Realizability of Monotone Coinductive Definitions and Its Application to Program Synthesis

Makoto Tatsuta

Department of Mathematics, Kyoto University, Kyoto 606-8502, JAPAN
e-mail: tatsuta@kusm.kyoto-u.ac.jp

Abstract. Two realizability interpretations of monotone coinductive definitions are studied. One interpretation is defined so that a realizer of a coinductively defined predicate is the same as that of its expansion. For this interpretation, the paper proves that full monotone coinductive definitions are not sound and restricted monotone coinductive definitions are sound. The other interpreration is based on second order logic and can interpret least-upper-bound coinductive definitions, which is generalization of monotone coinductive definitions. By using these interprerations, the paper shows that a program which treats coinductively defined infinite data structures such as streams can be synthesized from a constructive proof of its specification.

1 Introduction

Coinductive definitions have been studied widely [17, 14, 13, 3, 15, 12, 11, 8]. Coinductive definitions formalize the greatest fixed point X satisfying $X \leftrightarrow A(X)$ for a formula $A(X)$. Coinductive definitions are important since they can formalize naturally many important recursive infinite data structures such as streams, infinite lists and infinite trees. Moreover properties of these data structures such as extensional equality of streams can be also formalized by coinductive definitions. Specifications and properties of programs are formally represented in a natural way in a logical system having coinductive definitions. Coinductive definitions formalize notions semantically given by the greatest fixed point. In particular, they can formalize bisimulation of processes [9].

Program synthesis using constructive logic or type theories has been studied widely [7, 2, 5, 10, 11, 16, 17, 20, 13, 14]. The method of program synthesis using constructive logic is called constructive programming. By constructive programming, a program f and the proof of its verification $\forall x(A(x) \to B(x, fx))$ are automatically produced from a constructive proof of the specification $\forall x(A(x) \to \exists y B(x, y))$ of the program, where x means an input, y means an output, $A(x)$ means an input condition and $B(x, y)$ means an input-output relation. By this method, we write a constructive proof to get a program instead of writing a program directly. The method integrates two separated steps of writing a program and verifying a program into a single step of writing a constructive

proof. The method proposes a new programming paradigm based on the notion of proofs-as-programs.

The notion of **q**-realizability is the main tool for constructive programming [5, 16, 17, 20, 6]. By using **q**-realizability, a program and its verification proof are automatically produced from a constructive proof in the following way: From the constructive proof of $\forall x(A(x) \to \exists y B(x,y))$ and the soundness theorem of **q**-realizability, a realizer e is produced such that $(e \quad \mathbf{q} \quad \forall x(A(x) \to \exists y B(x,y)))$. According to the definition of **q**-realizability, the realizer e produces a program f and a proof of its verification $\forall x(A(x) \to B(x, fx))$.

Hence realizability of coinductive definitions is important. By using the logical system having coinductive definitions and realizability, specification of programs with coinductively defined infinite data structures can be naturally represented and such programs can be produced from constructive proofs by using realizability.

Until now, realizability of coinductive definitions was studied only for the restricted case of positive formulas [17]. Positive coinductive definitions formalize the greatest fixed point X satisfying $X \leftrightarrow A(X)$ for a positive formula $A(X)$. [17] studies positive coinductive definitions, which require the condition that $A(X)$ is positive with respect to X. [14] also studies positive coinductive definitions in a type theoretical framework.

Monotone coinductive definitions are the way to define the greatest fixed point X of $A(X)$ under the monotone condition that $A(X)$ is monotone with respect to X. Monotone coinductive definitions are worth studying at least for the following four reasons. (1) Coinductive definitions mean the greatest fixed points. Semantically the monotone condition is sufficient to get the greatest fixed point and in this sense the positive condition is too restrictive. (2) The monotone condition is useful for program modules since it is semantical, while the positive condition is not so useful since it is syntactical. It is explained in detail in Section 6. (3) A non-positive monotone formula can be transformed into a positive formula in many cases. But the transformation loses the expansion-preserving property of realizability since transformation of a formula may change a realizer of the formula. (4) For inductive definitions, realizability of monotone inductive definitions have been already studied [19] and improved the results for the positive inductive definitions given in [6]. Realizability of monotone coinductive definitions can also improve results for positive coinductive definitions.

In this paper, we study realizability of monotone coinductive definitions and its application to program synthesis. We study two kinds of realizability interpretations of monotone coinductive definitions. One interpretation is expansion-preserving, that is, a realizer of a coinductively defined predicate is the same as that of its expansion. The other interpretation is based on second order logic.

For the expansion-preserving realizability interpretation, we present untyped theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$. $\mathbf{TID}_{\nu\mathbf{1}}$ has full monotone coinductive definitions, while $\mathbf{TID}_{\nu\mathbf{0}}$ has only restricted monotone coinductive definitions. We define a **q**-realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ which is expansion-preserving. This **q**-realizability interpretation is an extension of the realizability in [17]. We

show that the realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ in $\mathbf{TID}_{\nu\mathbf{1}}$ is sound and that the realizability interpretation for $\mathbf{TID}_{\nu\mathbf{1}}$ in $\mathbf{TID}_{\nu\mathbf{1}}$ is not sound. We explain that there is no expansion-preserving realizability interpretation of full monotone coinductive definitions. The soundness proof for $\mathbf{TID}_{\nu\mathbf{0}}$ entails and simplifies the soundness proof for the positive case in [17] as a special case.

For the realizability interpretation based on second order logic, we present an untyped theory $\mathbf{TID}_{\nu\mathbf{2}}$, which has least-upper-bound coinductive definitions. Least-upper-bound coinductive definitions cover monotone coinductive definitions and are more natural when we think the meaning given by their coding. $\mathbf{TID}_{\nu\mathbf{2}}$ includes $\mathbf{TID}_{\nu\mathbf{1}}$. We define a $\mathbf{q}$-realizability interpretation for $\mathbf{TID}_{\nu\mathbf{2}}$ in a second order system $\mathbf{T_2}$, using the higher order coding and $\mathbf{q}$-realizability for second order logic [4]. We prove its soundness.

We show these two realizability interpretation can be used for program synthesis. In particular, the program extraction theorem and the stream program extraction theorem are proved by using these interpretations.

In our words, [13] and [14] also studied an expansion-preserving realizability interpretation for positive inductive and coinductive definitions in a type theoretical framework. In their words, our reresults prove the followings: For $\nu X x A\langle t\rangle$ such that A may not be positive with respect to X, (1) the condition $(X \to Y) \to (A \to A[Y/X])$ does not imply the conservation lemma, and (2) the condition $(X \subseteq Y) \to (A \subseteq A[Y/X])$ implies the conservation lemma where $(X \subseteq Y)$ denotes $\forall x(x \in X \to x \in Y)$. The condition for (2) corresponds to the condition (MONO-Q) in our theory.

In Section 2, we define the theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ and explain that coinductive definitions are useful for program specification. In Section 3, we discuss models of $\mathbf{TID}_{\nu\mathbf{1}}$ and prove its consistency. In Section 4, we present the $\mathbf{q}$-realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$. We prove the soundness theorem for $\mathbf{TID}_{\nu\mathbf{0}}$ and show that the interpretation for $\mathbf{TID}_{\nu\mathbf{1}}$ is not sound. In Section 5, we present the $\mathbf{q}$-realizability interpretation of least-upper-bound coinductive definitions. In Section 6, we explain that a program with streams can be synthesized by using these results.

2 Theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$

We present the theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ in this section. Both are theories with λ-calculus and coinductive definitions, and based on Beeson's elementary theory of operations and numbers [1]. $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ differ only in the rules of coinductive definitions. These systems correspond to Martin-Löf's Type Theory with coinductive types.

In the paper, we choose λ-calculus with pairing and natural numbers as the target programming language for simplicity. We suppose that the evaluation strategy of terms is lazy or call-by-name because coinductive definitions are useful to formalize infinite objects such as streams, which are non-terminating terms.

2.1 Language of $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$

The language of $\mathbf{TID}_{\nu\mathbf{0}}$ and that of $\mathbf{TID}_{\nu\mathbf{1}}$ are the same. This is based on a first order language and extended for coinductive definitions of predicates.

The constants are: $\mathbf{p}$, $\mathbf{p}_0$, $\mathbf{p}_1$, 0, $\mathbf{s}_{\mathrm{N}}$, $\mathbf{p}_{\mathrm{N}}$, $\mathbf{d}$.

We have natural numbers as primitives, which are given by 0, the successor function $\mathbf{s}_{\mathrm{N}}$ and the predecessor function $\mathbf{p}_{\mathrm{N}}$. We also have the pairing functions $\mathbf{p}$, $\mathbf{p}_0$ and $\mathbf{p}_1$ as built-in, which correspond to `cons`, `car` and `cdr` in LISP respectively. $\mathbf{d}$ is a combinator judging equality of numbers and corresponds to an if-then-else statement in a usual programming language.

We have only one function symbol $\mathbf{App}$ whose arity is 2. It means a functional application of terms.

Terms t are defined by: $t ::= x|c|\mathbf{App}(t,t)|\lambda x.t$ where x stands for a variable and c stands for a constant.

For terms s, t, we abbreviate $\mathbf{App}(s,t)$ as st. For terms s, t, we also use abbreviations $(s,t) \equiv \mathbf{p}st$, $t_0 \equiv \mathbf{p}_0 t$, $t_1 \equiv \mathbf{p}_1 t$, $t_{i_1 \ldots i_n 0} \equiv \mathbf{p}_0(t_{i_1 \ldots i_n})$ and $t_{i_1 \ldots i_n 1} \equiv \mathbf{p}_1(t_{i_1 \ldots i_n})$ for $i_j \equiv 0$ or $1 \quad (1 \leq j \leq n)$.

The predicate symbols are: $\bot$, $\mathbf{N}$, $=$.

$\bot$ means contradiction. $\mathbf{N}(a)$ means that a is a natural number. $a = b$ means that a equals b.

We have predicate variables, which a first order language does not have. The predicate variables are: $X, Y, Z, \ldots, X^*, Y^*, Z^*, \ldots$. Each predicate variable has fixed arity. We assume that the arity of X^* is $n+1$ if the arity of X is n.

We abbreviate $\mathrm{Y}(\lambda x.t)$ as $\mu x.t$ where $\mathrm{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Definition 2.1 (Formula).

1. If a, b are terms, $\bot$, $\mathbf{N}(a)$, $a = b$ are formulas. We call them atomic formulas.
2. If X is a predicate variable of arity n and $t_1, \ldots, t_n$ are terms, $X(t_1, \ldots, t_n)$ is a formula.
3. If A and B are formulas, $A \,\&\, B$, $A \vee B$ and $A \to B$ are formulas.
4. If A is a formula, $\forall x A$ and $\exists x A$ are formulas.
5. $(\lambda x_1 \ldots x_n.A)(t_1, \ldots, t_n)$ is a formula if A is a formula and $t_1, \ldots, t_n$ are terms.
6. $(\nu X.\lambda x_1 \ldots x_n.A)(t_1, \ldots, t_n)$ is a formula where X is a predicate variable of arity n, A is a formula and $t_1, \ldots, t_n$ are terms.

$\neg A$ denotes $A \to \bot$.

The last case corresponds to coinductively defined predicates. Note that X and $x_1, \ldots, x_n$ may occur freely in A. The intuitive meaning of the formula $(\nu X.\lambda x_1 \ldots x_n.A(X, x_1, \ldots, x_n))(t_1, \ldots, t_n)$ is as follows: Let P be the predicate of arity n which is the greatest solution of the equation

$$P(x_1, \ldots, x_n) \leftrightarrow A(P, x_1, \ldots, x_n).$$

Then $(\nu X.\lambda x_1 \ldots x_n.A(X, x_1, \ldots, x_n))(t_1, \ldots, t_n)$ means $P(t_1, \ldots, t_n)$.

We abbreviate a sequence as a bold type symbol, for example, $x_1, \ldots, x_n$ as $\mathbf{x}$, and $t_1, \ldots, t_n$ as $\mathbf{t}$.

Example 2.2. We give an example of a formula. We suppose X is a predicate variable of arity 1. Then

$$(\nu X.\lambda x.x = (x_0, x_1) \,\&\, x_0 = 0 \,\&\, X(x_1))(x)$$

is a formula.

