A Curry-Style Realizability Interpretation for Monotone Inductive Definitions

Favio E. Miranda-Perea
Institut für Informatik der Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München, Germany
miranda@informatik.uni-muenchen.de[1]

Abstract. The logical system AF2μ, an extension of second-order predicate calculus with monotone inductive definitions, is presented. Some properties of this system are shown including that it is a good system for extracting programs from proofs by means of a realizability interpretation in the style of Krivine-Parigot, where the realizers are terms of the Curry-system of $\lambda^{\rightarrow}$-calculus and the realizability formulas belong to second-order logic.

1 Introduction

Realizability interpretations have been used extensively in proof theory and recently also as a tool in computer science to extract programs from proofs. However, the class of programs obtained with this so-called *proofs as programs* paradigm is still quite restricted. On the other hand the use of inductive definitions, polymorphic and inductively defined datatypes in modern programming languages is indispensable. This motivates us to investigate logical systems of inductive definitions and realizability interpretations, which allow to extract programs from proofs in these systems.

2 A Logical System for Monotone Inductive Definitions

In this section we present a system of monotone inductive definitions based in second-order minimal logic together with some good properties that the system possess.

[1]**This Research is being supported by grant 154186 of the** CONACyT-DAAD **treaty.**

Proceedings of the Seventh ESSLLI Student Session
Malvina Nissim (editor)

2.1 A Language for Inductive Definitions

The language is second-order and it is based on a term language which is a subset of the untyped lambda calculus with function symbols.

The formulas are defined by the following grammar:

$$A, B, C ::= X^{(n)} t_1 \dots t_n \mid P^{(n)} t_1 \dots t_n \mid A \to B \mid \forall x.A \mid \forall X.B,$$

where $X^{(n)}$ is a n-ary predicate variable and $P^{(n)}$ is either a predicate symbol or a *defined predicate* of arity n. A *defined predicate* is either a comprehension predicate or an inductive predicate. More precisely we have a simultaneous definition of defined predicates and formulas.

Definition 1 *Let F be a formula. The expression $\lambda\vec{y}.F$ is called a comprehension predicate.*

Intuitively, the expression $\lambda\vec{y}.F$ represents the set $\{\vec{t} \mid F[\vec{y} := \vec{t}]\}$, so the expression $(\lambda\vec{y}.F)\vec{t}$ must be understood as $F[\vec{y} := \vec{t}]$. From now on, we denote a comprehension predicate defined by a formula F with the respective calligraphic letter $\mathcal{F}$.

Definition 2 *Let $\mathcal{F}$ be a comprehension predicate. The expression $\mu X.\mathcal{F}$ is called an inductive predicate.*

Intuitively $\mu X.\mathcal{F}$ represents the least fixed point of the operator defined by $\mathcal{F}$, and in a model $\mathcal{M}$ it would be interpreted as the least predicate $\mathcal{K}$ such that $\mathcal{M}[X/\mathcal{K}] \models \forall\vec{y}.X\vec{y} \leftrightarrow F$ holds. The existence of such predicate $\mathcal{K}$ will be guaranteed by the monotonicity of $\mathcal{F}$, ensured by the rule (μI). If $\mathcal{F}$ is not monotone $\mu X.\mathcal{F}$ would be interpreted by the empty set.

Let $0, s$ be function symbols and Set $\mathcal{F} := \lambda y.\forall Y.Y0, (\forall z.Xz \to Ys(z)) \to Yy$,[2] the defining formula $F[X, y]$ may be seen as a function associating to every predicate X the least predicate containing 0 and all successors of an element of X. So $\mathbb{N} := \mu X.\mathcal{F}$ is the least predicate which contains 0 and is closed under the succesor function s, which is the intended set of (recursive) natural numbers. Similarly we can define lists of natural numbers as

$$\mathsf{List}(\mathbb{N}) := \mu X.\lambda z.\forall Y.\, Y\mathsf{nil}, (\forall n \forall y.\, \mathbb{N}n, Xy \to Y\mathsf{cons}(n, y)) \to Yz.$$

As syntactical sugar we have Leibniz equality and inclusion between predicates:

$$x = y := \forall X.Xx \to Xy \qquad P \subseteq Q := \forall\vec{z}.P\vec{z} \to Q\vec{z}.$$

2.2 System Rules

The logical rules are the usual natural deduction rules for $\to, \forall, \forall^2$, the rule for Leibniz Equality, and rules for Inductive Predicates, annotated by

[2] $A_1, \dots, A_k \to B$ means $A_1 \to (A_2 \to \dots \to B)$

proof-terms, it is important to remark that we make no syntactic distinction between object variables and proof-term variables.
Let $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$, the relation $\Gamma \vdash t : A$ is defined inductively, from

$$\{x_1 : A_1, \ldots, x_n : A_n\} \vdash x_i : A_i \ \ (Var),$$

as follows:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to I) \quad \frac{\Gamma \vdash r : A \to B \quad \Gamma \vdash s : A}{\Gamma \vdash rs : B}\ (\to E)$$

