Logical Methods in Computer Science
Volume 17, Issue 2, 2021, pp. 7:1–7:27
https://lmcs.episciences.org/

Submitted May 01, 2019
Published Apr. 20, 2021

# LOGIC FOR EXACT REAL ARITHMETIC

HELMUT SCHWICHTENBERG [a] AND FRANZISKUS WIESNET [b]

[a] Mathematisches Institut, LMU, München
*e-mail address*: schwicht@math.lmu.de

[b] Mathematisches Institut, LMU, München and Dipartimento di Matematica, Università degli Studi di Trento
*e-mail address*: franziskus.wiesnet@unitn.it

Abstract. Continuing earlier work of the first author with U. Berger, K. Miyamoto and H. Tsuiki, it is shown how a division algorithm for real numbers given as a stream of signed digits can be extracted from an appropriate formal proof. The property of being a real number represented as a stream is formulated by means of coinductively defined predicates, and formal proofs involve coinduction. The proof assistant Minlog is used to generate the formal proofs and extract their computational content as terms of the underlying theory, a form of type theory for finite or infinite data. Some experiments with running the extracted term are described, after its translation to Haskell.

Real numbers in the exact (as opposed to floating-point) sense can be defined as Cauchy sequences (of rationals, with modulus). However, for computational purposes it is better to see them as coded by "streams" of signed digits $\{1, 0, -1\}$. A variant stream representation is the so-called "binary reflected" or Gray-code [Gia99, Tsu02] explained below. Apart from being practically more useful, the stream view turns real numbers into "infinite data" and hence objects of type level 0. As a consequence the type level of other concepts in constructive analysis [Bis67] is lowered by one, which simplifies matters considerably.

Our overall goal is to obtain formally verified algorithms (given by terms in our language) operating on stream represented real numbers. Given an informal idea of how the algorithm should work, there are two methods how this can be achieved.

(I) Formulate (using corecursion) the algorithm in the term language of a suitable theory, and then formally prove that this term satifies the specification;

(II) Find a formal existence proof $M$ (using coinduction) for the object the algorithm is supposed to return. Then apply a proof theoretic method ("realizability") to extract $M$'s computational content as a term (involving corecursion) in the term language of the underlying theory. The verification is done by a formal soundness proof of the realizability

*Key words and phrases:* signed digit code, real number computation, inductive and coinductive definitions, corecursion, program extraction, realizability.

This project has received funding from the European Union's 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143. The second author is a Marie Skłodowska-Curie fellow of the Istituto Nazionale di Alta Matematica.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-17(2:7)2021

interpretation. The extraction of the computational content and the verification are automatic.

A general advantage of (II) over (I) is that one does not need to begin with a detailed formulation of the algorithm, but instead can stay on a more abstract level when proving the (existential) specification. In mathematics we know how to organize proofs, for instance by splitting them into lemmas or on occasion make use of more abstract concepts. In short, mathematical experience can help to find a well-structured algorithmic solution.

Method (I) was employed in [CG06] using Coq, and method (II) in [Ber09, MS15, BMST16] using Minlog. The present paper continues previous work [MS15, BMST16] by a case study on division. It is shown how a division algorithm for real numbers represented as either streams of signed digits or else in binary reflected form (Gray code) can be extracted from an appropriate formal proof dealing with concrete real numbers (Cauchy sequences of rationals, with moduli); in [MS15, BMST16] real numbers were treated axiomatically. The reason for switching to concrete real numbers is the desire to have reliable programs, which are less studied for real number computation. The method described in this paper achieves reliability by providing an (automatically generated) formal proof that the extracted program meets its specification. These proofs are in a formal theory whose consistency is guaranteed by a concrete model [Sco70, Ers77]. There is no need to rely on an axiom system for real numbers as in [MS15, BMST16].

We will work with constructive existence proofs in the style of [Bis67], but in such a way that we can switch on and off the availability of input data for the constructions implicit in the proof [Ber93]. In the present context this will be applied to real numbers as input data: we do not want to make use of the Cauchy sequence for the constructions to be done, but only the computational content of an appropriate coinductive predicate to which the real number is supposed to belong. We consider division of real numbers as a non-trivial case study; it has been dealt with in [CG06] using method (I). Based on the algorithmic idea in [CG06], we employ method (II) to extract signed digit and Gray code based stream algorithms for division from proofs that the reals are closed under division (under some obvious restrictions). As a benefit from the necessary organization into a sequence of lemmas we obtain a relatively easy analysis of the "look-ahead", i.e., how far we have to look into the argument streams to obtain the $n$-th digit of the result stream.