Definition 2.3 (Abstract).

1. A predicate symbol of arity n is an abstract of arity n.
2. A predicate variable of arity n is an abstract of arity n.
3. If A is a formula, $\lambda x_1 \ldots x_n.A$ is an abstract of arity n.
4. If $(\nu X.\lambda x_1 \ldots x_n.A)(x_1, \ldots, x_n)$ is a formula, $\nu X.\lambda x_1 \ldots x_n.A$ is an abstract of arity n.

For a formula A, a predicate variable X of arity n and an abstract F of arity n, the formula $A_X[F]$ denotes the formula obtained from A by replacing all the occurrences of X by F.

2.2 Basic Axioms and Inference Rules of $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$

Theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ are based on intuitionistic natural deduction NJ.

We give axioms and inference rules except those of coinductive definitions in this subsection. They are usual axioms and inference rules of λ-calculus, pairing and natural numbers and we only list them here.

Axioms for Equality:

$$x = x, \tag{E1}$$

$$x = y \,\&\, A(x) \to A(y). \tag{E2}$$

Axiom for β-conversion:

$$(\lambda x.a)b = a[x := b] \tag{B1}$$

for terms a and b, where $a[x := b]$ means the term obtained from a by replacing every free occurrence of x by b.

Axioms for Pairing:

$$\mathbf{p}_0(\mathbf{p}xy) = x, \tag{P1}$$

$$\mathbf{p}_1(\mathbf{p}xy) = y, \tag{P2}$$

Axioms for Natural Numbers:

$$\mathbf{N}(0), \tag{N1}$$

$$\mathbf{N}(x) \to \mathbf{N}(\mathbf{s}_\mathrm{N}x), \tag{N2}$$

$$\mathbf{N}(x) \to \mathbf{p}_\mathrm{N}(\mathbf{s}_\mathrm{N}x) = x, \tag{N3}$$

$$\mathbf{N}(x) \to \mathbf{s}_\mathrm{N}x \neq 0, \tag{N4}$$

$$A(0) \,\&\, \forall x(\mathbf{N}(x) \,\&\, A(x) \to A(\mathbf{s}_\mathrm{N}x)) \to \forall x(\mathbf{N}(x) \to A(x)). \tag{N5}$$

Axioms for $\mathbf{d}$:

$$\forall xyab(\mathbf{N}(x) \,\&\, \mathbf{N}(y) \,\&\, x = y \to \mathbf{d}xyab = a), \tag{D1}$$

$$\forall xyab(\mathbf{N}(x) \,\&\, \mathbf{N}(y) \,\&\, x \neq y \to \mathbf{d}xyab = b). \tag{D2}$$

Axioms for Abstracts:

$$(\lambda \mathbf{x}.A)(\mathbf{x}) \leftrightarrow A. \tag{A1}$$

2.3 Inference Rules of Coinductive Definitions

We discuss inference rules of coinductive definitions. Let $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$ where $\mathbf{x}$ is a sequence of variables whose length is the arity of a predicate variable X and $A(X)$ is a formula. $A(C)$ denotes the formula obtained from $A(X)$ by replacing all the occurrences of X by C. Suppose that

$$(\nu 1) \equiv \forall \mathbf{x}(\nu(\mathbf{x}) \to A(\nu)),$$
$$(\nu 2) \equiv \forall \mathbf{x}(C(\mathbf{x}) \to A(C)) \to \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})),$$
$$(\text{MONO}) \equiv \forall \mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall \mathbf{x}(A(X) \to A(Y)),$$
$$(\text{MONO-Q}) \equiv \forall \mathbf{x} r(X^*(r,\mathbf{x}) \to Y^*(r,\mathbf{x})) \to$$
$$\forall \mathbf{x} r((r \quad \mathbf{q}_X[X,X^*] \quad A(X)) \to (r \quad \mathbf{q}_X[X,Y^*] \quad A(X)))$$

where X, Y, X^* and Y^* are predicate variables, $A(X)$ is a formula and $C(\mathbf{x})$ is a formula. The formula $(e \quad \mathbf{q}_X[X,X^*] \quad A(X))$ denotes the realizability of the formula A and is defined in Section 4.

The condition (MONO) states that the formula $A(X)$ is monotone with respect to X.

The condition (MONO-Q) states that the formula $(e \quad \mathbf{q}_X[X,X^*] \quad A(X))$ is monotone with respect to X^*. The formula $(e \quad \mathbf{q}_X[X,X^*] \quad A(X))$ means the formula obtained from the usual $\mathbf{q}$-realizability $(e \quad \mathbf{q} \quad A(X))$ by replacing every occurrence of the formula $(r \quad \mathbf{q} \quad X(x))$ by the formula $X^*(r,x)$. The condition means that the realizability $(e \quad \mathbf{q} \quad A(X))$ is monotone with respect to the occurrences $(r \quad \mathbf{q} \quad X(x))$. The condition (MONO-Q) is proved to bring the soundness of the realizability in Section 4. In words of AF_2 [13, 14], the condition (MONO-Q) represents $X \subseteq Y \to A(X) \subseteq A(Y)$ where $X \subseteq Y$ denotes $\forall x(x \in X \to x \in Y)$.

We have 2 inference rules for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ respectively. For $\mathbf{TID}_{\nu\mathbf{0}}$ the inference rules are as follows:

$$\frac{(\text{MONO}) \quad (\text{MONO-Q})}{(\nu 1)} \ (\nu R1')$$

$$\frac{(\text{MONO}) \quad (\text{MONO-Q})}{(\nu 2)} \ (\nu R2')$$

where X, Y, X^* and Y^* do not occur freely in open assumptions of deductions of the premises.

For $\mathbf{TID}_{\nu\mathbf{1}}$ the inference rules are as follows:

$$\frac{(\text{MONO})}{(\nu 1)} \ (\nu R1)$$

$$\frac{(\text{MONO})}{(\nu 2)} \ (\nu R2)$$

where X and Y do not occur freely in open assumptions of a deduction of the premise.

Proposition 4.13 will state that the positivity condition implies (MONO) and (MONO-Q). In fact, the set of formulas satisfying the positivity condition is a proper subset of the set of formulas satisfying (MONO) and (MONO-Q), and the set of formulas satisfying (MONO) and (MONO-Q) is a proper subset of the set of formulas satisfying (MONO). If $A(X)$ is $\bot \,\&\, \neg X$, then $A(X)$ is in the set of formulas satisfying (MONO) and (MONO-Q), but is not in the set of formulas satisfying the positivity condition. If $A(X)$ is $X \to X$, $A(X)$ is in the set of formulas satisfying (MONO), but is not in the set of formulas satisfying (MONO) and (MONO-Q).

We define a theory $\mathbf{TID}^-$ as the theory whose language is the same as $\mathbf{TID}_{\nu\mathbf{1}}$ and whose axioms and inference rules are only basic axioms and inference rules given in Subsection 2.2.

2.4 Monotone Coinductive Definitions of Predicates

We will explain how useful the coinductive definitions are to formalize properties of programs.

First, We explain the coinductive definitions. Let $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$ where $\mathbf{x}$ is a sequence of variables whose length is the arity of a predicate variable X and $A(X)$ is a formula. Suppose that (MONO) and (MONO-Q) hold for the formula $A(X)$. So we can use $(\nu 1)$ and $(\nu 2)$ for $A(X)$ in each theory $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$. We will discuss $(\nu 1)$ and $(\nu 2)$.

By $(\nu 1)$ and $(\nu 2)$,

$$\nu(\mathbf{x}) \leftrightarrow A(\nu) \qquad (\nu 1')$$

holds. This ν is the greatest solution of the equation for a predicate variable X:

$$X(\mathbf{x}) \leftrightarrow A(X).$$

Because

$$\forall \mathbf{x}(C(\mathbf{x}) \leftrightarrow A(C)) \to \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x}))$$

holds by $(\nu 2)$ for any formula $C(\mathbf{x})$.

By the facility of coinductive definitions, many useful infinite recursive data structures such as streams, infinite lists and infinite trees can be defined inside the theory [17]. The coinduction principle corresponding to each data structure is derived from $(\nu 2)$. We will give some examples.

Example 2.4 (Bit Stream). We will define the predicate BS which means that the argument is a bit stream. Here we suppose that a stream is represented by an infinite list constructed by pairing. One way to define the predicate BS is as follows:

$$BS(x) \equiv x = (x_0, x_1) \,\&\, (x_0 = 0 \vee x_0 = 1) \,\&\, BS(x_1).$$

But it is not a definition of BS in a usual sense since BS appears also in the right hand side which is the body of the definition. What we want to present by this equation is the greatest solution of the recursive equation for a predicate variable X:

$$X(x) \leftrightarrow x = (x_0, x_1) \,\&\, (x_0 = 0 \vee x_0 = 1) \,\&\, X(x_1).$$

In $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ this solution can be represented by the formula

$$(\nu X.\lambda x.x = (x_0, x_1) \,\&\, (x_0 = 0 \vee x_0 = 1) \,\&\, X(x_1))(x)$$

and ($\nu 1$) and ($\nu 2$) state that this formula is actually the greatest solution of the equation. By this facility we can define BS in $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ as follows:

$$BS(x) \equiv (\nu X.\lambda x.x = (x_0, x_1) \,\&\, (x_0 = 0 \vee x_0 = 1) \,\&\, X(x_1))(x).$$

($\nu 2$) for BS represents coinduction for bit streams classified by BS. ($\nu 2$) represents the generalized coinduction principle.

Let $\overline{0}$ be $\mu s.(0, s)$. $\overline{0}$ represents the zero stream whose elements are all 0, that is, $\overline{0} = (0, (0, (0, \ldots)))$. By ($\nu 2$), $BS(\overline{0})$ is proved.

Example 2.5 (Extensional Equality of Streams). The extensional equality $s \approx t$ of two streams s and t is represented by coinductive definitions as follows:

$$s \approx t \equiv (\nu X.\lambda xy.x_0 = y_0 \,\&\, X(x_1, y_1))(s, t).$$

Example 2.6 (Infinite Lists). The predicate $InfList(x)$ which means that x is an infinite (maybe finite) list of natural numbers is represented by coinductive definitions. We suppose that the empty list is represented by 0.

$$InfList(x) \equiv (\nu X.\lambda x.x = 0 \vee \exists yz(x = (y, z) \,\&\, \mathbf{N}(y) \,\&\, X(z)))(x).$$

Example 2.7 (Infinite Trees). The predicate $InfTree(x)$ which means that x is an infinite tree is represented by coinductive definitions. We suppose that the leaf is a natural number and the node is represented by (y, z, w) where y is a natural number and z and w are infinite trees.

$$InfTree(x) \equiv$$
$$(\nu X.\lambda x.\mathbf{N}(x) \vee \exists yzw(x = (y, z, w) \,\&\, \mathbf{N}(y) \,\&\, X(z) \,\&\, X(w)))(x).$$

The theories $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$ give a framework in which specification formulas of programs are represented in a natural way. The facility of coinductive definitions can formalize notions given by the greatest fixed point such as streams in an expressive way.

3 Models of $\mathbf{TID}_{\nu\mathbf{1}}$

We will briefly explain semantics of $\mathbf{TID}_{\nu\mathbf{1}}$ by giving its intended model.

We will use the well-known greatest fixed point theorem for model construction.

Greatest Fixed Point Theorem. Let S be a set and $p(S)$ be the power set of S. If $f : p(S) \to p(S)$ is a monotone function, there exists a such that $a \in p(S)$ and

1. $f(a) = a$,
2. For any $b \in p(S)$, if $b \subset f(b)$, then $b \subset a$.

The notation $\mathrm{gfp}(f)$ denotes the element a.

We will construct a model of $\mathbf{TID}_{\nu\mathbf{1}}$ extending an arbitrary model of $\mathbf{TID}^-$. The models of $\mathbf{TID}^-$ are the same as the models of Beeson's theory **EON** and many models of that theory are known [1]. We denote the universe by U.

We will define $\rho \models A$ in almost the same way as for first order logic, where A is a formula and ρ is an environment which assigns an element of U to a first

order variable and a subset of U^n to a predicate variable of arity n. We present only the definition for the case $X(\mathbf{t})$ for a predicate variable X and the case $(\nu X.\lambda \mathbf{x}.A(X))(\mathbf{t})$.