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

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

$$\frac{\Gamma \vdash u = v \quad \Gamma \vdash r : A[x := u]}{\Gamma \vdash r : A[x := v]}\ (Eq)^3$$

$$\frac{\Gamma \vdash t : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s} \quad \Gamma \vdash m : \mathcal{F}\mathsf{mon}X}{\Gamma \vdash Cmt : (\mu X.\mathcal{F})\vec{s}}\ (\mu I)$$

where $\mathcal{F}\mathsf{mon}X := \forall X \forall Y. X \subseteq Y \to \mathcal{F} \subseteq \mathcal{F}[X := Y]$.

$$\frac{\Gamma \vdash r : (\mu X.\mathcal{F})\vec{s} \quad \Gamma \vdash s : \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K}}{\Gamma \vdash rE_\mu s : \mathcal{K}\vec{s}}\ (\mu E)$$

where The rules $(\forall I)$ and $(\forall^2 I)$ have the usual variable conditions.
The rule (Eq) for Leibniz Equality is trivial if we do not have some equalities a priori, because we can prove $t = r$ if and only if $t \equiv r$, so the relation $\vdash$ really depends in some set of equalities $\mathbb{E}$ that we only make explicit if necessary. The starting system of equalities is

$$\mathbb{E} := \{t = r \mid t \to_\beta r \text{ or } r \to_\beta t\},$$

So we have β-equality but only for one-step reduction.
The annotated proof-terms for the rules involving a universal quantifier are equal for the premiss and for the conclusion, thus yielding a Curry-style system of proof-terms generated by the following grammar:

$$r, s, t, m ::= x \mid \lambda x.r \mid rs \mid Cmt \mid rE_\mu s.$$

The reduction relation $\to_\beta$ between proofs is defined as the term-closure of the following β-reduction relation $\mapsto_\beta$ for proof-terms:

$$\begin{array}{rcl} (\lambda x.t)r & \mapsto_\beta & t[x := r] \\ (Cmr)E_\mu s & \mapsto_\beta & s(m(\lambda x.xE_\mu s)r) \end{array}$$

[3]The proof-term for $u = v$ is irrelevant.

The second reduction rule corresponds to iteration on inductive types (see [Mat99]), the underlying untyped calculus is a Curry-style version of the iterative fragment of the system EMIT of monotone inductive types, studied in [Mat98].

As we can see, the introduction rule (μI) allows the introduction of an inductive definition $(\mu X.\mathcal{F})\vec{s}$ only when it has been proved that the comprehension predicate $\mathcal{F}$ is monotone in X. The proof of $\mathcal{F}\mathsf{mon}X$ is called a *monotonicity proof* and the proof-term m is called a *monotonicity witness*. The term m need not be closed, in that case we speak of conditional monotonicity. Moreover m can even be a variable, in which case we have hypothetical monotonicity.
These kind of definitions are therefore called *monotone inductive definitions*. Hence our system is an extension of the system AF2,[4] (see [Kri93]) with monotone inductive definitions and we call it AF2μ. It is well-known that if X has only positive occurrences in $\mathcal{F}$,[5] then $\mathcal{F}\mathsf{mon}X$ holds. Hence our system includes all positive inductive definitions and is therefore more general than Parigot's TTR (see [Par92]).

For our definition of natural numbers we can derive the expected properties, zero is a natural number: $\vdash Cm(\lambda u\lambda f.u) : \mathbb{N}0$ and the successor of a natural number is again a natural number: $\vdash \lambda n.Cm(\lambda u\lambda f.fn) : \forall x.\mathbb{N}x \to \mathbb{N}sx$, where m is the canonical monotonicity witness.

2.3 Embedding AF2μ into AF2

In this section we provide an embedding of AF2μ into AF2 which allows, knowing that AF2 strong normalizes, to show the strong normalization of AF2μ.

Definition 3 *The embedding* $(\cdot)' : \mathsf{AF2}\mu \to \mathsf{AF2}$ *is defined as follows:*

$$\begin{array}{rclcrcl}
x' & := & x & & (rs)' & := & r's' \\
(\lambda x.r)' & := & \lambda x.r' & & (Cmr)' & := & \lambda x.x(m'(\lambda z.zx)r') \\
(rE_\mu s)' & := & r's' & & & & \\
(X\vec{t})' & := & X\vec{t} & & (P\vec{t})' & := & P\vec{t} \\
(A \to B)' & := & A' \to B' & & (\forall\gamma.A)' & := & \forall\gamma.A' \\
\mathcal{F}' & := & \lambda\vec{y}.F' & & (\mu X.\mathcal{F})' & := & \lambda\vec{z}.\forall X.\mathcal{F}' \subseteq X \to X\vec{z}
\end{array}$$

where P is a predicate symbol and γ is a first or second order variable.
The embedding is compatible with substitution, that is, for object or proof terms $r, \vec{s}:\ r[\vec{x} := \vec{s}\,]' = r'[\vec{x} := \vec{s}\,']$ and for object-terms $\vec{r}:\ A[\vec{x} := \vec{r}\,]' = A'[\vec{x} := \vec{r}\,]$.

