] $A_1, \ldots, A_k \to B$ means $A_1 \to \ldots \to A_k \to B$ and $\boldsymbol{\mathcal{F}}\ \mathsf{mon}\ X$ denotes $\mathcal{F}_1\ \mathsf{mon}\ X, \ldots, \mathcal{F}_k\ \mathsf{mon}\ X$

coclosure, coinduction and inversion axioms as follows:

$$\begin{aligned}
\mathsf{CoCl}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k),i} &:= \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \subseteq (\mathcal{F}_i[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)])^{\boldsymbol{c_i}}\\
\mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} &:= \forall Z.\, \boldsymbol{\mathcal{F}}\,\mathsf{mon}\, X, Z \subseteq \mathcal{F}_1[X := Z]^{\boldsymbol{c_1}},\ldots, Z \subseteq \mathcal{F}_k[X := Z]^{\boldsymbol{c_k}}\\
&\qquad \to Z \subseteq \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\\
\mathsf{CoInd}^{+}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} &:= \forall Z.\, \boldsymbol{\mathcal{F}}\,\mathsf{mon}\, X, Z \subseteq \mathcal{F}_1[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \vee Z]^{\boldsymbol{c_1}},\ldots,\\
&\qquad Z \subseteq \mathcal{F}_k[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k) \vee Z]^{\boldsymbol{c_k}} \to Z \subseteq \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\\
\mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)} &:= \forall \boldsymbol{z}.\, \boldsymbol{\mathcal{F}}\,\mathsf{mon}\, X, \mathcal{F}_1[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)]\boldsymbol{c_1 z},\ldots,\\
&\qquad \mathcal{F}_k[X := \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)]\boldsymbol{c_k z} \to \nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)\boldsymbol{z}
\end{aligned}$$

The usual (co)induction axioms for each predicate can be derived from these general ones, for example from $\mathsf{Ind}^{+}_{\mathbb{N}}$ as the monotonicity assumptions are trivially derivable in this case, we can derive the extended induction axiom: $\forall Z.Z0, (\forall x.\mathbb{N}x \wedge Zx \to Zsx) \to \mathbb{N} \subseteq Z$, where 0 is defined as the global zero $0_{\mathtt{g}}$, applied to the unique inhabitant of the unit predicate $\star$, i.e., $0 := 0_{\mathtt{g}}\star$. In the rest of this section we look for realizers for the (co)closure and (co)induction axioms which will be needed in the proof of the soundness theorem.

Proposition 3.4 *Given an inductive predicate $\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)$ we have:*

$$\vdash \lambda x.\, \mathsf{in}_{k,j}\, x : \mathbb{C}^k_j\ \mathtt{r}\ \mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),j}$$

Proof. Using lemma 3.2, part (ii), we get that $\mathbb{C}^k_j\ \mathtt{r}\ \mathsf{Cl}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k),j}$ is equivalent to $\mathsf{Cl}_{\mu X^{+}(\mathcal{C}^{\mathtt{r}}_1,\ldots,\mathcal{C}^{\mathtt{r}}_k),j}$. The proof is now obvious. □

Proposition 3.5 *Given a coinductive predicate $\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)$ we have:*

$$\vdash \lambda x.\, \mathsf{out}_{k,j}\, x : \mathbb{D}^k_j\ \mathtt{r}\ \mathsf{CoCl}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k),j}$$

Proof. Analogous to proposition 3.4. □

Proposition 3.6 *If $\Theta \vdash \mathsf{m_i} : \mathcal{F}^{\mathtt{r}}_i\,\mathsf{mon}\, X^{+}$ for $1 \le i \le k$ then*

(i). $\Theta \vdash \lambda\boldsymbol{x}.\lambda\boldsymbol{y}.\lambda z.\mathsf{It}_k(\mathbf{m}, \mathbf{s}, z) : \mathbb{J}\ \mathtt{r}\ \mathsf{Ind}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$