The paper is organized as follows. Section 1 collects some background material. Section 1.1 describes (mainly by examples) the base type objects of the Scott-Ershov [Sco70, Ers77] model of partial continuous functionals, which is the intended model of the theory TCF we are going to use. It is a form of type theory suitable for reasoning with finite and infinite data. This theory is described in Section 1.2. The notion of computational content of proofs is introduced in Section 1.3. Section 1.4 recalls the constructive theory of real numbers of Bishop [Bis67, Ch.2]. In Section 2 division for stream-represented real numbers is studied, both for signed and binary reflected digit streams. Formalizations of the relevant proofs are described, including the terms (programs) extracted from them. Numerical experiments after translating these terms into Scheme or Haskell programs are discussed. Section 3 recalls the notion of realizability in Section 3.1, sketches a proof of the general soundness theorem in Section 3.2, and its formalization for real division algorithms in Section 3.3. Section 4 concludes.

## 1. Preliminaries

We informally describe a formal theory TCF [SW12, Section 7.1] which will be used to carry out proofs on real numbers and also functions and predicates on real numbers. In contrast to [MS15, BMST16] where an axiomatic approach was used we now work with concrete real numbers: Cauchy sequences of rationals, with moduli. The reason for this switch is that we aim at full reliability of the soundness proofs we will (automatically) generate (see Section 3). The axioms the formal theory is based upon, and hence all propositions proved, are valid in a model and therefore correct.

The base type data of the theory are free algebras given by their constructors, for instance lists of signed digits. Functions ("program constants") are defined by defining equations ("computation rules") whose left hand sides are non-overlapping constructor patterns. Termination is *not* required, which makes it possible the have the corecursion operator (see below) as an ordinary constant in the language. Predicates are least and greatest fixed points of clauses.. Examples are totality and cototality of lists of signed digits. The only axioms are the defining equations of the functions and the introduction and elimination axioms for the (co)inductively defined constants, which state their fixed point properties.

An important predicate is equality. Clearly we have the inductively defined Leibniz equality and the decidable equality for finitary algebras like the natural numbers. Equality on the (concrete) reals is more delicate. It is defined from an inductive $\leq$-predicate which refers to the underlying Cauchy sequences and their moduli. For all functions and predicates on the reals that we consider it is necessary to prove compatibility with real equality. The reason for this extension of the axiomatic treatment is to obtain formal proofs which rest only on the fixed point axioms and in this sense are correct: the propositions proved are valid in the concrete model of TCF.

### 1.1. Objects.

Base type objects of our theory are determined by the constructors of (free) algebras. More precisely, let $\alpha, \beta, \xi$ be type variables.

**Definition.** *Constructor types* $\kappa$ have the form

$$\vec{\alpha} \to (\xi)_{i<n} \to \xi$$

with all type variables $\alpha_i$ distinct from each other and from $\xi$. Iterated arrows are understood as associated to the right. An argument type of a constructor type is called a *parameter* argument type if it is different from $\xi$, and a *recursive* argument type otherwise. A constructor type $\kappa$ is *nullary* if it has no recursive argument types. We call

$$\iota := \mu_\xi \vec{\kappa} \quad (\vec{\kappa} \text{ not empty})$$

an *algebra form* ("form" since type parameters may occur). An algebra form is *non-recursive* (or *explicit*) if it does not have recursive argument types.