The following propositions state that the embedding has a good reduction behaviour and preserves derivations.

[4] AF2 is just second-order logic with the above given rules, but those for inductive predicates.

[5] i.e. X does not occur in F to the left of an odd number of implications.

Proposition 1 *The embedding $'$ translates a reduction step $r \to_\beta s$ in at least one reduction step $r' \to_\beta^+ s'$.*

Proof. Induction on $\to_\beta$. See [Mir02]. ℚed

Proposition 2 *If $\Gamma \vdash_{\mathsf{AF2}\mu} t : A$ then $\Gamma' \vdash_{\mathsf{AF2}} t' : A'$.*

Proof. Induction on $\vdash_{\mathsf{AF2}\mu}$. See [Mir02]. ℚed

Now, if we consider $\mathsf{AF2}\mu$ as a system of proof transformation rules, we obtain the following

Proposition 3 *$\mathsf{AF2}\mu$ is strongly normalizing.*

Proof. From propositions 1 and 2 knowing that $\mathsf{AF2}$ is strongly normalizing (see [Kri93]). ℚed

2.4 Subject Reduction

In this section we prove subject reduction for $\mathsf{AF2}\mu$. Recall that this property is not trivial for Curry-systems because of the rules for $\forall, \forall^2$ and the rule (Eq), which are not reflected in the proof-term system. Instead of Barendregt's method, we prefer to follow the more easily extensible Krivine method ([Kri93]).

Definition 4 *Given a formula A and a context Γ, we define the set $\mathcal{C}_{\Gamma,A}$ of Γ-instances of A as the least class of formulas which contains A and such that:*

- *If $B \in \mathcal{C}_{\Gamma,A}$ and $x \notin FV(\Gamma)$[6] then $B[x := t] \in \mathcal{C}_{\Gamma,A}$.*
- *If $B \in \mathcal{C}_{\Gamma,A}$ and $X \notin FV(\Gamma)$ then $B[X := \mathcal{F}] \in \mathcal{C}_{\Gamma,A}$.*
- *If $B[x := u] \in \mathcal{C}_{\Gamma,A}$ and $u = v$ then $B[x := v] \in \mathcal{C}_{\Gamma,A}$.*

Definition 5 *A formula A is an open formula if it does not start with $\forall$. In the formula $G := \forall\vec{\gamma}.F$, where F is an open formula, F is called the interior of G and is denoted G°.*

Lemma 1 *Let $\widetilde{A}$ be an open formula. If $\Gamma \vdash t : \widetilde{A}$ can be derived from $\Gamma \vdash t : A$ using only the rules $(\forall I), (\forall E), (\forall^2 I), (\forall^2 E), (Eq)$ then $\widetilde{A} \in \mathcal{C}_{\Gamma,A^\circ}$.*

Proof. Induction on the number of steps in the derivation of $\Gamma \vdash t : \widetilde{A}$ from $\Gamma \vdash t : A$. Case analysis on the first rule used in that derivation. $(\forall^2 E)$. We have $A \equiv \forall X.\forall\vec{\gamma}.A^\circ$ and after $(\forall^2 E)$, $\Gamma \vdash t : (\forall\vec{\gamma}.A^\circ)[X := \mathcal{F}]$. By IH we

[6] If $\Gamma = \{x_1 : A_1, \ldots, x_k : A_k\}$ then $FV(\Gamma) = FV(A_1) \cup \ldots \cup FV(A_k)$

have $\widetilde{A} \in \mathcal{C}_{\Gamma,(\forall\vec{\gamma}.A^\circ[X:=\mathcal{F}])^\circ}$, i.e., $\widetilde{A} \in \mathcal{C}_{\Gamma,(A^\circ[X:=\mathcal{F}])^\circ}$. We have two subcases: A° is atomic not beginning with X, or an implication. This implies that $A^\circ[X := \mathcal{F}]$ is of the same form, i.e., $A^\circ[X := \mathcal{F}]$ is open. Therefore we have $\widetilde{A} \in \mathcal{C}_{\Gamma,A^\circ[X:=\mathcal{F}]}$, and because w.l.o.g. $X \notin FV(\Gamma)$, we have $\widetilde{A} \in \mathcal{C}_{\Gamma,A^\circ}$. $A^\circ \equiv X\vec{r}$. We have $A^\circ[X := \mathcal{F}] = \mathcal{F}\vec{r}$, therefore $\widetilde{A} \in \mathcal{C}_{\Gamma,(\mathcal{F}\vec{r})^\circ}$, that is $\widetilde{A} \in \mathcal{C}_{\Gamma,F^\circ[\vec{y}:=\vec{r}]}$. Finally we have $F^\circ[\vec{y} := \vec{r}] = (A^\circ[X := \mathcal{F}])^\circ = A^\circ[X := \mathcal{F}]$. Hence $\widetilde{A} \in \mathcal{C}_{\Gamma,A^\circ[X:=\mathcal{F}]}$ which leads to $\widetilde{A} \in \mathcal{C}_{\Gamma,A^\circ}$.
The remaining cases are easier. ℚed