Case $X(\mathbf{t})$. $\rho \models X(\mathbf{t})$ is defined as $\mathbf{t} \in \rho(X)$.

Case $(\nu X.\lambda \mathbf{x}.A(X))(\mathbf{t})$. Define F as follows:

$F : p(U^n) \to p(U^n)$,

$F(V) = \{\mathbf{x} \in U^n \mid \rho[X := V] \models A(X)\}$,

where $\rho[X := V]$ is defined as follows:

$\rho[X := V](X) = V$.

$\rho[X := V](x) = \rho(x)$, if x is not X.

Then $\rho \models (\nu X.\lambda \mathbf{x}.A(X))(\mathbf{t})$ is defined as $\mathbf{t} \in \mathrm{gfp}(F)$ if F is monotone. $\rho \models (\nu X.\lambda \mathbf{x}.A(X))(\mathbf{t})$ is defined as false if F is not monotone.

Theorem 3.1. *If* $\mathbf{TID}_{\nu\mathbf{1}} \vdash A$ *holds, then* $\rho \models A$ *holds for any environment* ρ.

This theorem is proved in the appendix.

Theorem 3.2. $\mathbf{TID}_{\nu\mathbf{0}}$ *and* $\mathbf{TID}_{\nu\mathbf{1}}$ *are consistent.*

This theorem is proved immediately from Theorem 3.1.

4 q-Realizability for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$

In this section, we give a **q**-realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{1}}$, which is expansion-preserving.

We say a realizability interpretation is expansion-preserving if a realizer of a coinductively defined predicate is the same as that of its expansion, that is, $(e \quad \mathbf{q} \quad \nu(x)) \leftrightarrow (e \quad \mathbf{q} \quad A(\nu, x))$ holds where $\nu \equiv \nu X.\lambda x.A(X, x)$. The expansion-preserving property is important for the following reasons: (1) Since we give the same semantics to $\nu(x)$ and $A(\nu, x)$ in our model, it is naturally expected to give the same interpretation to $\nu(x)$ and $A(\nu, x)$. (2) A realizer of coinduction can give a loop structure to a synthesized program. This idea of the expansion-preserving property is also adopted in the theory AF_2 [13, 14].

First, we give an intuitive explanation of our notion of **q**-realizability. We start with usual **q**-realizability and sketch how $(\nu X.\lambda x.A(X, x))(x)$ is interpreted. Let ν be $\nu X.\lambda x.A(X, x)$. Since we require the expansion-preserving property, it is natural to define $(e \quad \mathbf{q} \quad \nu(x))$ as $\nu^*(e, x)$ where $\nu^*(e, x)$ is the greatest solution of the recursive equation for a predicate variable X^*:

$X^*(e, x) \leftrightarrow (e \quad \mathbf{q} \quad A(\nu, x))[(r \quad \mathbf{q} \quad \nu(y)) := X^*(r, y)]$.

where $[(r \quad \mathbf{q} \quad \nu(y)) := X^*(r, y)]$ of the right hand side means replacing each subformula $(r \quad \mathbf{q} \quad \nu(y))$ by the formula $X^*(r, y)$ in the formula $(e \quad \mathbf{q} \quad A(\nu, x))$. We get Definition 4.1 of our notion of **q**-realizability by describing this idea in a rigorous way. This interpretation is an extension of interpretations of positive coinductive definitions [17].

Our **q**-realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ and that for $\mathbf{TID}_{\nu\mathbf{1}}$ are defined in the same way.

Definition 4.1 (q-Realizability). Suppose A is a formula, $n \geq 0$, $X_1, \ldots, X_n$ is a sequence of predicate variables of arity $m_1, \ldots, m_n$ respectively and $F_1, G_1, \ldots, F_n, G_n$ is a sequence of abstracts of arity $m_1, m_1+1, \ldots, m_n, m_n+1$ respectively. A formula

$(e \quad \mathbf{q}_{X_1,\ldots,X_n}[F_1, G_1, \ldots, F_n, G_n] \quad A)$

is defined by induction on the construction of A as follows.

We use the following abbreviations.

$(e \quad \mathbf{q}' \quad A) \equiv (e \quad \mathbf{q}_{X_1,\ldots,X_n}[F_1, G_1, \ldots, F_n, G_n] \quad A)$,

$(e \quad \mathbf{q}'_X[F, G] \quad A) \equiv (e \quad \mathbf{q}_{X_1,\ldots,X_n,X}[F_1, G_1, \ldots, F_n, G_n, F, G] \quad A)$,

$\mathbf{F} \equiv F_1, \ldots, F_n$,

$\mathbf{X} \equiv X_1, \ldots, X_n$.

1. $(e \quad \mathbf{q}' \quad A) \equiv e = 0 \,\&\, A$ if A is an atomic formula.
2. $(e \quad \mathbf{q}' \quad X_i(\mathbf{t})) \equiv F_i(\mathbf{t}) \,\&\, G_i(e, \mathbf{t})$.
3. $(e \quad \mathbf{q}' \quad Y(\mathbf{t})) \equiv Y(\mathbf{t}) \,\&\, Y^*(e, \mathbf{t})$ if $Y \not\equiv X_i \quad (1 \leq i \leq n)$.
4. $(e \quad \mathbf{q}' \quad A \,\&\, B) \equiv (e_0 \quad \mathbf{q}' \quad A) \,\&\, (e_1 \quad \mathbf{q}' \quad B)$.
5. $(e \quad \mathbf{q}' \quad A \vee B) \equiv \mathbf{N}(e_0) \,\&\, (e_0 = 0 \to (e_1 \quad \mathbf{q}' \quad A)) \,\&\, (e_0 \neq 0 \to (e_1 \quad \mathbf{q}' \quad B))$.
6. $(e \quad \mathbf{q}' \quad A \to B) \equiv (A_{\mathbf{X}}[\mathbf{F}] \to B_{\mathbf{X}}[\mathbf{F}]) \,\&\, \forall q((q \quad \mathbf{q}' \quad A) \to (eq \quad \mathbf{q}' \quad B))$.
7. $(e \quad \mathbf{q}' \quad \forall x A(x)) \equiv \forall x(ex \quad \mathbf{q}' \quad A(x))$.
8. $(e \quad \mathbf{q}' \quad \exists x A(x)) \equiv (e_1 \quad \mathbf{q}' \quad A(e_0))$.
9. $(e \quad \mathbf{q}' \quad (\lambda \mathbf{x}.A)(\mathbf{t})) \equiv (\lambda r \mathbf{x}.(r \quad \mathbf{q}' \quad A))(e, \mathbf{t})$.
10. $(e \quad \mathbf{q}' \quad \nu(\mathbf{t})) \equiv \nu(\mathbf{t}) \,\&\, (\nu Y^*.\lambda r \mathbf{x}.(r \quad \mathbf{q}'_Y[\nu_{\mathbf{X}}[\mathbf{F}], Y^*] \quad A(Y)))(e, \mathbf{t})$ where $\nu \equiv \nu Y.\lambda \mathbf{x}.A(Y)$.

We write simply $(e \quad \mathbf{q} \quad A)$ if $n = 0$.

Our realizability interpretation is a usual realizability interpretation accompanied with substitution. The formula $(e \quad \mathbf{q}_X[F, G] \quad A)$ is obtained from the usual realizability $(e \quad \mathbf{q} \quad A)$ by replacing all the occurrences of X by F and X^* by G. In particular, if $n = 0$, the above $(e \quad \mathbf{q}' \quad A)$ is exactly the same as the usual $\mathbf{q}$-realizability.

The interpretation $(e \quad \mathbf{q} \quad A)$ corresponds to $e : A$ in type theories.

The last case of the definition is slightly changed from the definition given in [17] to prove Proposition 4.3, since $(\nu 1)$ and $(\nu 2)$ cannot be used without extra conditions (MONO) and (MONO-Q), on the other hand, the proposition is proved by using $(\nu 1)$ and $(\nu 2)$ in [17].

Example 4.2. We will give the interpretation of the bit stream predicate $BS(x)$ as an example. The interpretation $(e \quad \mathbf{q} \quad BS(x))$ is

$BS(x) \,\&\, (\nu X^*.\lambda r x.(r \quad \mathbf{q}_X[BS, X^*] \quad x = (x_0, x_1) \,\&$

$(x_0 = 0 \vee x_0 = 1) \,\&\, X(x_1)))(e, x)$.

That is

$BS(x) \,\&\, (\nu X^*.\lambda r x.(r_0 \quad \mathbf{q}_X[BS, X^*] \quad x = (x_0, x_1)) \,\&$

$(r_{10} \quad \mathbf{q}_X[BS, X^*] \quad x_0 = 0 \vee x_0 = 1) \,\&\, (r_{11} \quad \mathbf{q}_X[BS, X^*] \quad X(x_1)))(e, x)$.

Here we have

$$(r_0 \quad \mathbf{q}_X[BS, X^*] \quad x = (x_0, x_1)) \equiv r_0 = 0 \,\&\, x = (x_0, x_1),$$
$$(r_{10} \quad \mathbf{q}_X[BS, X^*] \quad x_0 = 0 \vee x_0 = 1) \equiv \mathbf{N}(r_{100}) \,\&$$
$$(r_{100} = 0 \to r_{101} = 0 \,\&\, x_0 = 0) \,\&\, (r_{100} \neq 0 \to r_{101} = 0 \,\&\, x_0 = 1),$$
$$(r_{11} \quad \mathbf{q}_X[BS, X^*] \quad X(x_1)) \equiv BS(x_1) \,\&\, X^*(r_{11}, x_1).$$

Therefore the interpretation $(e \quad \mathbf{q} \quad BS(x))$ is

$$BS(x) \,\&\, (\nu X^*.\lambda r x.(r_0 = 0 \,\&\, x = (x_0, x_1)) \,\&\, (\mathbf{N}(r_{100}) \,\&$$
$$(r_{100} = 0 \to r_{101} = 0 \,\&\, x_0 = 0) \,\&\, (r_{100} \neq 0 \to r_{101} = 0 \,\&\, x_0 = 1)) \,\&$$
$$(BS(x_1) \,\&\, X^*(r_{11}, x_1)))(e, x).$$

Proposition 4.3. $\mathbf{TID}_{\nu\mathbf{0}} \vdash (e \quad \mathbf{q}_{\mathbf{X}}[F_1, G_1, \ldots, F_n, G_n] \quad A) \to A_{\mathbf{X}}[\mathbf{F}]$.

The claim is proved by induction on the construction of the formula A.

We will show that this interpretation works well for $\mathbf{TID}_{\nu\mathbf{0}}$ but goes wrong for $\mathbf{TID}_{\nu\mathbf{1}}$. That is, it will be shown that the interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ is sound but the interpretation for $\mathbf{TID}_{\nu\mathbf{1}}$ is not sound. The difference is brought by the additional condition (MONO-Q). In general, we cannot conclude that $(e \quad \mathbf{q}_X[X, X^*] \quad A)$ is monotone with respect to X^* even if A is monotone with respect to X. For such A, we cannot use $(\nu 1)$ and $(\nu 2)$ for $(e \quad \mathbf{q} \quad (\nu X.\lambda x.A)(t))$ in $\mathbf{TID}_{\nu\mathbf{1}}$. On the other hand, we can always use $(\nu 1)$ and $(\nu 2)$ for $(e \quad \mathbf{q} \quad (\nu X.\lambda x.A)(t))$ in $\mathbf{TID}_{\nu\mathbf{0}}$ since the additional (MONO-Q) guarantees that $(e \quad \mathbf{q}_X[X, X^*] \quad A)$ is monotone with respect to X^*.

Suppose we could define realizability $(e \quad \mathbf{q}_2 \quad A)$ so that $(e \quad \mathbf{q}_2 \quad \nu) \leftrightarrow (e \quad \mathbf{q}_2 \quad A(\nu))$ where $\nu \equiv \nu X.A(X)$ and (MONO-Q) does not hold for $A(X)$. Then the meaning of $(e \quad \mathbf{q}_2 \quad \nu)$ is a fixed point of a non-monotone function F such that $(e \quad \mathbf{q}_2 \quad A(X))$ means $F(X^*)$. But it is hopeless in general to find a meaning of a fixed point of a non-monotone function. The next theorem shows this fact for the case of full monotone coinductive definitions.