**Examples.** We list some algebra forms without parameters, with standard names for the constructors added to each constructor type.

$$\begin{aligned}
&\mathbb{U} := \mu_\xi(\mathrm{Dummy}\colon \xi) && \text{(unit)},\\
&\mathbb{B} := \mu_\xi(\mathtt{tt}\colon \xi, \mathtt{ff}\colon \xi) && \text{(booleans)},\\
&\mathbb{D} := \mu_\xi(\mathrm{SdR}\colon \xi, \mathrm{SdM}\colon \xi, \mathrm{SdL}\colon \xi) && \text{(signed digits, for 1, 0, } -1),\\
&\mathbb{N} := \mu_\xi(0\colon \xi, S\colon \xi \to \xi) && \text{(natural numbers, unary)},\\
&\mathbb{P} := \mu_\xi(1\colon \xi, S_0\colon \xi \to \xi, S_1\colon \xi \to \xi) && \text{(positive numbers, binary)},\\
&\mathbb{Z} := \mu_\xi(0_\mathbb{Z}\colon \xi, \mathrm{IntP}\colon \mathbb{P} \to \xi, \mathrm{IntN}\colon \mathbb{P} \to \xi) && \text{(integers)},\\
&\mathbb{Y} := \mu_\xi(-\colon \xi, \mathrm{Branch}\colon \xi \to \xi \to \xi) && \text{(binary trees)}.
\end{aligned}$$

Algebra forms with type parameters are

$$\begin{aligned}
&\mathbb{I}(\alpha) &&:= \mu_\xi(\mathrm{Id}\colon \alpha \to \xi) && \text{(identity)},\\
&\mathbb{L}(\alpha) &&:= \mu_\xi(\mathrm{Nil}\colon \xi, \mathrm{Cons}\colon \alpha \to \xi \to \xi) && \text{(lists)},\\
&\mathbb{S}(\alpha) &&:= \mu_\xi(\mathrm{SCons}\colon \alpha \to \xi \to \xi) && \text{(streams)},\\
&\alpha \times \beta &&:= \mu_\xi(\mathrm{Pair}\colon \alpha \to \beta \to \xi) && \text{(product)},\\
&\alpha + \beta &&:= \mu_\xi(\mathrm{InL}\colon \alpha \to \xi, \mathrm{InR}\colon \beta \to \xi) && \text{(sum)}.
\end{aligned}$$

The default name for the $i$-th constructor of an algebra form is $\mathrm{C}_i$.

**Definition** (Type)**.**

$$\rho, \sigma, \tau ::= \alpha \mid \iota(\vec{\rho}) \mid \tau \to \sigma,$$

where $\iota$ is an algebra form with $\vec{\alpha}$ its parameter type variables, and $\iota(\vec{\rho})$ the result of substituting the (already generated) types $\vec{\rho}$ for $\vec{\alpha}$. Types of the form $\iota(\vec{\rho})$ are called *algebras*. An algebra is *closed* if it has no type variables. The *level* of a type is defined by

$$\begin{aligned}
\mathrm{lev}(\alpha) &:= 0,\\
\mathrm{lev}(\iota(\vec{\rho})) &:= \max(\mathrm{lev}(\vec{\rho})),\\
\mathrm{lev}(\tau \to \sigma) &:= \max(\mathrm{lev}(\sigma), 1 + \mathrm{lev}(\tau)).
\end{aligned}$$

*Base* types are types of level 0, and a *higher* type has level at least 1.

**Examples.** 1. $\mathbb{L}(\alpha)$, $\mathbb{L}(\mathbb{L}(\alpha))$, $\alpha \times \beta$ are algebras.
2. $\mathbb{L}(\mathbb{L}(\mathbb{N}))$, $\mathbb{N} + \mathbb{B}$, $\mathbb{Q} := \mathbb{Z} \times \mathbb{P}$ are closed base types.
3. $(\mathbb{N} \to \mathbb{Q}) \times (\mathbb{P} \to \mathbb{N})$ is a closed algebra of level 1.

There can be many equivalent ways to define a particular type. For instance, we could take $\mathbb{U} + \mathbb{U}$ to be the type of booleans, $\mathbb{L}(\mathbb{U})$ to be the type of natural numbers, and $\mathbb{L}(\mathbb{B})$ to be the type of positive binary numbers.