Lemma 2 (Generation Lemma) *Let $\Gamma \vdash t : A$, where A is an open formula. Then*

- *If $t = x$ then there exists $(x : B) \in \Gamma$ such that $A \in \mathcal{C}_{\Gamma,B^\circ}$.*
- *If $t = \lambda x.r$ then $A \equiv B \to C$ and $\Gamma, x : B \vdash r : C$ for some B, C.*
- *If $t = rs$ then there exist B, C such that $\Gamma \vdash r : C \to B, \Gamma \vdash s : C$ and $A \in \mathcal{C}_{\Gamma,B^\circ}$.*
- *If $t = Cmr$ then there is a formula F and terms $\vec{s}$ such that $A \equiv (\mu X.\mathcal{F})\vec{s}$, $\Gamma \vdash m : \mathcal{F}\mathsf{mon}X$ and $\Gamma \vdash r : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}$.*
- *If $t = rE_\mu s$ then there are formulas F, K and terms $\vec{s}$ such that $\Gamma \vdash r : (\mu X.\mathcal{F})\vec{s}$, $\Gamma \vdash s : \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K}$ and $A \in \mathcal{C}_{\Gamma,(\mathcal{K}\vec{s})^\circ}$.*

Proof. Analogous to the proof en [Kri93]. See [Mir02]. ℚed

Lemma 3 *$\Gamma \vdash t : A$ if and only if $\Gamma \vdash t : A^\circ$.*

Proof. Straightforward. ℚed

Lemma 4 *If $\Gamma \vdash t : A$ and $\widetilde{A} \in \mathcal{C}_{\Gamma,A}$ then $\Gamma \vdash t : \widetilde{A}$.*

Proof. Let $\mathcal{C} = \{B | \Gamma \vdash t : B\}$. We claim that $\mathcal{C}_{\Gamma,A} \subseteq \mathcal{C}$. By Hypothesis $A \in \mathcal{C}$. Now if $B \in \mathcal{C}$ and $x \notin FV(\Gamma)$ then $\Gamma \vdash t : \forall x.B$, by $(\forall I)$, and applying $(\forall E)$ we obtain $\Gamma \vdash t : B[x := r]$. Therefore $B[x := r] \in \mathcal{C}$, analogously we conclude that $B[X := \mathcal{F}] \in \mathcal{C}$. Finally if $B[x := u] \in \mathcal{C}$ and $u = v$ using (Eq) we obtain $B[x := v] \in \mathcal{C}$. The lemma follows immediately. ℚed

Lemma 5 (Subject Reduction for Open Formulas) *Let A be an open formula. If $\Gamma \vdash t : A$ and $t \to_\beta \hat{t}$ then $\Gamma \vdash \hat{t} : A$.*

Proof. Induction on t. If $t \equiv x$, there is no redex, and the claim is trivial. $t \equiv \lambda x.r$, which implies $\hat{t} \equiv \lambda x.\hat{r}$ with $r \to_\beta \hat{r}$. By lemma 2 we have $A \equiv C \to B$ and $\Gamma, x : C \vdash r : B \underset{\text{lemma 3}}{\Rightarrow} \Gamma, x : C \vdash r : B^\circ \underset{IH}{\Rightarrow} \Gamma, x : C \vdash \hat{r} :$