Theorem 4.4. *The* $\mathbf{q}$*-realizability interpretation is not sound for* $\mathbf{TID}_{\nu\mathbf{1}}$*. That is, there is a formula* A *such that* $\mathbf{TID}_{\nu\mathbf{1}} \vdash A$ *holds but* $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad A)$ *does not hold for any* e*.*

This theorem is proved in the appendix.

We will show soundness of the realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ in $\mathbf{TID}_{\nu\mathbf{1}}$.

Proposition 4.5. *For a formula* $A(X)$ *and an abstract* C*,*

$$(e \quad \mathbf{q} \quad A(X))[X := C, X^* := \lambda r\mathbf{x}.(r \quad \mathbf{q} \quad C(\mathbf{x}))] \leftrightarrow (e \quad \mathbf{q} \quad A(C))$$

holds, where $[X := C, X^* := \lambda r\mathbf{x}.(r \quad \mathbf{q} \quad C(\mathbf{x}))]$ *denotes a substitution of abstracts for predicate variables.*

This claim is proved by induction on A.

Proposition 4.6. *Let* $\nu \equiv \nu X.\lambda\mathbf{x}.A(X)$ *and suppose that* (MONO) *and* (MONO-Q) *hold for* $A(X)$*. Then*

$$\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad \nu(\mathbf{x})) \leftrightarrow (e \quad \mathbf{q} \quad A(\nu))$$

holds.

This proposition is proved in the appendix. This proposition shows that the the realizability interpretation is expansion-preserving.

We give realizers of ($\nu 1$) and ($\nu 2$) in the next two propositions.

Proposition 4.7. *Suppose that* $\mathbf{TID}_{\nu\mathbf{0}} \vdash$ (MONO) *and* $\mathbf{TID}_{\nu\mathbf{0}} \vdash$ (MONO-Q) *hold for a formula* $A(X)$. *Let* $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$. *Then*

$\mathbf{TID}_{\nu\mathbf{1}} \vdash \lambda \mathbf{x} r.r \quad \mathbf{q} \quad (\nu 1)$

holds.

This proposition is proved immediate from Proposition 4.6.

Proposition 4.8. *Let* $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$ *and suppose that* (MONO) *and* (MONO-Q) *hold for a formula* $A(X)$. *Then*

$\mathbf{TID}_{\nu\mathbf{1}} \vdash \lambda q.\mu f.\lambda \mathbf{x} r.mf\mathbf{x}(q\mathbf{x}r) \quad \mathbf{q} \quad (\nu 2)$

holds where

$\mathbf{TID}_{\nu\mathbf{1}} \vdash m \quad \mathbf{q} \quad$ (MONO).

This proposition is proved in the appendix.

Theorem 4.9 (Soundness Theorem). *If* $\mathbf{TID}_{\nu\mathbf{0}} \vdash A$, *we can get a term* e *from a proof and* $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad A)$ *holds where all the free variables of* e *are included in all the free variables of* A.

Proof 4.10. By induction on the proof of $\mathbf{TID}_{\nu\mathbf{0}} \vdash A$. Cases are classified by the last rule applied to the proof. Cases for basic axioms and inference rules are proved in almost the same way as [17]. Case ($\nu R1'$) is proved by using Proposition 4.7. Case ($\nu R2'$) is proved by using Proposition 4.8. □

By the soundness of the realizability, we get the term existence property, the disjunction property and the program extraction theorem. Our realizability interpretation works well whenever the usual realizability works. We also get the soundness proof for **r**-realizability by modifying the definition of the realizability and the soundness proof described here so that we get consistency with some choice axioms.

Theorem 4.11 (Term Existence Property and Disjunction Property). *(1) If* $\mathbf{TID}_{\nu\mathbf{0}} \vdash \exists x A(x)$, *we can get a term* t *from the proof and* $\mathbf{TID}_{\nu\mathbf{1}} \vdash A(t)$ *holds.*

(2) If $\mathbf{TID}_{\nu\mathbf{0}} \vdash A \vee B$, *then* $\mathbf{TID}_{\nu\mathbf{1}} \vdash A$ *or* $\mathbf{TID}_{\nu\mathbf{1}} \vdash B$ *holds.*

Proof 4.12. (1) Suppose $\mathbf{TID}_{\nu\mathbf{0}} \vdash \exists x A(x)$. By Theorem 4.9, we can get a term e from the proof and $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad \exists x A(x))$. By Definition 4.1 and Proposition 4.3, it implies $\mathbf{TID}_{\nu\mathbf{1}} \vdash A(e_0)$. By putting $t := e_0$, we have the claim.

(2) Suppose $\mathbf{TID}_{\nu\mathbf{0}} \vdash A \vee B$. By Theorem 4.9, we can get a term e from the proof and $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad A \vee B)$. By Definition 4.1, $\mathbf{TID}_{\nu\mathbf{1}} \vdash e_0 = 0$ or $\mathbf{TID}_{\nu\mathbf{1}} \vdash e_0 \neq 0$ holds, and hence $\mathbf{TID}_{\nu\mathbf{1}} \vdash A$ or $\mathbf{TID}_{\nu\mathbf{1}} \vdash B$ holds. □

We will show the discussion of the realizability given in [17] is a special case of the above discussion.

Proposition 4.13. *If X occurs only positively in $A(X)$,* (MONO) *and* (MONO-Q)*hold.*

Corollary 4.14. $\mathbf{TID}_{\nu\mathbf{0}}$ *is an extension of* $\mathbf{TID}_{\nu}$ *presented in [17].*

If we choose an appropriate monotone realizer m for $A(X)$ where X is positive in $A(X)$, we get almost the same soundness proof for $\mathbf{TID}_{\nu}$ as in [17]. But the soundness proof of this paper is simpler than that for $\mathbf{TID}_{\nu}$ in [17] because of the monotone condition. The main difference of two proofs is in the proof of the case of the coinduction. This simplicity comes from the fact that we pay attention to the realizer m of the monotonicity (MONO) in this paper, while in [17] we calculate concretely $\sigma^{X,f}_{A(X)}$, which corresponds to the realizer $mf\mathbf{x}$.

5 Theory $\mathbf{TID}_{\nu\mathbf{2}}$ and Its q-Realizability

In this section, we present another **q**-realizability interpretation, which is based on second order logic [4]. It interprets a theory $\mathbf{TID}_{\nu\mathbf{2}}$, which has least-upper-bound coinductive definitions, in a second order theory $\mathbf{T_2}$. It gives also a sound interpretation for $\mathbf{TID}_{\nu\mathbf{1}}$. The new point is to give least-upper-bound coinductive definitions, which cover monotone coinductive definitions and have natural meaning given by semantics with the second order coding.

Definition 5.1 (Theory $\mathbf{TID}_{\nu\mathbf{2}}$). The language of the theory $\mathbf{TID}_{\nu\mathbf{2}}$ is the same as that of $\mathbf{TID}^{-}$.

The axioms and inference rules of $\mathbf{TID}_{\nu\mathbf{2}}$ are those of $\mathbf{TID}^{-}$ and the following inference rule ($\nu 3$) and axiom ($\nu 2$):

$$\frac{\forall\mathbf{x}(X(\mathbf{x})\to A(X))\to\forall\mathbf{x}(X(\mathbf{x})\to C(\mathbf{x}))}{\forall\mathbf{x}(\nu(\mathbf{x})\to C(\mathbf{x}))}\ (\nu 3),$$

$$\forall\mathbf{x}(C(\mathbf{x})\to A(C))\to\forall\mathbf{x}(C(\mathbf{x})\to\nu(\mathbf{x})) \qquad (\nu 2)$$

where $\nu\equiv\nu X.\lambda\mathbf{x}.A(X)$ and in the rule ($\nu 3$) the predicate variable X does not occur freely in formulas $C(\mathbf{x})$ and open assumptions of a deduction of the premise.

The intended meaning of $\nu(\mathbf{x})$ is the least upper bound $\cup\{X|\forall\mathbf{x}(X(\mathbf{x})\to A(X))\}$ with respect to the order relation $\sqsubseteq$ given by $X\sqsubseteq Y\leftrightarrow\forall x(X(x)\to Y(x))$. ($\nu 2$) means that ν is an upper bound and ($\nu 3$) means that ν is the least one.

We call the definitions of $\nu(\mathbf{x})$ by ($\nu 3$) and ($\nu 2$) *least-upper-bound coinductive definitions*. Note that monotonicity of $A(X)$ is not required here. In this sense, least-upper-bound coinductive definitions are generalization of monotone coinductive definitions.

Theorem 5.2. $\mathbf{TID}_{\nu\mathbf{2}}$ *is consistent.*

This theorem is proved in the appendix.

We use a second order theory $\mathbf{T_2}$ to give **q**-realizability for $\mathbf{TID}_{\nu\mathbf{2}}$.

Definition 5.3 (Theory $\mathbf{T_2}$). The theory $\mathbf{T_2}$ is defined as the theory obtained from $\mathbf{TID}_{\nu\mathbf{2}}$ by adding the second order universal quantifiers.

If A is a formula and X is a predicate variable, then $\forall XA$ is a formula.

The axioms and inference rules of $\mathbf{T_2}$ are those of $\mathbf{TID}_{\nu\mathbf{2}}$ and the following inference rules:

$$\frac{A(X)}{\forall XA(X)}\ (\forall^2 I), \qquad \frac{\forall XA(X)}{A(F)}\ (\forall^2 E)$$

where $A(X)$ denotes a formula, F denotes an abstract of the same arity as the predicate variable X, $A(F)$ denotes the formula obtained from $A(X)$ by replacing all the occurrences of X by F, and in the rule $(\forall^2 I)$ X does not occur freely in open assumptions of a deduction of the premise.

Theorem 5.4. $\mathbf{T_2}$ *is consistent.*

This theorem is proved in the appendix.

Proposition 5.5. *If a formula $A(X)$ is monotone with respect to a predicate variable X, that is,*

$$\forall\mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall\mathbf{x}(A(X) \to A(Y))$$

holds in $\mathbf{TID}_{\nu\mathbf{2}}$, then $\mathbf{TID}_{\nu\mathbf{2}}$ proves

$$\forall\mathbf{x}(\nu(\mathbf{x}) \to A(\nu)) \qquad (\nu 1)$$

where $\nu \equiv \nu X.\lambda\mathbf{x}.A(X)$.

This proposition shows that least-upper-bound coinductive definitions cover monotone coinductive definitions.

Proof 5.6. Assume $\forall\mathbf{x}(X(\mathbf{x}) \to A(X))$. By $(\nu 2)$, $\forall\mathbf{x}(X(\mathbf{x}) \to \nu(\mathbf{x}))$ holds. By monotonicity of $A(X)$, $\forall\mathbf{x}(A(X) \to A(\nu))$ holds. Combining it with the assumption, we have $\forall\mathbf{x}(X(\mathbf{x}) \to A(\nu))$. Hence we have $\forall\mathbf{x}(X(\mathbf{x}) \to A(X)) \to \forall\mathbf{x}(X(\mathbf{x}) \to A(\nu))$ without assumptions. By $(\nu 3)$, $\forall\mathbf{x}(\nu(\mathbf{x}) \to A(\nu))$ holds. □

Theorem 5.7. *If a formula A is provable in $\mathbf{TID}_{\nu\mathbf{1}}$, then A is provable in $\mathbf{TID}_{\nu\mathbf{2}}$.*

This claim is proved immediately by Proposition 5.5.