We introduce the base type objects by example only, for $\mathbb{N}$, $\mathbb{Y}$, $\mathbb{L}(\mathbb{B})$; cf. [SW12, Section 6.1.4] for the general case. These objects are built from *tokens*, which are type correct constructor expressions possibly containing the special symbol $*$ which carries no information. Examples in the algebra $\mathbb{Y}$ of binary trees are

$$\mathrm{Branch}(-, \mathrm{Branch}(-, -)) \qquad \mathrm{Branch}(-, \mathrm{Branch}(*, -))$$

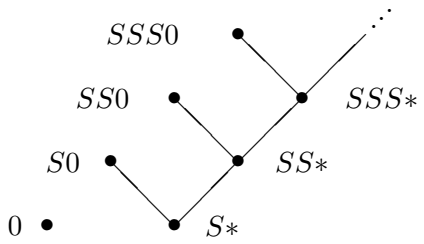
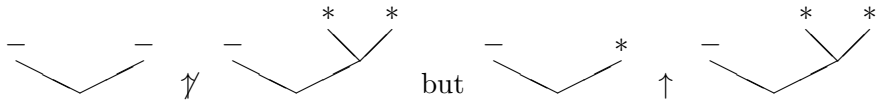


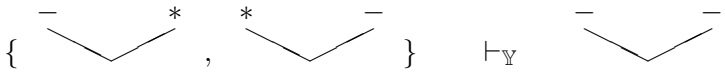

Figure 1: Tokens and entailment for $\mathbb{N}$

or pictorially

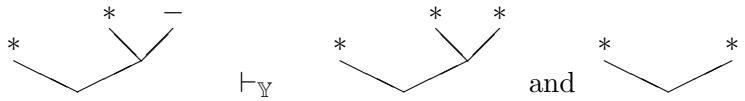
For tokens we have obvious concepts called consistency ($\uparrow$) and entailment ($\vdash$). For instance




Examples for entailment in $\mathbb{Y}$ are



and also

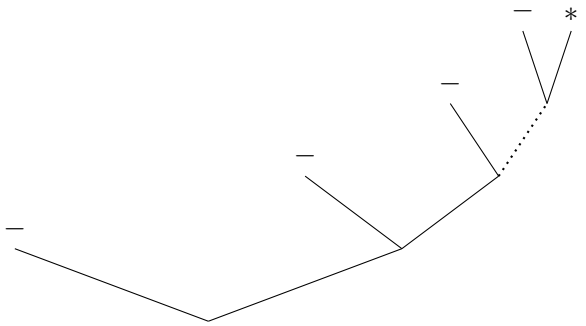



**Definition.** *Objects* of a base type are finite or infinite sets of tokens with are consistent and closed under entailment. An object $x$ is *cototal* if for each of its tokens $P(*)$ with a distinguished occurrence of $*$ there is another token in $x$ where this occurrence of $*$ is replaced by a constructor expression with only $*$'s as arguments. We call $x$ *total* if it is cototal and finite.

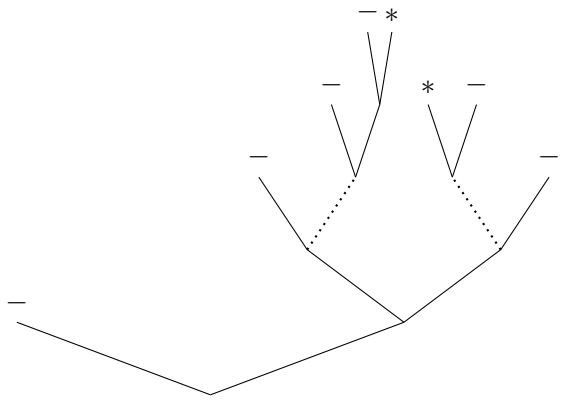
The tokens of $\mathbb{N}$ are shown in Figure 1. For tokens $a, b$ we have $\{a\} \vdash b$ if and only if there is a path from $a$ (up) to $b$ (down). There is exactly one cototal object, consisting of $S*$, $SS*$, $SSS*$ .... Note that $*$ is not a token; the "bottom" object is the empty set.