$B^\circ \underset{\text{lemma 3}}{\Rightarrow} \Gamma, x : C \vdash \hat{r} : B \Rightarrow \Gamma \vdash \lambda x.\hat{r} : C \to B$.
$t \equiv rs$. We have two subcases: $\hat{t} \equiv \hat{r}\hat{s}$ with $r \to_\beta \hat{r}, s \to_\beta \hat{s}$. By lemma 2 we have $\Gamma \vdash r : C \to B, \Gamma \vdash s : C$ and $A \in \mathcal{C}_{\Gamma,B^\circ}$. Lemma 3 implies $\Gamma \vdash s : C^\circ$. By IH we have $\Gamma \vdash \hat{r} : C \to B, \Gamma \vdash \hat{s} : C^\circ$, applying again lemma 3 yields $\Gamma \vdash \hat{s} : C$. Therefore $\Gamma \vdash \hat{r}\hat{s} : B \Rightarrow \Gamma \vdash \hat{r}\hat{s} : B^\circ$. Finally by lemma 4 we have $\Gamma \vdash \hat{r}\hat{s} : A$.
$t \equiv (\lambda x.r)s$ and $\hat{t} \equiv r[x := s]$. The Generation lemma implies $\Gamma \vdash \lambda x.r : C \to B, \Gamma \vdash s : C$ and $A \in \mathcal{C}_{\Gamma,B^\circ}$. Generation implies also $\Gamma, x : C \vdash r : B$ which yields[7] $\Gamma \vdash r[x := s] : B \underset{\text{lemma 3}}{\Rightarrow} \Gamma \vdash r[x := s] : B^\circ \underset{\text{lemma 4}}{\Rightarrow} \Gamma \vdash r[x := s] : A$.
$t \equiv Cmr$, which implies $\hat{t} \equiv C\hat{m}\hat{r}$ with $m \to_\beta \hat{m}, r \to_\beta \hat{r}$. By Generation lemma we have $A \equiv (\mu X.\mathcal{F})\vec{s}$, $\Gamma \vdash m : \mathcal{F}\mathsf{mon}X$ and $\Gamma \vdash r : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}$. Lemma 3 implies $\Gamma \vdash m : (\mathcal{F}\mathsf{mon}X)^\circ$ and $\Gamma \vdash r : (\mathcal{F}[X := \mu X.\mathcal{F}]\vec{s})^\circ \underset{IH}{\Rightarrow} \Gamma \vdash \hat{m} : (\mathcal{F}\mathsf{mon}X)^\circ$ and $\Gamma \vdash \hat{r} : (\mathcal{F}[X := \mu X.\mathcal{F}]\vec{s})^\circ \underset{\text{lemma 3}}{\Rightarrow} \Gamma \vdash \hat{m} : \mathcal{F}\mathsf{mon}X$ and $\Gamma \vdash \hat{r} : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}$. Finally (μI) implies $\Gamma \vdash C\hat{m}\hat{r} : (\mu X.\mathcal{F})\vec{s}$.
$t \equiv rE_\mu s$. We have two subcases: $t \equiv \hat{r}E_\mu\hat{s}$ with $r \to_\beta \hat{r}, s \to_\beta \hat{s}$. This case is analogous to the first subcase for application.
$t \equiv (Cmr)E_\mu s$ and $\hat{t} \equiv s(m(\lambda x.xE_\mu s)r)$. The generation lemma yields $\Gamma \vdash m : \mathcal{F}\mathsf{mon}X, \Gamma \vdash r : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}, \Gamma \vdash s : \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K}$ and $A \in \mathcal{C}_{\Gamma,(\mathcal{K}\vec{s})^\circ}$. It is easy to see that $\Gamma \vdash \lambda x.xE_\mu s : \mu X.\mathcal{F} \subseteq \mathcal{K}$. Now by $(\forall^2 E), (\to E)$ we have $\Gamma \vdash m(\lambda x.xE_\mu s) : \mathcal{F}[X := \mu X.\mathcal{F}] \subseteq \mathcal{F}[X := \mathcal{K}] \Rightarrow \Gamma \vdash m(\lambda x.xE_\mu s)r : \mathcal{F}[X := \mathcal{K}]\vec{s} \Rightarrow \Gamma \vdash s(m(\lambda x.xE_\mu s)r) : \mathcal{K}\vec{s} \underset{\text{lemma 3}}{\Rightarrow} \Gamma \vdash s(m(\lambda x.xE_\mu s)r) : (\mathcal{K}\vec{s})^\circ$. Finally, by lemma 4 we conclude $\Gamma \vdash s(m(\lambda x.xE_\mu s)r) : A$. ℚed

Proposition 4 (Subject Reduction) AF2μ *has subject reduction.*

Proof. Immediately from lemmas 3,5. ℚed

3 Realizability

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 simply lambda terms in Curry-style. The concept *the program t realizes the specification A* is formalized by means of a new formula t `r` A, which generally belongs to an extended language.

Let $\mathcal{L}^+$ be the extension of $\mathcal{L}_{\mathsf{AF2}}$ given by extending the object-term system with pure λ-terms and adding a new predicate variable (symbol) X^+ (P^+) for every predicate variable (symbol) X (P) where the arity of X^+ (P^+) equals the arity of X (P) plus 1.

[7] Using that if $\Gamma, x_1 : A_1, \ldots, x_k : A_k \vdash r : B$ and $\Gamma \vdash t_i : A_i$ then $\Gamma \vdash r[\vec{x} := \vec{t}] : B$

Definition 6 *Given a $\lambda^{\to}$-term t and a $\mathsf{AF2}\mu$-formula A, we define the $\mathsf{AF2}$-formula $t\ \mathtt{r}\ A$ as follows:*

$$t\ \mathtt{r}\ X\vec{s} := X^+\vec{s}t \qquad t\ \mathtt{r}\ P\vec{s} := P^+\vec{s}t$$
$$t\ \mathtt{r}\ A \to B := \forall z.z\ \mathtt{r}\ A \to tz\ \mathtt{r}\ B$$
$$t\ \mathtt{r}\ \forall x.A := \forall x.t\ \mathtt{r}\ A \qquad t\ \mathtt{r}\ \forall X.A := \forall X^+.t\ \mathtt{r}\ A$$
$$t\ \mathtt{r}\ (\mu X.\mathcal{F})\vec{s} := t\ \mathtt{r}\ \forall X.\mathcal{F} \subseteq X \to X\vec{s}$$

The following notation will be useful: $\mathcal{F}^{\mathtt{r}} := \lambda\vec{y}, z.z\ \mathtt{r}\ F$, where $z \notin FV(F)$. Given a context $\Gamma = \{x_1 : A_1, \ldots, x_k : A_k\}$ and arbitrary but fixed realizers t_i of A_i we set $\Gamma^{\mathtt{r}} := \{x_1 : t_1\ \mathtt{r}\ A_1, \ldots, x_k : t_k\ \mathtt{r}\ A_k\}$.