We will explain the principle of our notion of **q**-realizability defined here, before we define **q**-realizability for $\mathbf{TID}_{\nu\mathbf{2}}$ in $\mathbf{T_2}$. We define **q**-realizability of $\nu(t)$ by using the higher order coding and the **q**-realizability for second order logic. The least-upper-bound coinductive definitions give semantically the least upper bound $\cup\{X | X \sqsubseteq A(X)\}$, where $X \sqsubseteq Y$ denotes $\forall x(X(x) \to Y(x))$. Second order logic can code $y \in \cup\{X | P(X)\}$ by $\exists X(P(X) \& X(y))$, since, by letting $\nu(y)$ be $\exists X(P(X)\ \&\ X(y))$, (1) $P(A)$ implies $A \sqsubseteq \nu$, and (2) $\forall X(P(X) \to X \sqsubseteq C)$ implies $\nu \sqsubseteq C$. Combining them, we get the higher order coding of $\nu(y)$ by $\exists X(\forall x(X(x) \to A(x, X)) \& X(y))$. The second order existential quantifier $\exists X$ is coded by $\exists X.A(X) \equiv \forall Y(\forall X(A(X) \to Y) \to Y)$. Hence we get the coding of $\nu(y)$ by

$$\nu(y) \equiv \forall Y(\forall X(\forall x(X(x) \to A(x, X)) \& X(y) \to Y) \to Y).$$

By the realizability for second order logic, we have $(e \quad \mathbf{q} \quad \forall X.A(X)) \equiv \forall X X^*(e \quad \mathbf{q} \quad A(X))$. By using these, we define $(e \quad \mathbf{q} \quad \nu(y))$ by

$$(e \quad \mathbf{q} \quad \nu(y)) \equiv (e \quad \mathbf{q} \quad \forall Y(\forall X(\forall x(X(x) \to A(x,X)) \,\&\, X(y) \to Y) \to Y)).$$

Definition 5.8 (q-Realizability for $\mathbf{TID}_{\nu\mathbf{2}}$).
A $\mathbf{q}$-realizability interpretation $(e \quad \mathbf{q} \quad A)$ of the formula A is defined by induction on the construction of A as follows:

1. $(e \quad \mathbf{q} \quad A) \equiv e = 0 \,\&\, A$ if A is an atomic formula.
2. $(e \quad \mathbf{q} \quad X(\mathbf{t})) \equiv X(\mathbf{t}) \,\&\, X^*(e, \mathbf{t})$ if X is a predicate variable.
3. $(e \quad \mathbf{q} \quad A \,\&\, B) \equiv (e_0 \quad \mathbf{q} \quad A) \,\&\, (e_1 \quad \mathbf{q} \quad B)$.
4. $(e \quad \mathbf{q} \quad A \vee B) \equiv \mathbf{N}(e_0) \,\&\, (e_0 = 0 \to (e_1 \quad \mathbf{q} \quad A)) \,\&\, (e_0 \neq 0 \to (e_1 \quad \mathbf{q} \quad B))$.
5. $(e \quad \mathbf{q} \quad A \to B) \equiv \forall q((q \quad \mathbf{q} \quad A) \to (eq \quad \mathbf{q} \quad B)) \,\&\, (A \to B)$.
6. $(e \quad \mathbf{q} \quad \forall x A(x)) \equiv \forall x(ex \quad \mathbf{q} \quad A(x))$.
7. $(e \quad \mathbf{q} \quad \exists x A(x)) \equiv (e_1 \quad \mathbf{q} \quad A(e_0))$.
8. $(e \quad \mathbf{q} \quad (\lambda \mathbf{x}.A)(\mathbf{t})) \equiv (\lambda r \mathbf{x}.(r \quad \mathbf{q} \quad A))(e, \mathbf{t})$.
9. $(e \quad \mathbf{q} \quad \nu(\mathbf{t})) \equiv \forall Y Y^* r(\forall X X^*(r \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{t}) \to Y) \to Y \,\&\, Y^*(er)) \,\&\, \nu(\mathbf{t})$ where $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$.

Proposition 5.9. *Suppose that $A(X)$ is a formula and $A(C)$ is the formula obtained from $A(X)$ by replacing all the occurrences of a predicate variable X by an abstract C. Then*

$$(e \quad \mathbf{q} \quad A(C)) \leftrightarrow (e \quad \mathbf{q} \quad A(X))[X := C, X^* := (\lambda r \mathbf{x}.(r \quad \mathbf{q} \quad C(\mathbf{x})))]$$

holds.

This claim is proved by induction the construction of $A(X)$.

Proposition 5.10. *$((e \quad \mathbf{q} \quad A) \to A)$ holds in $\mathbf{T_2}$.*

This claim is proved by induction on the construction of A.
We give realizers of $(\nu 3)$ and $(\nu 2)$ in the next proposition.

Proposition 5.11. *Suppose that $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$.*

1. The rule

$$\frac{q \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \to \forall \mathbf{x}(X(\mathbf{x}) \to C(\mathbf{x}))}{\lambda \mathbf{x} r.r(\lambda p.qp_0 \mathbf{x} p_1) \quad \mathbf{q} \quad \forall \mathbf{x}(\nu(\mathbf{x}) \to C(\mathbf{x}))}$$

is derivable in $\mathbf{T_2}$ where X and X^* do not occur freely in open assumptions of a deduction of the premise.

2. $(\lambda q \mathbf{x} r p.p(q,r) \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to A(C)) \to \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})))$ holds in $\mathbf{T_2}$.

This proposition is proved in the appendix.

Theorem 5.12 (Soundness of the Realizability for $\mathbf{TID}_{\nu\mathbf{2}}$). *If a formula A is provable in $\mathbf{TID}_{\nu\mathbf{2}}$, then we can get a term e from the proof and $(e \quad \mathbf{q} \quad A)$ holds in $\mathbf{T_2}$.*

This theorem is proved by induction on the proof of A by using Proposition 5.11.

We give a realizer of $(\nu 1)$ in the next proposition.

Proposition 5.13. *If m is a realizer of monotonicity of $A(X)$, that is,*

$$m \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall \mathbf{x}(A(X) \to A(Y))$$

holds in $\mathbf{T_2}$*, then*

$$\lambda \mathbf{x} r.r(\lambda p.m(\lambda \mathbf{x} r s.s(p_0, r))\mathbf{x}(p_0 \mathbf{x} p_1)) \quad \mathbf{q} \quad \forall \mathbf{x}(\nu(\mathbf{x}) \to A(\nu))$$

holds in $\mathbf{T_2}$ *where* $\nu \equiv \nu X.\lambda \mathbf{x}.A(X)$.

This claim is proved by calculating the realizer according to the proof of Proposition 5.5.

Theorem 5.14 (Soundness of the Realizability for $\mathbf{TID}_{\nu\mathbf{1}}$). *If a formula A is provable in* $\mathbf{TID}_{\nu\mathbf{1}}$*, then we can get a term e from the proof and $(e \quad \mathbf{q} \quad A)$ holds in* $\mathbf{T_2}$.

This theorem is proved by induction on the proof of A by using Theorem 5.12 and Proposition 5.13.

By using the soundness theorem, we can prove the term existence property, the disjunction property, the program extraction theorem and so on.

Theorem 5.15. *(1) If* $\mathbf{TID}_{\nu\mathbf{2}} \vdash \exists x A(x)$*, we can get a term t from the proof and* $\mathbf{T_2} \vdash A(t)$ *holds.*

(2) If $\mathbf{TID}_{\nu\mathbf{2}} \vdash A \vee B$*, then* $\mathbf{T_2} \vdash A$ *or* $\mathbf{T_2} \vdash B$ *holds.*

This theorem is proved in the same way as Theorem 4.11.

6 Application to Program Synthesis

We will explain that $\mathbf{q}$-realizability of coinductive definitions can be used to synthesize programs which treats coinductively defined infinite data structures such as streams. For this purpose, both realizability interpretation for $\mathbf{TID}_{\nu\mathbf{0}}$ and $\mathbf{TID}_{\nu\mathbf{2}}$ can be used.

Theorem 6.1 (Program Extraction). *(1) If there exists a term j such that* $\mathbf{TID}_{\nu\mathbf{1}} \vdash \forall x(A(x) \to (j(x) \quad \mathbf{q} \quad A(x)))$ *and* $\mathbf{TID}_{\nu\mathbf{0}} \vdash \forall x(A(x) \to \exists y B(x, y))$*, we can get a program f from the proof and* $\mathbf{TID}_{\nu\mathbf{1}} \vdash \forall x(A(x) \to B(x, f(x)))$ *holds.*

(2) The same claim holds for $\mathbf{TID}_{\nu\mathbf{2}}$ *and* $\mathbf{T_2}$.

Proof 6.2. We show only the case (1). Suppose that there exists a term j such that $\mathbf{TID}_{\nu\mathbf{1}} \vdash \forall x(A(x) \to (j(x) \quad \mathbf{q} \quad A(x)))$ and $\mathbf{TID}_{\nu\mathbf{0}} \vdash \forall x(A(x) \to \exists y B(x, y))$. By Theorem 4.9, we can get a term e from the proof and $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad \forall x(A(x) \to \exists y B(x, y)))$. By Definition 4.1 and Proposition 4.3, it implies that $\mathbf{TID}_{\nu\mathbf{1}} \vdash \forall x(A(x) \to B(x, (ex(jx))_0))$. Put $f := \lambda x.(ex(jx))_0$, then the claim holds. □

The next theorem holds for $\mathbf{TID}_{\nu\mathbf{1}}$. By the theorem, we can synthesize programs which treat streams. This theorem improves the stream program extraction theorem given in [17] by weakening required conditions.

Theorem 6.3 (Stream Program Extraction). *Suppose that the specification formula is* $\forall x(A(x) \to \exists y B(x,y))$,

$B \equiv \nu X.\lambda xy.\widetilde{B}(x,y_0) \,\&\, X(t,y_1)$,

for some formula $\widetilde{B}(x,y)$ *and some term* t *in which the variable* y *does not occur freely, and we have a term* j *such that* $\forall x(A(x) \to (jx \;\mathbf{q}\; A(x)))$. *Then we define*

$B^{\Box} \equiv \nu X.\lambda x.\exists z\widetilde{B}(x,z) \,\&\, X(t)$.

If we have e *such that*

$e \;\mathbf{q}\; \forall x(A(x) \to B^{\Box}(x))$,

we can get a term F *such that*

$\forall x(A(x) \to B(x,Fx))$

where

$\mathit{filter} \equiv \mu f.\lambda x.(x_{00}, fx_1)$,

$F \equiv \lambda x.\mathit{filter}(ex(jx))$.

The theorem is proved in Appendix B.

In the specification formula $\forall x(A(x) \to \exists y B(x,y))$, x means the input, $A(x)$ means the input condition, y means the output stream and $B(x,y)$ means the input-output relation. The input-output relation $B(x,y)$ is restricted to the following form: (1) Each element b of the stream y satisfies the condition $\widetilde{B}(x',b)$ where x' is usually a substream obtained from the input stream x. (2) If an element b is decided by $\widetilde{B}(x',b)$, then the next element c of the output stream is decided by $\widetilde{B}(t[x := x'],c)$. The formula $B^{\Box}(x)$ means that for each input x there exists some value z satisfying $\widetilde{B}(x,z)$. The term j means a self-realizer of the formula $A(x)$. A realizer e of $\forall x(A(x) \to B^{\Box}(x))$ is produced from a constructive proof of $\forall x(A(x) \to B^{\Box}(x))$.

We will give examples of program synthesis by using Theorem 6.3.

Example 6.4 (Bit Stream Transformer). We will discuss the program which gets a bit stream and returns a stream in which each element is the alternative of the element of the input stream.

The predicate $BS(x)$ which says that x is a bit stream can be represented in the theory by the facility of coinductive definitions as follows:

$BS \equiv \nu X.\lambda x.x = (x_0,x_1) \,\&\, (x_0 = 0 \vee x_0 = 1) \,\&\, X(x_1)$.

The input condition of the specification is the formula $BS(x)$.

The input-output relation of the specification is the formula $ALT(x,y)$ which is defined as follows:

$ALT \equiv \nu X.\lambda xy.(x_0 = 0 \,\&\, y_0 = 1 \vee x_0 = 1 \,\&\, y_0 = 0) \,\&\, X(x_1,y_1)$.

The specification formula is:

$\forall x(BS(x) \to \exists y ALT(x,y))$.

To produce the program by using Theorem 6.3, we prove the following formula corresponding to the specification:

$$\forall x(BS(x) \to ALT^{\Box}(x)) \tag{1}$$

where

$ALT^{\Box} \equiv \nu X.\lambda x.\exists z(x_0 = 0 \,\&\, z = 1 \vee x_0 = 1 \,\&\, z = 0) \,\&\, X(x_1)$.