Examples of more interesting objects in $\mathbb{Y}$ are

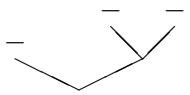
(1) $R :=$ closure (under entailment) of all tokens of the form



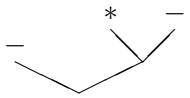
(2) $L$ is defined similarly
(3) $L \cup R$
(4) $\frac{1}{2} :=$ closure of all tokens of the form



(5) Closure of



(6) Closure of



Among these are

| | |
|---|---|
| (1) – (4) | cototal objects, |
| (5) | a total object, |
| (6) | a general object. |

Finally we consider the algebra $\mathbb{L}(\mathbb{B})$ of lists of booleans, to represent finite or infinite paths. Tokens in $\mathbb{L}(\mathbb{B})$ are for instance (writing :: for Cons)

$$\{\mathsf{ff}\} :: \{\mathsf{ff}\} :: \{\mathsf{tt}\} :: \mathrm{Nil} \qquad \text{abbreviated} \qquad RRL[]$$

Consistency in $\mathbb{L}(\mathbb{B})$:

$$RL[] \not\gamma RRL*, \qquad RR* \uparrow RRL*$$

Entailment in $\mathbb{L}(\mathbb{B})$:

$$RRL* \vdash RR*, R* \qquad \text{(and also e.g. } RR\emptyset*\text{)},$$
$$RRL[] \vdash RRL*, RR*, R*$$

Objects in $\mathbb{L}(\mathbb{B})$:

| | |
|---|---|
| total: | Closure of $RLRR[]$ |
| cototal: | Closure of all $RLRR\dots R*$ and all $RRLL\dots L*$ |
| general: | Closure of $RLRR*$ |

Objects of higher type are canonically defined in the Scott-Ershov model [Sco70, Ers77]. A detailed exposition is in [SW12, Chapter 6].

1.2. **Type theory for finite and infinite data.** Terms are built from (typed) variables, constructors and (equationally) defined constants by $\lambda$-abstraction and application, where termination of the defining equations is not required. Defined constants are given by their type and computation rules. Examples for the type $\mathbb{L} := \mathbb{L}(\mathbb{D})$ of lists of signed digits are

Terms are built from (typed) variables, constructors and (equationally) defined constants by $\lambda$-abstraction and application, where termination of the defining equations is not required. Defined constants are given by their type and computation rules. Examples for the type $\mathbb{L} := \mathbb{L}(\mathbb{D})$ of lists of signed digits are

$$\begin{aligned}
&\mathcal{R}^\tau_{\mathbb{L}} && : \mathbb{L} \to \tau \to (\mathbb{D} \to \mathbb{L} \to \tau \to \tau) \to \tau && \text{(recursion operator)},\\
&\mathcal{D}_{\mathbb{L}} && : \mathbb{L} \to \mathbb{U} + (\mathbb{D} \times \mathbb{L}) && \text{(destructor)},\\
&{}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}} && : \tau \to (\tau \to \mathbb{U} + (\mathbb{D} \times (\mathbb{L} + \tau))) \to \mathbb{L} && \text{(corecursion operator)}.
\end{aligned}$$

$\mathcal{R}^\tau_{\mathbb{L}}$ takes a recursion argument $\ell\colon \mathbb{L}$, an initial value $z\colon \tau$ and a "step" argument $f\colon (\mathbb{D} \to \mathbb{L} \to \tau \to \tau)$ operating according to the structure of the given list. The meaning of $\mathcal{R}^\tau_{\mathbb{L}}$ is determined by the computation rules

$$\mathcal{R}^\tau_{\mathbb{L}}([], z, f) = z, \qquad \mathcal{R}^\tau_{\mathbb{L}}(s :: \ell, z, f) = f(s, \ell, \mathcal{R}^\tau_{\mathbb{L}}(\ell, z, f)).$$

$\mathcal{D}_{\mathbb{L}}$ operates by disassembling the given list $\ell\colon \mathbb{L}$ into its head and tail. The computation rules are

$$\mathcal{D}_{\mathbb{L}}([]) = \mathrm{InL}(\mathrm{Dummy}), \qquad \mathcal{D}_{\mathbb{L}}(s :: \ell) = \mathrm{InR}\langle s, \ell\rangle.$$

For ${}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}}$ the "step" $f\colon (\tau \to \mathbb{U} + (\mathbb{D} \times (\mathbb{L} + \tau)))$ operates by inspection (or observation) of its argument $z\colon \tau$. More precisely, the meaning of ${}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}}$ is given by the (non-terminating) computation rule

$${}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}} z f = \begin{cases} [] & \text{if } fz \equiv \mathrm{InL}(\mathrm{Dummy}),\\ s :: u & \text{if } fz \equiv \mathrm{InR}\langle s, \mathrm{InL}(u)\rangle,\\ s :: {}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}} z' f & \text{if } fz \equiv \mathrm{InR}\langle s, \mathrm{InR}(z')\rangle. \end{cases}$$

We use $\equiv$ for Leibniz equality (inductively defined by the clause $\forall_z(z \equiv z)$).