(ii). $\Theta \vdash \lambda\boldsymbol{x}.\lambda\boldsymbol{y}.\lambda z.\mathsf{Rec}_k(\mathbf{m}, \mathbf{s}, z) : \mathbb{R}\ \mathtt{r}\ \mathsf{Ind}^{+}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$

where $\mathbb{J} := \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{It}_k(\boldsymbol{x}, \boldsymbol{y}, z)$, $\mathbb{R} := \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{Rec}_k(\boldsymbol{x}, \boldsymbol{y}, z)$ and for $1 \le i \le k$, $\mathsf{s_i} := \lambda u_i.y_i(x_i(\lambda v.v)u_i)$

Proof. We prove part (ii). Set $\mathcal{C}_i = \langle \mathcal{F}_i, \boldsymbol{c_i}\rangle$, $\mu := \mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)$ and $\mu^{\mathtt{r}} := \mu X^{+}(\mathcal{C}^{\mathtt{r}}_1,\ldots,\mathcal{C}^{\mathtt{r}}_k)$. Our goal is to prove $\Theta \vdash \mathbb{R}\ \mathtt{r}\ \mathsf{Ind}^{+}_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$, which unfolds

to

$$\begin{aligned}\forall Z^{+}.\forall \boldsymbol{m}.\ & \ldots, m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X, \ldots_{(1\leq i\leq k)} \rightarrow \\ & \forall \boldsymbol{f}.\ \ldots, f_i\ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\boldsymbol{c_i}}, \ldots_{(1\leq i\leq k)} \rightarrow \\ & \mathbb{R}\boldsymbol{mf}\ \mathtt{r}\ \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z\end{aligned} \tag{1}$$

Assume for $1 \leq i \leq k$

$$x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X \tag{2}$$

and $y_i : f_i\ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\boldsymbol{c_i}}$, that is

$$y_i : \forall \boldsymbol{v}.\forall u.u\ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{v} \rightarrow f_i u\ \mathtt{r}\ Z(\boldsymbol{c_i v}). \tag{3}$$

We need to show $\mathbb{R}\boldsymbol{mf}\ \mathtt{r}\ \mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k) \subseteq Z$, i.e.

$$\forall \boldsymbol{v}.\forall w.w\ \mathtt{r}\ (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\boldsymbol{v} \rightarrow \mathbb{R}\boldsymbol{mf}w\ \mathtt{r}\ Z\boldsymbol{v}$$

Assume now $z : w\ \mathtt{r}\ (\mu X(\mathcal{C}_1, \ldots, \mathcal{C}_k))\boldsymbol{v} \equiv (\mu X^{+}(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\boldsymbol{v}w$, and define $\mathcal{Q} := \lambda \boldsymbol{x}, z.\mathbb{R}\boldsymbol{mf}z\ \mathtt{r}\ Z\boldsymbol{x}$. Set

$$\begin{aligned}\Gamma := \Theta, & x_i : m_i\ \mathtt{r}\ \mathcal{F}_i\,\mathsf{mon}\,X,\ y_i : f_i\ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z] \subseteq Z^{\boldsymbol{c_i}}, \\ & z : (\mu X^{+}(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\boldsymbol{v}w\end{aligned}$$

where $1 \leq i \leq k$. We need to prove $\Gamma \vdash \mathcal{Q}\boldsymbol{v}w$.

Obviously $\Gamma \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^{+}$ and $\Gamma \vdash z : (\mu X^{+}(\mathcal{C}_1^{\mathtt{r}}, \ldots, \mathcal{C}_k^{\mathtt{r}}))\boldsymbol{v}w$. Therefore using the elimination rule (μE^{+}) it suffices to show

$$\Gamma \vdash \mathcal{F}_i^{\mathtt{r}}[X^{+} := \mu^{\mathtt{r}} \wedge \mathcal{Q}] \subseteq \mathcal{Q}^{\boldsymbol{c_i}, \mathbb{C}_i^k},\ (1 \leq i \leq k)$$

that is

$$\forall \boldsymbol{x}.\forall z.\mathcal{F}_i^{\mathtt{r}}[X^{+} := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{x}z \rightarrow \mathcal{Q}(\boldsymbol{c_i x})(\mathbb{C}_i^k z).$$