We prove (1) in the following way here: First, we prove

$$\forall x(BS(x) \to \exists z(x_0 = 0 \,\&\, z = 1 \vee x_0 = 1 \,\&\, z = 0) \,\&\, BS(x_1)). \tag{2}$$

This is proved by letting z be $\mathbf{d}x_0010$. Secondly, by letting C be BS in $(\nu 2)$ for $ALT^{\Box}$, we have

$$\forall x(BS(x) \to \exists z(x_0 = 0 \,\&\, z = 1 \vee x_0 = 1 \,\&\, z = 0) \,\&\, BS(x_1)) \to \forall x(BS(x) \to ALT^{\Box}(x)). \tag{3}$$

Finally, by (2) and (3), we get (1).

We calculate realizers corresponding to the above proofs as follows: The realizer e_1 corresponding to the proof of (2) is:

$e_1 \equiv \lambda xr.((\mathbf{d}x_0010, 0), r_{11})$,

$e_1 \quad \mathbf{q} \quad \forall x(BS(x) \to \exists z(x_0 = 0 \,\&\, z = 1 \vee x_0 = 1 \,\&\, z = 0) \,\&\, BS(x_1))$.

The realizer e_2 corresponding to the proof of (3) is:

$e_2 \equiv \lambda q.\mu f.\lambda xr.mfx(qxr)$,

$e_2 \quad \mathbf{q} \quad \forall x(BS(x) \to \exists z(x_0 = 0 \,\&\, z = 1 \vee x_0 = 1 \,\&\, z = 0) \,\&\, BS(x_1)) \to \forall x(BS(x) \to ALT^{\Box}(x))$

where

$m \equiv \lambda fxr.(r_0, fx_1r_1)$.

The realizer e corresponding to the proof of (1) is:

$e \equiv e_2e_1$,

$e \quad \mathbf{q} \quad \forall x(BS(x) \to ALT^{\Box}(x))$.

We get

$e = \mu f.\lambda xr.((\mathbf{d}x_0010, 0), fx_1r_{11})$.

The extracted program F is:

$$\begin{aligned} Fx &= \mathit{filter}(ex(jx)) \\ &= \mathit{filter}(fx(\mu s.(0, s))) \end{aligned}$$

where $f \equiv \mu f.\lambda xr.((\mathbf{d}x_0010, 0), fx_1r_{11})$. Then we have $Fx \approx (\mu g.\lambda x.(\mathbf{d}x_0010, gx_1))x$. This is the program we expect.

Example 6.5 (Fibonacci Sequence). We will produce a stream which outputs the Fibonacci sequence. We will synthesize the generalized program $F(a, b)$ which outputs a variant of the Fibonacci sequence which starts at natural numbers a and b instead of starting at 1 and 1.

The input is a pair of natural numbers and the input condition of the specification is the formula $\mathbf{N}(x_0) \,\&\, \mathbf{N}(x_1)$.

The input-output relation of the specification is the formula $\mathit{FIB}(x, y)$ which is defined as follows:

$\mathit{FIB} \equiv \nu X.\lambda xy.y_0 = x_0 \,\&\, X((x_1, x_0 + x_1), y_1)$.

The specification formula is:

$\forall x(\mathbf{N}(x_0) \,\&\, \mathbf{N}(x_1) \to \exists y \mathit{FIB}(x, y))$.

To produce the program by using Theorem 6.3, we prove the following formula corresponding to the specification:

$$\forall x(\mathbf{N}(x_0) \,\&\, \mathbf{N}(x_1) \to \mathit{FIB}^{\Box}(x)) \tag{4}$$

where

$FIB^{\Box} \equiv \nu X.\lambda x.\exists z(z = x_0)\ \&\ X((x_1, x_0 + x_1))$.

We prove (4) in the following way here: First, we prove

$$\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to \exists z(z = x_0)\ \&\ (\mathbf{N}(x_1)\ \&\ \mathbf{N}(x_0 + x_1))). \tag{5}$$

This is proved by letting z be x_0. Secondly, by letting C be $\lambda x.\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1)$ in $(\nu 2)$ for $FIB^{\Box}$, we have

$$\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to \exists z(z = x_0)\ \&\ (\mathbf{N}(x_1)\ \&\ \mathbf{N}(x_0 + x_1))) \to$$
$$\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to FIB^{\Box}(x)). \tag{6}$$

Finally, by (5) and (6), we get (4).

We calculate realizers corresponding to the above proofs as follows: The realizer e_1 corresponding to the proof of (5) is:

$e_1 \equiv \lambda xr.((x_0, 0), 0)$,

e_1 **q** $\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to \exists z(z = x_0)\ \&\ (\mathbf{N}(x_1)\ \&\ \mathbf{N}(x_0 + x_1)))$.

The realizer e_2 corresponding to the proof of (6) is:

$e_2 \equiv \lambda q.\mu f.\lambda xr.mfx(qxr)$,

e_2 **q** $\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to \exists z(z = x_0)\ \&\ (\mathbf{N}(x_1)\ \&\ \mathbf{N}(x_0 + x_1))) \to$
$\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to FIB^{\Box}(x))$

where

$m \equiv \lambda fxr.(r_0, f(x_1, x_0 + x_1)r_1)$.

The realizer e corresponding to the proof of (4) is:

$e \equiv e_2 e_1$,

e **q** $\forall x(\mathbf{N}(x_0)\ \&\ \mathbf{N}(x_1) \to FIB^{\Box}(x))$.

We get

$e = \mu f.\lambda xr.((x_0, 0), f(x_1, x_0 + x_1)0)$.

The extracted program F is:

$$\begin{aligned} Fx &= \mathit{filter}(ex(jx)) \\ &= \mathit{filter}(fx(0,0)) \end{aligned}$$

where $f \equiv \mu f.\lambda xr.((x_0, 0), f(x_1, x_0 + x_1)0)$.

Then we have the program $F(1,1)$ as the Fibonacci sequence stream. This is the program we expect, since $F(1,1) \approx (\mu g.\lambda xy.(x, gy(x + y)))11$ holds.

Monotone coinductive definitions are better fit to module programming than positive coinductive definitions. We will explain the situation by example. Suppose that a base theory T can not prove that a formula $A(X)$ is transformed into a positive formula and that $A(X)$ is monotone. Suppose that by using an additional theory module E, $T + E$ proves that $A(X)$ is monotone and transformed into a positive formula. Suppose we can use only T to prove the specification and after that, E is given. In this setting, we can produce a program module $f(r)$ with a program parameter r by using T and when E and a program e are given later, $T + E$ proves that $f(e)$ is a correct program. We will compare monotone coinductive definitions with positive coinductive definitions in the setting.

(I) The monotone case. The program is produced in the following steps: (1) We make a constructive proof in $T + (\mathrm{MONO}) + (\mathrm{MONO\text{-}Q}) + (r \ \mathbf{q} \ (\mathrm{MONO}))$. (2) We produce a program module $f(r)$ with a program parameter r. (3) When E is given later, we prove (MONO) and (MONO-Q) in $T+E$ and find a program m such that $T+E$ proves $(m \ \mathbf{q} \ (\mathrm{MONO}))$. (4) We get a correct program $f(m)$.

(II) The positive case. The program is produced in the following steps: (1) We make a constructive proof in $T + (\nu 1) + (\nu 2) + (r_1 \ \mathbf{q} \ (\nu 1)) + (r_2 \ \mathbf{q} \ (\nu 2))$. (2) We produce a program module $g(r_1, r_2)$ with program parameters r_1 and r_2. (3) When E is given later, we find a positive formula $A_2(X)$ such that $T + E$ proves $A(X) \leftrightarrow A_2(X)$, and programs i and j such that $T + E$ proves $(i \ \mathbf{q} \ A(X) \to A_2(X))$ and $(j \ \mathbf{q} \ A_2(X) \to A(X))$. The form of $A_2(X)$ gives a program e such that $T + E$ proves $(e \ \mathbf{q} \ (\nu 2))$ for $\nu X.A_2(X)$. (4) We get a correct program $g(i, \lambda q.e(\lambda x.q(jx)))$.

Comparing two cases, we can say monotone coinductive definitions are better fit to module programming for the following reasons: (1) The monotone case is simpler. (2) In the positive case, we need to find $A_2(X)$. (3) In the positive case, before E is given, we cannot get a realizer e of coinduction, which is an important part of the program module since it may give a loop structure to the program.

References

1. M. Beeson, *Foundations of Constructive Mathematics* (Springer, 1985).
2. R.L. Constable et al, *Implementing Mathematics with the Nuprl Proof Development System* (Prentice-Hall, 1986).
3. T. Coquand, Infinite Objects in Type Theory, *LNCS* **806** (Springer, 1993) 62–78.
4. H. Friedman, On the derivability of instantiation properties, *Journal of Symbolic Logic* **42** (4) (1977) 506–514.
5. S. Hayashi and H. Nakano, **PX***: A Computational Logic* (MIT Press, 1988).
6. S. Kobayashi and M. Tatsuta, Realizability interpretation of generalized inductive definitions, *Theoretical Computer Science* **131** (1994) 121–138.
7. P. Martin-Löf, Constructive mathematics and computer programming, In: *Logic, Methodology, and Philosophy of Science VI*, eds. L.J. Cohen et. al (North-Holland, Amsterdam, 1982) 153–179.
8. N. P. Mendler, Inductive types and type constraints in the second-order lambda calculus, *Annals of Pure and Applied Logic* **51** (1991) 159–172.
9. R. Milner, *Communication and Concurrency* (Prentice Hall, 1989).
10. C. Paulin-Mohring, Extracting F_ω's programs from proofs in the Calculus of Constructions, *Proc.* 16^{th} *Symp. Principles of Programming Languages* (1989) 89–104.
11. F. Leclerc and C. Paulin-Mohring, Programming with Streams in Coq, A case study: the Sieve of Eratosthenes, *LNCS* **806** (Springer, 1993) 191–212.
12. L. C. Paulson, Mechanizing Coinduction and Corecursion in Higher-order Logic, *Journal of Logic and Computation* **7** (2) (1997) 175–204.
13. M. Parigot, Recursive programming with proofs, *Theoretical Computer Science* **94** (1992) 335–356.
14. C. Raffalli, Data types, infinity and equality in system AF_2, *LNCS* **832** (Springer, 1993) 280–294.

15. C. Talcott, A theory for program and data type specification, *Theoretical Computer Science* **104** (1992) 129–159.
16. M. Tatsuta, Program Synthesis Using Realizability, *Theoretical Computer Science* **90** (1991) 309–353.
17. M. Tatsuta, Realizability interpretation of coinductive definitions and program synthesis with streams, *Theoretical Computer Science* **122** (1994) 119–136.
18. M. Tatsuta, Monotone Recursive Definition of Predicates and Its Realizability Interpretation, *Proceedings of International Conference on Theoretical Aspects of Computer Software*, LNCS **526** (1991) 38–52.
19. M. Tatsuta, Two realizability interpretations of monotone inductive definitions, *International Journal of Foundations of Computer Science* **5** (1) (1994) 1–21.
20. M. Tatsuta, Realizability for Constructive Theory of Functions and Classes and Its Application to Program Synthesis, *Proceedings of Thirteenth Annual IEEE Symposium on Logic in Computer Science* (1998).

Appendix

A Proofs

Proof A.1 (of Theorem 3.1). The claim is proved by induction on the proof of A. Cases are classified by the last rule applied to the proof. We give only two cases here, since other cases are the same as first order logic.

Case ($\nu R1$). We must show $\rho \models \forall \mathbf{x}(\nu(\mathbf{x}) \to A(\nu))$. By induction hypothesis, we have $\rho \models \forall \mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall \mathbf{x}(A(X) \to A(Y))$. Define $F(V) = \{\mathbf{x} \in U^n \mid \rho[X := V] \models A(X)\}$. Then F is monotone. Fix $\mathbf{t} \in U^n$ and assume $\rho \models \nu(\mathbf{t})$. Then we have $\mathbf{t} \in \mathrm{gfp}(F)$. Hence $\mathbf{t} \in F(\mathrm{gfp}(F))$. Therefore $\rho[X := \mathrm{gfp}(F)] \models A(X)[\mathbf{x} := \mathbf{t}]$. Then we have $\rho \models A(\nu)$. This shows the claim for this case.