Proposition 5 $t\ \mathtt{r}\ A[X := \mathcal{F}] = (t\ \mathtt{r}\ A)[X^+ := \mathcal{F}^{\mathtt{r}}]$.

Proof. Induction on A. ℚed

Theorem 1 (Soundness of Realizability) *Let $'$ be the embedding of definition 3. If $\Gamma \vdash_{\mathsf{AF2}\mu} t : A$, then $\Gamma^{\mathtt{r}} \vdash_{\mathsf{AF2}} t' : t'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ A$*

Proof. Induction on $\vdash_{\mathsf{AF2}\mu}$. We concentrate on the cases for μ:
Case (μE). We have $A \equiv \mathcal{K}\vec{s}$, $t \equiv rE_\mu s$, $\Gamma \vdash r : (\mu X.\mathcal{F})\vec{s}$ and $\Gamma \vdash s : \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K}$. By IH we have $\Gamma^{\mathtt{r}} \vdash r' : r'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ (\mu X.\mathcal{F})\vec{s}$, that is

$$\Gamma^{\mathtt{r}} \vdash r' : \forall X^+.\forall z.z\ \mathtt{r}\ \mathcal{F} \subseteq X \to r'[\vec{x} := \vec{t}\,]z\ \mathtt{r}\ X\vec{s},$$

from this, instantiating X^+ with $\mathcal{K}^{\mathtt{r}}$ and using proposition 5 we get

$$\Gamma^{\mathtt{r}} \vdash r' : \forall z.z\ \mathtt{r}\ (\mathcal{F} \subseteq X)[X := \mathcal{K}] \to r'[\vec{x} := \vec{t}\,]z\ \mathtt{r}\ (X\vec{s})[X := \mathcal{K}].$$

Next we instantiate z with $s'[\vec{x} := \vec{t}\,]$ to get

$$\Gamma^{\mathtt{r}} \vdash r' : s'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K} \to r'[\vec{x} := \vec{t}\,]s'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ \mathcal{K}\vec{s},$$

but by IH we have $\Gamma^{\mathtt{r}} \vdash s' : s'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ \mathcal{F}[X := \mathcal{K}] \subseteq \mathcal{K}$. Therefore by $(\to E)$ we finally obtain

$$\Gamma^{\mathtt{r}} \vdash r's' : r'[\vec{x} := \vec{t}\,]s'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ \mathcal{K}\vec{s},$$

which is the same as

$$\Gamma^{\mathtt{r}} \vdash (rE_\mu s)' : (rE_\mu s)'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ \mathcal{K}\vec{s}.$$

Case (μI). We have $A \equiv (\mu X.\mathcal{F})\vec{s}, t \equiv Cmr, \Gamma \vdash m : \mathcal{F}\mathsf{mon}X$ and $\Gamma \vdash r : \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}$.
We want to show that $\Gamma^{\mathtt{r}} \vdash (Cmr)' : (Cmr)'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ (\mu X.\mathcal{F})\vec{s}$.

It suffices to show

$$\Gamma^{\mathtt{r}} \vdash \lambda y.y(m'(\lambda u.uy)r') : z\ \mathtt{r}\ \mathcal{F} \subseteq X \to (Cmr)'[\vec{x} := \vec{t}\,]z\ \mathtt{r}\ X\vec{s}$$

and to show this is enough to show:

$$\Gamma^{\mathtt{r}}, y : z \ \mathtt{r}\ \mathcal{F} \subseteq X \vdash y(m'(\lambda u.uy)r') : (Cmr)'[\vec{x} := \vec{t}\,]z \ \mathtt{r}\ X\vec{s} \tag{1.1}$$

and then apply $(\to I)$.

Let $\widehat{m} := m'[\vec{x} := \vec{t}\,], \widehat{r} := r'[\vec{x} := \vec{t}\,], \Delta := \Gamma^{\mathtt{r}}, y : z \ \mathtt{r}\ \mathcal{F} \subseteq X$,
we show first the following two derivations:

$$\Delta \vdash \lambda u.uy : \lambda w.wz \ \mathtt{r}\ \mu X.\mathcal{F} \subseteq X \tag{1.2}$$

$$\Delta \vdash m'(\lambda u.uy)r' : \widehat{m}(\lambda w.wz)\widehat{r} \ \mathtt{r}\ \mathcal{F}\vec{s} \tag{1.3}$$