Assume

$$u_i : \mathcal{F}_i^{\mathtt{r}}[X^{+} := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{x}z \quad (1 \leq i \leq k) \tag{4}$$

and set $\Pi := \Gamma, u_i : \mathcal{F}_i^{\mathtt{r}}[X^{+} := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{x}z$. We need to prove

$$\Pi \vdash \mathcal{Q}(\boldsymbol{c_i x})(\mathbb{C}_i^k z) \quad (1 \leq i \leq k) \tag{5}$$

The assumptions (2) unfold to:

$$\begin{aligned}x_i : \forall X^{+}\forall Y^{+}\forall z.\ & (\forall \boldsymbol{y}\,\forall w.w\ \mathtt{r}\ X\boldsymbol{y} \rightarrow zw\ \mathtt{r}\ Y\boldsymbol{y}) \rightarrow \\ & (\forall \boldsymbol{y}\,\forall u.u\ \mathtt{r}\ \mathcal{F}_i\boldsymbol{y} \rightarrow m_i z u\ \mathtt{r}\ \mathcal{F}_i[X := Y]\boldsymbol{y})\end{aligned} \tag{6}$$

Next we instantiate the predicate variables $X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}, Y^+ := (\mu \wedge Z)^{\mathtt{r}}$ to obtain:

$$x_i : \forall z.(\forall \boldsymbol{y} \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\boldsymbol{y}w \to (\mu \wedge Z)^{\mathtt{r}}\boldsymbol{y}(zw)) \to \\ (\forall \boldsymbol{y} \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{y}u \to m_i z u \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{y})$$

Instantiate now $z := \lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle$, getting

$$x_i : (\forall \boldsymbol{y} \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\boldsymbol{y}w \to (\mu \wedge Z)^{\mathtt{r}}\boldsymbol{y}((\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)w)) \to \\ (\forall \boldsymbol{y} \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{y}u \to m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)u \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{y})$$

Observing that $(\mathcal{F} \wedge \mathcal{G})^{\mathtt{r}} \equiv \lambda \boldsymbol{z}, u.\mathcal{F}^{\mathtt{r}}\boldsymbol{z}(\pi_1 u) \wedge \mathcal{G}^{\mathtt{r}}\boldsymbol{z}(\pi_2 u)$ and

$$\mathbb{E}_\beta \vdash \pi_1\big((\lambda x.\langle x, \pi_2\boldsymbol{mf}x\rangle)w\big) = w \\ \mathbb{E}_\beta \vdash \pi_2\big((\lambda x.\langle x, \pi_2\boldsymbol{mf}x\rangle)w\big) = \pi_2\boldsymbol{mf}w$$

using the rule for Leibniz' Equality (Eq) it is easy to see that

$$\vdash \lambda v.v : \forall \boldsymbol{y} \forall w.(\mu^{\mathtt{r}} \wedge \mathcal{Q})\boldsymbol{y}w \to (\mu \wedge Z)^{\mathtt{r}}\boldsymbol{y}((\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)w).$$

Therefore we can eliminate the implication and obtain

$$\Pi \vdash x_i(\lambda v.v) : \forall \boldsymbol{y} \forall u.\mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}]\boldsymbol{y}u \to \\ m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)u \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{y}.$$

From this, instantiating $\boldsymbol{y}, u := \boldsymbol{x}, z$ and using assumption (4) we get

$$\Pi \vdash x_i(\lambda v.v)u_i : m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{x}.$$

On the other hand, from assumptions (3), with $\boldsymbol{v}, u := \boldsymbol{x}, m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z$ we get:

$$\Pi \vdash y_i : m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z \ \mathtt{r}\ \mathcal{F}_i[X := \mu \wedge Z]\boldsymbol{x} \to f_i(m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z) \ \mathtt{r}\ Z(\boldsymbol{c_i x}).$$