Case ($\nu R2$). We must show $\rho \models \forall \mathbf{x}(C(\mathbf{x}) \to A(C)) \to \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x}))$. By induction hypothesis, we have $\rho \models \forall \mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall \mathbf{x}(A(X) \to A(Y))$. Define $F(V) = \{\mathbf{x} \in U^n \mid \rho[X := V] \models A(X)\}$. Then F is monotone. Assume $\rho \models \forall \mathbf{x}(C(\mathbf{x}) \to A(C))$.

Assume $\mathbf{t} \in \{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\}$. Then $\rho \models C(\mathbf{t})$. Hence $\rho \models A(C)[\mathbf{x} := \mathbf{t}]$. Therefore $\mathbf{t} \in F(\{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\})$. Hence we have $\{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\} \subset F(\{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\})$ and by the definition of gfp, we get $\{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\} \subset \mathrm{gfp}(F)$.

We will show $\rho \models \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x}))$. Fix $\mathbf{t} \in U^n$ and assume $\rho \models C(\mathbf{t})$. Then $\mathbf{t} \in \{\mathbf{x} \in U^n \mid \rho \models C(\mathbf{x})\}$. Therefore $\mathbf{t} \in \mathrm{gfp}(F)$. That is, $\rho \models \nu(\mathbf{t})$ holds. This shows the claim of this case. □

Proof A.2 (of Theorem 4.4). Let X,Y be predicate variables of arity 0 and

$\nu \equiv \nu X.(X \to X)\ \&\ Y$,

$\nu^* \equiv \nu X^*.\lambda r.(r \quad \mathbf{q}_X[\nu, X^*] \quad (X \to X)\ \&\ Y)$,

$A \equiv Y \to \nu$.

Then clearly $\mathbf{TID}_{\nu\mathbf{1}} \vdash A$ holds. But $\mathbf{TID}_{\nu\mathbf{1}} \vdash (e \quad \mathbf{q} \quad A)$ does not hold for any e. We will show it by the model given in Section 3. By Definition 4.1 we have

$$(e \quad \mathbf{q} \quad A) \equiv (e \quad \mathbf{q} \quad Y \to \nu)$$

$$\equiv (Y \to \nu) \,\&\, \forall r((r \quad \mathbf{q} \quad Y) \to (er \quad \mathbf{q} \quad \nu))$$
$$\equiv (Y \to \nu) \,\&\, \forall r(Y \,\&\, Y^*(r) \to \nu \,\&\, \nu^*(er)),$$

where

$$\begin{aligned}
\nu^* &\equiv \nu X^*.\lambda r.(r \quad \mathbf{q}_X[\nu, X^*] \quad (X \to X) \,\&\, Y) \\
&\equiv \nu X^*.\lambda r.(r_0 \quad \mathbf{q}_X[\nu, X^*] \quad X \to X) \,\&\, (r_1 \quad \mathbf{q}_X[\nu, X^*] \quad Y) \\
&\equiv \nu X^*.\lambda r.((\nu \to \nu) \,\&\, \forall q((q \quad \mathbf{q}_X[\nu, X^*] \quad X) \to (r_0 q \quad \mathbf{q}_X[\nu, X^*] \quad X))) \\
&\qquad \,\&\, Y \,\&\, Y^*(r_1) \\
&\equiv \nu X^*.\lambda r.((\nu \to \nu) \,\&\, \forall q(\nu \,\&\, X^*(q) \to \nu \,\&\, X^*(r_0 q))) \,\&\, Y \,\&\, Y^*(r_1).
\end{aligned}$$

Choose ρ such that $\rho \models Y$ and $\rho[Y^*] = U^n$. Define F as follows:

$F : p(U) \to p(U)$,

$F(V) = \{r \in U \mid \rho[X^* := V] \models (\nu \to \nu) \,\&\,$
$\forall q(\nu \,\&\, X^*(q) \to \nu \,\&\, X^*(r_0 q)) \,\&\, Y \,\&\, Y^*(r_1)\}$.

Since F is not monotone, immediately by the definition of the interpretation of ν^*, $\rho[r := 0] \models \nu^*(er)$ does not hold for any e. Then we have

$\rho[r := 0] \models Y$,

$\rho[r := 0] \models Y^*(r)$,

$\rho[r := 0] \not\models \nu^*(er)$

for any e. Therefore we have

$\rho[r := 0] \not\models Y \,\&\, Y^*(r) \to \nu \,\&\, \nu^*(er)$.

Hence

$\rho \not\models \forall r(Y \,\&\, Y^*(r) \to \nu \,\&\, \nu^*(er))$.

Therefore $\mathbf{TID}_{\nu\mathbf{1}} \not\vdash (e \quad \mathbf{q} \quad A)$ for any e. □

Lemma A.3. *Suppose that* (MONO-Q) *holds for a formula* $A(X)$. *Then* $(e \quad \mathbf{q}_X[F, G] \quad A(X)) \to (e \quad \mathbf{q}_X[F, \lambda r\mathbf{x}.F(\mathbf{x}) \,\&\, G(r, \mathbf{x})] \quad A(X))$ *holds.*

This lemma is proved by induction on the construction of the formula $A(X)$.

Proof A.4 (of Proposition 4.6). By definition, we have $(e \quad \mathbf{q} \quad \nu(\mathbf{x})) \equiv \nu(\mathbf{x}) \,\&\, \nu^*(e, \mathbf{x})$ where $\nu^* \equiv \nu X^*.\lambda r\mathbf{x}.(r \quad \mathbf{q}_X[\nu, X^*] \quad A(X))$. By the assumption, (MONO-Q) holds for $A(X)$ with respect to X. Hence (MONO) holds for $(r \quad \mathbf{q}_X[\nu, X^*] \quad A(X))$ with respect to X^*. By Proposition 4.5, $(e \quad \mathbf{q} \quad A(\nu)) \leftrightarrow (e \quad \mathbf{q}_X[\nu, \lambda r\mathbf{x}.\nu(\mathbf{x}) \,\&\, \nu^*(r, \mathbf{x})] \quad A(X))$. By Lemma A.3, it is equivalent to $(e \quad \mathbf{q}_X[\nu, \nu^*] \quad A(X))$. It is equivalent to $\nu^*(e, \mathbf{x})$ by $(\nu 1')$ in Section 2.4, and implies $A(\nu)$ by Proposition 4.3. By $(\nu 1')$ and (MONO) for $A(X)$, we have $A(\nu) \leftrightarrow \nu(\mathbf{x})$. Hence $(e \quad \mathbf{q} \quad A(\nu))$ is equivalent to $(e \quad \mathbf{q} \quad \nu(\mathbf{x}))$. □

Proof A.5 (of Proposition 4.8). We assume that

$$q \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to A(C)). \tag{7}$$

Let

$f \equiv \mu f.\lambda \mathbf{x}r.mf\mathbf{x}(q\mathbf{x}r)$.

We will show

$$f \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})) \tag{8}$$

under these conditions. This is equivalent to

$\forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})) \,\&$

$\forall \mathbf{x} r((r \quad \mathbf{q} \quad C(\mathbf{x})) \to \nu(\mathbf{x}) \,\&\, (\nu X^*.\lambda r \mathbf{x}.(r \quad \mathbf{q}_X[\nu, X^*] \quad A(X)))(f\mathbf{x}r, \mathbf{x}))$.

We write $(r \quad \mathbf{q} \quad C(\mathbf{x}))$ as $C^*(r, \mathbf{x})$.

The first conjunct is proved by (7), Proposition 4.3 and $(\nu 2)$. We will show the second conjunct:

$\forall \mathbf{x} r(C^*(r, \mathbf{x}) \to (\nu X^*.\lambda r \mathbf{x}.(r \quad \mathbf{q}_X[\nu, X^*] \quad A(X)))(f\mathbf{x}r, \mathbf{x}))$.

By using the rules of NJ, this is equivalent to

$\forall \mathbf{x} s(\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r) \to (\nu X^*.\lambda r \mathbf{x}.(r \quad \mathbf{q}_X[\nu, X^*] \quad A(X)))(s, \mathbf{x}))$.

By $(\nu 2)$, it is sufficient to show

$\forall \mathbf{x} s(\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r) \to$

$\quad (s \quad \mathbf{q}_X[\nu, \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X)))$.

By using the rules of NJ, this is equivalent to

$\forall \mathbf{x} r(C^*(r, \mathbf{x}) \to (f\mathbf{x}r \quad \mathbf{q}_X[\nu, \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X)))$.

We will show this. Fix $\mathbf{x}$ and r and assume

$$C^*(r, \mathbf{x}). \tag{9}$$

We will show

$$(f\mathbf{x}r \quad \mathbf{q}_X[\nu, \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X)). \tag{10}$$

By (7),

$\forall \mathbf{x} r(C^*(r, \mathbf{x}) \to (q\mathbf{x}r \quad \mathbf{q} \quad A(C)))$

holds. Therefore by (9),

$$q\mathbf{x}r \quad \mathbf{q} \quad A(C) \tag{11}$$

holds.

Suppose that

$m \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to Y(\mathbf{x})) \to \forall \mathbf{x}(A(X) \to A(Y))$.

By expanding it, we get

$\forall f(\forall \mathbf{x}((X(\mathbf{x}) \to Y(\mathbf{x})) \,\&\, \forall r(X(\mathbf{x}) \,\&\, X^*(r, \mathbf{x}) \to Y(\mathbf{x}) \,\&\, Y^*(f\mathbf{x}r, \mathbf{x})))$

$\to \forall \mathbf{x} r((r \quad \mathbf{q}_X[X, X^*] \quad A(X)) \to (mf\mathbf{x}r \quad \mathbf{q}_X[Y, Y^*] \quad A(X))))$.

By putting

$X := C$,

$X^* := C^*$,

$Y := \nu$,

$Y^* := \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)$,

$f := f$

in this formula and using (7),Proposition 4.3, $(\nu 2)$ and Proposition 4.5, we get

$\forall \mathbf{x} r((r \quad \mathbf{q} \quad A(C)) \to (mf\mathbf{x}r \quad \mathbf{q}_X[\nu, \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X)))$.

Therefore by putting $\mathbf{x} := \mathbf{x}, r := q\mathbf{x}r$ in this formula and using (11), we get

$(mf\mathbf{x}(q\mathbf{x}r) \quad \mathbf{q}_X[\nu, \lambda s \mathbf{x}.\exists r(C^*(r, \mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X))$.

Therefore, by the definition of f, we have

$$f\mathbf{x}r \quad \mathbf{q}_X[\nu, \lambda s\mathbf{x}.\exists r(C^*(r,\mathbf{x}) \,\&\, s = f\mathbf{x}r)] \quad A(X).$$

Then (10) holds. Therefore we get (8). □

Proof A.6 (of Theorem 5.2). We will construct a model of $\mathbf{TID}_{\nu\mathbf{2}}$ extending an arbitrary model of $\mathbf{TID}^-$. We denote the universe by U.

We will define $\rho \models A$ in almost the same way as for first order logic where A is a formula and ρ is an environment which assigns an element of U to a first order variable and a subset of U^n to a predicate variable of arity n. We present only the definition for the case $\nu(\mathbf{t})$ where $\nu \equiv \nu X.\lambda\mathbf{x}.A(X)$.

We define $\rho \models \nu(\mathbf{t})$ as

$$\mathbf{t} \in \cup\{S \in p(U^n) | \rho[X := S] \models \forall\mathbf{x}(X(\mathbf{x}) \to A(X))\}$$

where $\rho[X := S]$ is defined as follows:

$$\rho[X := S](X) = S;$$

$$\rho[X := S](x) = \rho(x), \qquad \text{if } x \text{ is not } X.$$

Then ($\nu 3$) and ($\nu 2$) hold in this model. □

Proof A.7 (of Theorem 5.4). A theory $\mathbf{T_2}'$ is defined as the theory obtained from $\mathbf{T_2}$ by removing the formula constructor ν.