Proof. (*of derivation 1.2*) By rule (Var) and definition of realizability we have:

$$\Delta, u : w \ \mathtt{r}\ (\mu X.\mathcal{F})\vec{y} \vdash u : \forall X^{+}\forall z.z \ \mathtt{r}\ \mathcal{F} \subseteq X \to wz \ \mathtt{r}\ X\vec{y}$$

By $(\forall^2 E), (\forall E), (\to E)$ we obtain

$$\Delta, u : w \ \mathtt{r}\ (\mu X.\mathcal{F})\vec{y} \vdash uy : wz \ \mathtt{r}\ X\vec{y}$$

and by $(\to I)$,

$$\Delta \vdash \lambda u.uy : w \ \mathtt{r}\ (\mu X.\mathcal{F})\vec{y} \to wz \ \mathtt{r}\ X\vec{y}$$

Using (Eq) with $(\lambda w.wz)w = wz \in \mathbb{E}$ we get

$$\Delta \vdash \lambda u.uy : w \ \mathtt{r}\ (\mu X.\mathcal{F})\vec{y} \to (\lambda w.wz)w \ \mathtt{r}\ X\vec{y}$$

Finally by $(\forall I), (\forall I)$ and definition of realizability

$$\Delta \vdash \lambda u.uy : \lambda w.wz \ \mathtt{r}\ \forall\vec{y}.(\mu X.\mathcal{F})\vec{y} \to X\vec{y}$$

and derivation (1.2) is proved. $\mathbb{Q}\mathbb{e}\mathbb{d}$

Proof. (*of derivation 1.3*) By IH we have $\Gamma^{\mathtt{r}} \vdash m' : \widehat{m} \ \mathtt{r}\ \mathcal{F}\mathsf{mon}X$, which implies

$$\Delta \vdash m' : \forall X^{+}\forall Y^{+}\forall w.w \ \mathtt{r}\ X \subseteq Y \to \widehat{m}w \ \mathtt{r}\ \mathcal{F} \subseteq \mathcal{F}[X := Y]$$

Instantiating $X^{+}, Y^{+} := (\mu X.\mathcal{F})^{\mathtt{r}}, X^{+}$ and using proposition 5 we get

$$\Delta \vdash m' : \forall w.w \ \mathtt{r}\ \mu X.\mathcal{F} \subseteq X \to \widehat{m}w \ \mathtt{r}\ \mathcal{F}[X := \mu X.\mathcal{F}] \subseteq \mathcal{F}$$

Instantiating $w := \lambda w.wz$ and using $(\to E)$ with derivation (1.2) we obtain

$$\Delta \vdash m'(\lambda u.uy) : \widehat{m}(\lambda w.wz) \ \mathtt{r}\ \mathcal{F}[X := \mu X.\mathcal{F}] \subseteq \mathcal{F}$$

Unfolding the definition of realizability we get

$$\Delta \vdash m'(\lambda u.uy) : \forall\vec{y}\forall x.x \ \mathtt{r}\ \mathcal{F}[X := \mu X.\mathcal{F}]\vec{y} \to \widehat{m}(\lambda w.wz)x \ \mathtt{r}\ \mathcal{F}\vec{y}$$

By IH we have $\Gamma^{\mathtt{r}} \vdash r' : \widehat{r}\ \mathtt{r}\ \mathcal{F}[X := \mu X.\mathcal{F}]\vec{s}$. From this, by instantiating in the previous derivation $\vec{y}, x := \vec{s}, \widehat{r}$, and applying $(\to E)$ we conclude

$$\Delta \vdash m'(\lambda u.uy)r' : \widehat{m}(\lambda w.wz)\widehat{r}\ \mathtt{r}\ \mathcal{F}\vec{s}$$

and derivation (1.3) is proved $\mathbb{Q}\mathrm{ed}$

Now we proceed to derive (1.1):

From rule (Var) and definition of realizability we obtain

$$\Delta \vdash y : \forall\vec{y}\forall x.x\ \mathtt{r}\ \mathcal{F}\vec{y} \to zx\ \mathtt{r}\ X\vec{y}$$

Instantiating $\vec{y}, x := \vec{s}, \widehat{m}(\lambda w.wz)\widehat{r}$ and applying $(\to E)$ by derivation (1.3) we get

$$\Delta \vdash y(m'(\lambda u.uy)r') : z(\widehat{m}(\lambda w.wz)\widehat{r})\ \mathtt{r}\ X\vec{s}$$

Now using that $z(\widehat{m}(\lambda w.wz)\widehat{r}) = (\lambda z.z(\widehat{m}(\lambda w.wz)\widehat{r}))z \in \mathbb{E}$ by rule (Eq) we have

$$\Delta \vdash y(m'(\lambda u.uy)r') : (\lambda z.z(\widehat{m}(\lambda w.wz)\widehat{r}))z\ \mathtt{r}\ X\vec{s}$$

Finally from $(\lambda z.z(\widehat{m}(\lambda w.wz)\widehat{r}))z \equiv (\lambda z.z(m'(\lambda w.wz)r'))[\vec{x} := \vec{t}\ ]z \equiv (Cmr)'[\vec{x} := \vec{t}\,]z$, we conclude (1.1).