Therefore $\Pi \vdash y_i(x_i(\lambda v.v)u_i) : f_i\big(m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z\big) \ \mathtt{r}\ Z(\boldsymbol{c_i x})$.
But it is easy to see that $\mathbb{E}_\beta \vdash f_i\big(m_i(\lambda x.\langle x, \mathbb{R}\boldsymbol{mf}x\rangle)z\big) = \mathbb{R}\boldsymbol{mf}(\mathbb{C}_i^k z)$, hence using (Eq) we get:

$$\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathbb{R}\boldsymbol{mf}(\mathbb{C}_i^k z) \ \mathtt{r}\ Z(\boldsymbol{c_i x}),$$

That is $\Pi \vdash y_i(x_i(\lambda v.v)u_i) : \mathcal{Q}(\boldsymbol{c_i x})(\mathbb{C}_i^k z)$ and the goal (5) is proved. Therefore $\Gamma \vdash \lambda u_i.y_i(x_i(\lambda v.v)u_i) : \mathcal{F}_i^{\mathtt{r}}[X^+ := \mu^{\mathtt{r}} \wedge \mathcal{Q}] \subseteq \mathcal{Q}^{\boldsymbol{c_i},\mathbb{C}_i^k}$, which by (μE^+) yields:

$$\Gamma \vdash \mathsf{Rec}_k(\mathbf{m}, \mathbf{s}, z) : \mathcal{Q}vw$$

Finally, discharging the assumptions $\boldsymbol{x}, \boldsymbol{y}, z$, we get formula (1), which coincides with:

$$\Theta \vdash_{\mathsf{MCICD}^\star} \lambda\boldsymbol{x}.\lambda\boldsymbol{y}.\lambda z.\mathsf{Rec}_k(\mathbf{m}, \mathbf{s}, z) : \mathbb{R}\ \mathtt{r}\ \mathsf{Ind}^+_{\mu X(\mathcal{C}_1,\ldots,\mathcal{C}_k)}$$

□

Analogously we can get the realizability of the coinduction axioms:

Proposition 3.7 *If* $\Theta \vdash \mathsf{m_i} : \mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^+$ *for* $1 \leq i \leq k$, *then*

(i). $\Theta \vdash \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{CoIt}_k(\mathbf{m}, \mathbf{s}, \mathsf{pack}\, z) : \mathbb{Kb}\ \mathtt{r}\ \mathsf{CoInd}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

(ii). $\Theta \vdash \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{CoRec}_k(\mathbf{m}, \mathbf{q}, \mathsf{pack}\, z) : \mathbb{Q}\ \mathtt{r}\ \mathsf{CoInd}^+_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

(iii). $\Theta \vdash \lambda\boldsymbol{x}\lambda\boldsymbol{y}.\mathsf{out}_k^{-1}(\mathbf{m}, \mathbf{r}) : \mathbb{I}\ \mathtt{r}\ \mathsf{Inv}_{\nu X(\mathcal{D}_1,\ldots,\mathcal{D}_k)}$

with $\mathbb{Kb} := \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{CoIt}_k(\boldsymbol{x}, \boldsymbol{y}, z)$, $\mathbb{Q} := \lambda\boldsymbol{x}\lambda\boldsymbol{y}\lambda z.\mathsf{CoRec}_k(\boldsymbol{x}, \boldsymbol{y}, z)$, $\mathbb{I} := \lambda\boldsymbol{x}\lambda\boldsymbol{y}.\mathsf{out}_k^{-1}(\boldsymbol{x}, \boldsymbol{y})$ *and for* $1 \leq i \leq k$, *we set:*

$$\mathsf{s_i} := \lambda v.\,\mathsf{open}(v, w.x_i(\lambda u.\,\mathsf{pack}\,u)(y_i w)),$$

$$\mathsf{q_i} := \lambda v.\,\mathsf{open}\left(v, w.x_i\Big(\lambda u.\,\mathsf{open}\left(u, v.\mathsf{case}(v, v_1.\mathsf{inl}\,v_1, v_2.\mathsf{inr}\,\mathsf{pack}\,v_2)\right)\Big)(y_i w)\right)$$

$$\mathsf{r_i} := x_i(\lambda z z)y_i.$$

Observe that the proof-terms are quite complicated due to the absence of existential rules in full Curry-style.