$\mathbf{T_2}'$ is clearly consistent by consistency of second order logic and the models of $\mathbf{TID}^-$.

It is sufficient to give an interpretation for $\mathbf{T_2}$ in $\mathbf{T_2}'$. An interpretation A° of the formula A is defined by induction on the construction of A as follows:

1. if A is either an atomic formula or $X(\mathbf{t})$, then $A^\circ \equiv A$;
2. $(A \,\&\, B)^\circ \equiv A^\circ \,\&\, B^\circ$;
3. $(A \to B)^\circ \equiv A^\circ \to B^\circ$;
4. $(A \vee B)^\circ \equiv A^\circ \vee B^\circ$;
5. $(\forall x A)^\circ \equiv \forall x(A^\circ)$;
6. $(\exists x A)^\circ \equiv \exists x(A^\circ)$;
7. $((\lambda\mathbf{x}.B)(\mathbf{t}))^\circ \equiv (\lambda\mathbf{x}.B^\circ)(\mathbf{t})$;
8. $((\nu X.\mathbf{x}.A(X))(\mathbf{t}))^\circ \equiv \forall Y(\forall X(\forall\mathbf{x}(X(\mathbf{x}) \to A(X)^\circ) \,\&\, X(\mathbf{t}) \to Y) \to Y)$;
9. $(\forall X A)^\circ \equiv \forall X(A^\circ)$.

For an abstract C, C° is defined by $C^\circ(\mathbf{x}) \equiv (C(\mathbf{x}))^\circ$. For a formula $A(X)$ displaying all the free occurrences of a predicate variable X, $A^\circ(X)$ is defined by $(A(X))^\circ$. Then $(A(C))^\circ \equiv A^\circ(C^\circ)$ holds. This is proved by induction on the construction of A.

If a formula A is proved under assumptions $A_1, \ldots, A_n$ in $\mathbf{T_2}$, then the formula A° is proved under assumptions $A_1^\circ, \ldots, A_n^\circ$ in $\mathbf{T_2}'$. This claim is proved by induction on a proof of a formula A. Cases are classified by the last rule applied to the proof. We show only the case ($\nu 3$) and the case ($\nu 2$) here.

Case ($\nu 3$). Assume

$$\forall\mathbf{x}(X(\mathbf{x}) \to A^\circ(X)) \to \forall\mathbf{x}(X(\mathbf{x}) \to C^\circ(\mathbf{x})). \tag{12}$$

We will show $\forall \mathbf{x}(\nu^\circ(\mathbf{x}) \to C^\circ(\mathbf{x}))$ under these conditions. Fix $\mathbf{x}$ and assume

$$\nu^\circ(\mathbf{x}). \tag{13}$$

We will show $C^\circ(\mathbf{x})$ under these conditions. By putting $Y := C^\circ(\mathbf{x})$ in (13), it is sufficient to show

$\forall X(\forall \mathbf{x}(X(\mathbf{x}) \to A^\circ(X)) \,\&\, X(\mathbf{x}) \to C^\circ(\mathbf{x}))$.

We will show it. Fix X and assume

$$\forall \mathbf{x}(X(\mathbf{x}) \to A^\circ(X)) \,\&\, X(\mathbf{x}). \tag{14}$$

We will show $C^\circ(\mathbf{x})$ under these conditions. By (12) and (14), we get $C^\circ(\mathbf{x})$. Then this case is proved.

Case ($\nu 2$). Assume

$$\forall \mathbf{x}(C^\circ(\mathbf{x}) \to A^\circ(C^\circ)). \tag{15}$$

We will show $\forall \mathbf{x}(C^\circ(\mathbf{x}) \to \nu^\circ(\mathbf{x}))$ under these conditions. Fix $\mathbf{x}$ and assume

$$C^\circ(\mathbf{x}). \tag{16}$$

We will show $\nu^\circ(\mathbf{x})$ under these conditions. Fix Y and assume

$$\forall X(\forall \mathbf{x}(X(\mathbf{x}) \to A^\circ(X)) \,\&\, X(\mathbf{x}) \to Y). \tag{17}$$

We will show Y under these conditions. By putting $X := C^\circ$ in (17) and using (15) and (16), we get Y. Then this case is proved. □

Proof A.8 (of Proposition 5.11).

1. Assume

$$q \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \to \forall \mathbf{x}(X(\mathbf{x}) \to C(\mathbf{x})). \tag{18}$$

We will show

$$\forall \mathbf{x} r((r \quad \mathbf{q} \quad \nu(\mathbf{x})) \to (r(\lambda p.qp_0\mathbf{x}p_1) \quad \mathbf{q} \quad C(\mathbf{x}))) \,\& \tag{19}$$

$$\forall \mathbf{x}(\nu(\mathbf{x}) \to C(\mathbf{x})). \tag{20}$$

(20) is proved from (18), Proposition 5.10 and ($\nu 3$).

We will show (19). Fix $\mathbf{x}$ and r and assume

$(r \quad \mathbf{q} \quad \nu(\mathbf{x}))$.

By the definition of $\mathbf{q}$-realizability, it implies

$\forall Y Y^* s(\forall X X^*(s \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x}) \to Y) \to Y \,\&\, Y^*(rs))$.

Putting

$Y := C(\mathbf{x})$,

$Y^* := \lambda r.(r \quad \mathbf{q} \quad C(\mathbf{x}))$,

$s := \lambda p.qp_0\mathbf{x}p_1$

in this formula, we get

$$\forall X X^*(\lambda p.qp_0\mathbf{x}p_1 \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x}) \to C(\mathbf{x})) \to$$

$$(r(\lambda p.qp_0\mathbf{x}p_1) \quad \mathbf{q} \quad C(\mathbf{x})). \tag{21}$$

Under these assumptions, we will show

$(r(\lambda p.qp_0\mathbf{x}p_1) \quad \mathbf{q} \quad C(\mathbf{x}))$.

By (21), it is sufficient to show

$\forall X X^*(\lambda p.qp_0\mathbf{x}p_1 \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x}) \to C(\mathbf{x}))$.

We will show it. Fix X and X^*. By the definition of $\mathbf{q}$-realizability, it is sufficient to show

$$\forall p((p \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x})) \to (qp_0\mathbf{x}p_1 \quad \mathbf{q} \quad C(\mathbf{x}))) \,\& \tag{22}$$

$$(\forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x}) \to C(\mathbf{x})). \tag{23}$$

(23) is proved by (18) and Proposition 5.10. We will show (22). Fix p and assume

$(p \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X)) \,\&\, X(\mathbf{x}))$.

Then by (18) we have $(qp_0\mathbf{x}p_1 \quad \mathbf{q} \quad C(\mathbf{x}))$. □

2. We will show

$$\forall q((q \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to A(C))) \to (\lambda \mathbf{x} r p.p(q,r) \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})))) \,\& \tag{24}$$

$$\forall \mathbf{x}(C(\mathbf{x}) \to A(C)) \to \forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})). \tag{25}$$

(25) is proved by ($\nu 2$).

We will show (24). Fix q and assume

$$q \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to A(C)). \tag{26}$$

We will show

$$\forall \mathbf{x} r((r \quad \mathbf{q} \quad C(\mathbf{x})) \to (\lambda p.p(q,r) \quad \mathbf{q} \quad \nu(\mathbf{x}))) \,\& \tag{27}$$

$$\forall \mathbf{x}(C(\mathbf{x}) \to \nu(\mathbf{x})). \tag{28}$$

(28) is proved from (26), Proposition 5.10 and ($\nu 2$).

We will show (27). Fix $\mathbf{x}$ and r and assume

$$(r \quad \mathbf{q} \quad C(\mathbf{x})). \tag{29}$$

We will show $(\lambda p.p(q,r) \quad \mathbf{q} \quad \nu(\mathbf{x}))$. It is sufficient to show

$$\forall Y Y^* s(\forall X X^* p((p_0 \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X))) \,\&\, (p_1 \quad \mathbf{q} \quad X(\mathbf{x})) \to (sp \quad \mathbf{q} \quad Y)) \to (s(q,r) \quad \mathbf{q} \quad Y)) \,\& \tag{30}$$

$\nu(\mathbf{x})$.

The second conjunct is proved by (28), (29) and Proposition 5.10.

We will show (30). Fix Y,Y^* and s and assume that

$$\forall X X^* p((p_0 \quad \mathbf{q} \quad \forall \mathbf{x}(X(\mathbf{x}) \to A(X))) \,\&\, (p_1 \quad \mathbf{q} \quad X(\mathbf{x})) \to (sp \quad \mathbf{q} \quad Y)). \tag{31}$$

We will show $(s(q,r) \quad \mathbf{q} \quad Y)$. Putting $X := C(\mathbf{x}), X^* := \lambda r.(r \quad \mathbf{q} \quad C(\mathbf{x})), p := (q,r)$ in (31), by using Proposition 5.9, we get

$(q \quad \mathbf{q} \quad \forall \mathbf{x}(C(\mathbf{x}) \to A(C))) \,\&\, (r \quad \mathbf{q} \quad C(\mathbf{x})) \to (s(q,r) \quad \mathbf{q} \quad Y)$.

From (26) and (29), we have $(s(q,r) \quad \mathbf{q} \quad Y)$. Hence we get (24). □

B Proof of Stream Program Extraction Theorem

We define

$nthcdr \equiv \mu f.\lambda yx.\mathbf{d}y0x(f(\mathbf{p}_{\mathrm{N}}y)x_1)$,

$s|_n \equiv nthcdr\, n\, s$,

$R \equiv \mu f.\lambda yx.\mathbf{d}y0x(t[x := f(\mathbf{p}_{\mathrm{N}}y)x])$.

Lemma B.1. *$(r \ \mathbf{q} \ B^{\Box}(x)) \to (r|_n \ \mathbf{q} \ B^{\Box}(Rnx))$ holds.*

Proof B.2. Suppose $(r \ \mathbf{q} \ B^{\Box}(x))$. We will show $(r|_n \ \mathbf{q} \ B^{\Box}(Rnx))$ by induction on n.

Case $n = 0$ follows from the assumption.

Case $n+1$. By induction hypothesis, we have $(r|_n \ \mathbf{q} \ B^{\Box}(Rnx))$. By definition, it implies $((r|_n)_1 \ \mathbf{q} \ B^{\Box}(t[x := Rnx]))$. Hence $(r|_{n+1} \ \mathbf{q} \ B^{\Box}(R(n+1)x))$ holds. □

Lemma B.3. *Assume $(r \ \mathbf{q} \ B^{\Box}(a))$. Then $\forall xy(\exists n(x = Rna \ \& \ y = (\mathit{filter}\, r)|_n) \to B(x,y))$ holds.*

Proof B.4. By $(\nu 2)$ for $B(x,y)$, it is sufficient to show

$\forall xy(\exists n(x = Rna \ \& \ y = (\mathit{filter}\, r)|_n) \to$

$\quad \widetilde{B}(x, y_0) \ \& \ \exists l(t = Rla \ \& \ y_1 = (\mathit{filter}\, r)|_l)$.

Fix x and y and suppose $x = Rna \ \& \ y = (\mathit{filter}\, r)|_n$.

From Lemma B.1, $(r|_n \ \mathbf{q} \ B^{\Box}(Rna))$ holds. Therefore $\widetilde{B}(Rna, (r|_n)_{00})$ holds. Since $y_0 = ((\mathit{filter}\, r)|_n)_0 = (r|_n)_{00}$, we have $\widetilde{B}(x, y_0)$.

Let l be $n+1$, then we have $t = Rla \ \& \ y_1 = (\mathit{filter}\, r)|_l$. □

Proposition B.5. *$(r \ \mathbf{q} \ B^{\Box}(x)) \to B(x, \mathit{filter}\, r)$ holds.*

Proof B.6. By putting $r := r, a := x, x := x, y := \mathit{filter}\, r$ and $n := 0$ in Lemma B.3, we have the claim. □

Proof B.7 (of Theorem 6.3). Fix x and suppose $A(x)$. Then $(ex(jx) \ \mathbf{q} \ B^{\Box}(x))$ holds. From Proposition B.5, we have $B(x, \mathit{filter}(ex(jx)))$. Then $B(x, Fx)$ holds. Hence we have $\forall x(A(x) \to B(x, Fx))$. □