$\mathbb{Q}\mathrm{ed}$

The soundness theorem guarantees the correctness of program extraction. If we have a proof t of a specification A from some assumptions A_i then if we assume that every hypothesis A_i is realized by some program t_i, we obtain a proof t' of the fact that the program $t'[\vec{x} := \vec{t}\,]$ realizes the specification A. Moreover the proof of $t'[\vec{x} := \vec{t}\,]\ \mathtt{r}\ A$ is in some sense the same proof of A because it is just the image of the original proof under the embedding $'$.

4 Final Remarks and Future Work

Systems like $\mathsf{AF2}\mu$ have been developed in [Par92, Raf94]. The essential difference is that our system works on monotone and not just positive inductive definitions and is, in contrast with those of [Par92, Raf94], strongly normalizing. The definition of realizability for inductive definitions, contrary to that of [Par92], has been inspired by the embedding of least fixed points in second order logic, as seen, for instance, in [Raf94, UuVe02].

The ultimate goal is to work out realizability interpretations for several logical systems of monotone inductive definitions, using as realizers not only $\lambda^{\to}$-terms but term systems of monotone inductive types, in particular Curry-versions (when possible) of the different extensions of system F developed in [Mat98], this also implies to define realizability of inductive definitions again as an inductive definition, like the definitions of modified realizability in [Ber95, Ben98] or $\mathtt{q}$-realizability in [Tat93]. First steps

in these direction are reported in [Mir02]. A parallel goal is to extend the Curry-Howard Isomorphism to logical systems of inductive definitions, using the type systems of [Mat98], a reason why we use full second-order logic and not only first-order logic plus inductive predicates.
Unfortunately, for the time being, we only have trivial examples of program extraction, we still need to develop a second important tool, namely an adequate concept of datatype. [Par92] defines a datatype in a model $\mathcal{M}$ as a formula $D[x]$ with $FV(D) = \{x\}$ such that

$$\mathcal{M} \models \forall x \forall y.y \ \mathtt{r} \ D[x] \leftrightarrow y = x \wedge D[x]$$

This concept plays an important role in program extraction, allows to represent typed terms in an untyped setting and provides a connection with Kreisel's modified realizability (see [Mir02]), which is an often used tool in program extraction (see [Ber93, BBS02]). We need either to consider realizability interpretations like that of [Par92], which allow to prove that formulas like $\mathbb{N}[x]$ are datatypes or to formulate an adequate concept of datatype for our purposes. A concept of datatype which seems to be related with our research is a type that is both parametric and extensive in the sense of [Wad01].
To simplify the syntactic machinery and to provide a connection with category theory, which would be useful to consider later coinductive definitions, we want to define logical systems of inductive definitions using the categorical point of view, where inductive predicates represent (weak) initial algebras of functors, and the existence of the algebra is used as an inductive definition principle (see [Geu92, JaRu97]). In this way the complicated inductive definitions in $\mathsf{AF2}\mu$, like that of $\mathbb{N}$, would be simplified to $\mathbb{N} := \mu X.1 + X$, which represents the initial algebra of the functor $T(X) = 1 + X$, where $+$ means coproduct. This is clearly related to algebraic types as presented in [Wad01].

Acknowledgements

Many thanks to the anonymous referees for their suggerences and comments. A special acknowledgement to Ralph Matthes for several fruitful discussions and for carefully reading of several previous drafts.
I'm also grateful to the *Graduiertenkolleg Logik in der Informatik (GK 301)* of the *Deutsche Forschungsgemeinschaft* for providing the funds to assist to this event.

Bibliography

[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 *Typed Lambda Calculus and Applications*, edited by, M. Bezem and J.F. Groote. 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.

[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. 193-217, 1992.

[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.

[Mat98] Ralph Matthes. Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types, Dissertation Universität München, 1999.

[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.

[Mir02] Favio E. Miranda-Perea. A Curry-style Realizability Interpretation for Monotone Inductive Definitions (Extended Version). Unpublished Draft. 2002 (Available upon request).

[Par92] M. Parigot. Recursive programming with proofs. In *Theoretical Computer Science* 94, pp 335-356. Elsevier. 1992.

[Raf94] C. Raffalli. L' Arithmétique Fonctionnelle du Second Ordre avec Points Fixes, Thése de l'Université Paris VII. 1994.

[Tat93] M. Tatsuta. Realizability of Inductive Definitions for Constructive Programming. PhD Thesis, University of Tokyo, 1993.

[UuVe02] T. Uustalu, V. Vene. Least and greatest fixed points in intuitionistic natural deduction. In *Theoretical Computer Science*, 272(1-2),pp. 315-339. Elsevier Science B.V. February 2002.

[Wad01] P. Wadler. The Girard-Reynolds Isomorphism. In TACS2001, *Theoretical Aspects of Computer Software*, edited by N. Kobayashi, B.C. Pierce. LNCS 2215. Springer Verlag. 2001