The additional requirements $\mathcal{F}_i^{\mathtt{r}}\,\mathsf{mon}\,X^+$ in propositions 3.6 and 3.7 are somehow unpleasant, we would like to obtain them from the fact that $\mathcal{F}_i\,\mathsf{mon}\,X$ is realizable, fact which will be available from some induction hypothesis. Unfortunately this is not true in general but we have the following result,

Proposition 3.8 *If* $\Theta \vdash_{\mathsf{MCICD}^\star} \widehat{m} : m\ \mathtt{r}\ \mathcal{F}\,\mathsf{mon}\,X$ *and* $\mathbb{E} \vdash m(\lambda xx) = \lambda xx$ *(i.e. the first functor law holds for* m*) then* $\Theta \vdash_{\mathsf{MCICD}^\star,\mathbb{E}} \widehat{m} : \mathcal{F}^{\mathtt{r}}\,\mathsf{mon}\,X^+$.

Proof. Instantiate $z := \lambda x.x$ in $m\ \mathtt{r}\ \mathcal{F}\,\mathsf{mon}\,X$ (cf. formula (6), page 10) and use some equational reasoning. □

This proposition allows to obtain a soundness theorem where both source and target logical systems only differ on the underlying object-term system and on the equational theory given by the equations on the above proposition,

which express the first functor law for m. This is an important difference with the treatment in [11].

3.4 The Soundness Theorem

We come to the main result of this paper, a soundness theorem for our realizability interpretation, which guarantees the correctness of program extraction. Important feature of this result is the fact that the extracted program coincides with the code of the original proof of the specification A.

Definition 3.9 Given a proof-term t, and a subterm m such that m occurs in $\boldsymbol{m}$ for some subterm of t of one of these forms: $\mathsf{It}_k(\boldsymbol{m}, \boldsymbol{s}, r), \mathsf{Rec}_k(\boldsymbol{m}, \boldsymbol{s}, r)$, $\mathsf{CoIt}_k(\boldsymbol{m}, \boldsymbol{s}, r), \mathsf{CoRec}_k(\boldsymbol{m}, \boldsymbol{s}, r), \mathsf{out}_k^{-1}(\boldsymbol{m}, \boldsymbol{s})$, 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 $\mathcal{W}(t)$.

Definition 3.10 Given a derivation $\Gamma \vdash_{\mathbb{E}} s : A$ we define $\mathbb{FFL}(s) := \{m(\lambda zz) = \lambda zz \mid m \in \mathcal{W}(s)\}$ and $\mathbb{E}^\star(s) := \mathbb{E} \cup \mathbb{FFL}(s)$ The equations in $\mathbb{FFL}(s)$ represent the first functor law for every on-display monotonicity witness m occurring in s.

The equations in $\mathbb{E}^\star(s)$ allow to derive all needed formulas of the form $\mathcal{F}^{\mathtt{r}}\,\mathsf{mon}\,X^+$ required by propositions 3.6 and 3.7.

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. $x_i \notin FV(A_i)$. It is important to remark that we make no syntactic distinction between object and proof-term variables.

Theorem 3.11 (Soundness of Realizability for MCICD) *If* $\Gamma \vdash_{\mathsf{MCICD},\mathbb{E}} s : A$ *then* $\Gamma^{\mathtt{r}} \vdash_{\mathsf{MCICD}^\star,\mathbb{E}^\star(s)} \widetilde{s} : s\ \mathtt{r}\ A$

Proof. Induction on $\vdash_{\mathsf{MCICD},\mathbb{E}}$. Lemma 3.2 as well as subject reduction of the system will be needed. The cases $(\mu I), (\mu E), (\mu E^+), (\nu E), (\nu I), (\nu I^+), (\nu I^i)$ are solved using the propositions 3.4,3.6,3.5,3.7, respectively. □

The proof-term $\widetilde{s}$ is in all cases but the disjunctions and (co)inductive predicates, homomorphic to s and can be automatically obtained from it.

4 Related Work

Logical systems handling only inductive definitions are presented in [9,11,7]. The system of [10], based on that of [9], is an extension of AF2 with positive (co)inductive definitions but does not include primitive recursion and uses a fixed point operator within the proof-term system, which is therefore not

strongly normalizing. An extension of Beeson's EON with monotone inductive definitions and only iteration is presented in [11], 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 [7] I present an extension of AF2 with monotone inductive definitions without clauses and only iteration and give a realizability interpretation into AF2, i.e. where the case for inductive predicates is reductive. [12] presents several strongly normalizing extensions of first order intuitionistic logic with positive (co)inductive definitions.
The type system MCICT of realizers is essentially a polymorphic and monotone version of the system developed in [2], which also uses clauses, but we include primitive (co)recursion and use a natural deduction approach, following [5] very closely. The use of clauses in the logic is already present in [1], this paper also inspires our definition of realizability for the case of inductive predicates, but we do not use modified realizability.

5 Conclusions and Further Work

The logical system MCICD together with its realizability interpretation are a logical framework suitable for extract typable programs from specifications including monotone (co)inductive definitions. These definitions are given by clauses, a concept which simplifies the defining formulas. The extracted programs are guaranteed to terminate because they belong to a strongly normalizing term rewrite system. To my knowledge this is the first realizability interpretation for full monotone (co)inductive definitions (including (co)recursion) where the system of realizers is strongly normalizing.
Although all usual examples of (co)inductive predicates are positive, the use of full monotonicity simplifies the proofs and allows for a direct extension to higher-order logic (corresponding to the type system F^{ω}) where a concept of positivity is not well determined. On the other hand we have defined a tarskian semantics for MCICD and extended the *programming with proofs* paradigm of [3,9], which allows to program functions defined by equations on formal datatypes (like the examples in page 4) without having to calculate realizers (see [8]).

Acknowledgement

I wish to thank Ralph Matthes for several fruitful discussions. The *Graduiertenkolleg Logik in der Informatik* of the *Deutsche Forschungsgemeinschaft* provided the travelling funds to Fontainebleau.

References

[1] Berger Ulrich. *A constructive interpretation of positive inductive definitions*. Unpublished Draft. 1995.

[2] Hagino T. *A Typed Lambda Calculus with Categorical Type Constructors*, "Category Theory and Computer Science" edited by Pitt D.H, A. Poigné, D.E. Rydeheard. LNCS **283**. Springer 1987.

[3] Krivine J.L, M. Parigot. *Programming with Proofs*. Journal of Information Processing and Cybernetics EIK **26(3)** (1990) pp. 149-167.

[4] Krivine J.L. "Lambda-Calculus, Types and Models". Ellis Horwood Series in Computers and their Applications. Ellis Horwood, Masson 1993.

[5] Matthes Ralph. "Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types". Dissertation Universität München, 1999.

[6] Matthes Ralph. *Monotone (co)inductive types and positive fixed-point types*. Theoretical Informatics and Applications **33(4-5)** (1999) pp. 309-328. EDP Sciences.

[7] Miranda-Perea Favio E. *A Curry-Style Realizability Interpretation for Monotone Inductive Definitions*. Proc. 7th. ESSLLI Student Session (2002) edited by Nissim M. Available from the web site: http://www.iccs.informatics.ed.ac.uk/~malvi/esslli02/index.html

[8] Miranda-Perea Favio E. *A Logic for Monotone and Clausular (Co)inductive Definitions*. Unpublished Draft. 2004 (Available upon request).

[9] Parigot M. *Recursive programming with proofs*. Theoretical Computer Science **94** (1992) pp.335-356.

[10] Raffalli C. "L' Arithmétique Fonctionnelle du Second Ordre avec Points Fixes". Thèse de l'Université Paris VII. 1994.

[11] Tatsuta M. *Two Realizability Interpretations of Monotone Inductive Definitions*. International Journal of Foundations of Computer Science **5(1)** (1994) pp. 1-21.

[12] Uustalu T. "Natural deduction for intuitionistic least and greatest fixedpoint logics, with an application to program construction". Dissertation TRITA-IT AVH 98:03, Dept. of Teleinformatics, Royal Inst of Technology (KTH), Stockholm, 1998.