Predicates are either (co)inductively defined or of the form $\{\, \vec{x} \mid A \,\}$ where $A$ is a formula. We identify $\{\, \vec{x} \mid A(\vec{x}\,) \,\}\vec{t}$ with $A(\vec{t}\,)$. Formulas are given as atomic formulas $P\vec{t}$ where $P$ is a predicate and $\vec{t}$ are terms, or else as $A \to B$ or $\forall_x A$ where $A$ and $B$ are already formulas. The formulas $\exists_x A$, $A \wedge B$ and $A \vee B$ technically are inductively defined (nullary) predicates.

An inductively defined predicate $I$ is given by *clauses* of the form

$$I_i^+ \colon \forall_{\vec{x}_i}((A_{ij}(I))_{j<n_i} \to I\vec{t}_i),$$

for $i \in \{0,\dots,k-1\}$, $k>0$ and $n_i \geq 0$, where $I$ only occurs strictly positive in each $A_{ij}(I)$ and $\vec{x}_i$ contains all free variables of $A_{ij}(I)$ and $\vec{t}_i$. The clauses are called introduction axioms of $I$.

An example is the totality predicate $T_{\mathbb{L}}$. Its clauses are

$$(T_{\mathbb{L}})_0^+ \colon T_{\mathbb{L}}[], \qquad (T_{\mathbb{L}})_1^+ \colon \forall_{s,\ell}(T_{\mathbb{D}}s \to T_{\mathbb{L}}\ell \to T_{\mathbb{L}}s :: \ell),$$

where $T_{\mathbb{D}}$ is the totality predicate for the three-element algebra $\mathbb{D}$ of signed digits, defined by the three clauses $T_{\mathbb{D}}s$ for $s$ a signed digit.

Intuitively, an inductively defined predicate is the smallest (w.r.t. $\subseteq$) predicate fulfilling its clauses. This is expressed by its elimination axiom:

$$I^- \colon (\forall_{\vec{x}_i}((A_{ij}(I \cap X))_{j<n_i} \to X\vec{t}_i))_{i<k} \to I \subseteq X,$$

where $X$ can be any predicate with the same arity as $I$. The axiom $I^-$ is also called induction axiom. For example,

$$T_{\mathbb{L}}^- \colon X[] \to \forall_{s,\ell}(T_{\mathbb{D}}s \to T_{\mathbb{L}}\ell \to X\ell \to X(s :: \ell)) \to T_{\mathbb{L}} \subseteq X.$$

Let $I$ be an inductively defined predicate given by its clauses $(I_i^+)_{i<k}$, and assume that there are recursive calls in some clauses. To every such $I$ we associate a "companion" ${}^{\mathrm{co}}I$ and call it a coinductively defined predicate. The elimination (or closure) axiom ${}^{\mathrm{co}}I^-$ of the coinductively defined predicate says that if ${}^{\mathrm{co}}I\vec{x}$ holds, then it comes from one of the clauses:

$${}^{\mathrm{co}}I^- \colon \forall_{\vec{x}}({}^{\mathrm{co}}I\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{j<n_i} A_{ij}({}^{\mathrm{co}}I) \land \vec{x} \equiv \vec{t}_i)).$$

In words: if ${}^{\mathrm{co}}I\vec{x}$ holds, then there is at least one clause $I_i^+$ whose premises $(A_{ij}({}^{\mathrm{co}}I))_{j<n_i}$ (with ${}^{\mathrm{co}}I$ instead of $I$) are fulfilled and whose conclusion is ${}^{\mathrm{co}}I\vec{x}$ up to Leibniz equality. For example, the co-predicate associated with $T_{\mathbb{L}}$ is cototality ${}^{\mathrm{co}}T_{\mathbb{L}}$. Its elimination axiom is

$${}^{\mathrm{co}}T_{\mathbb{L}}^- \colon {}^{\mathrm{co}}T_{\mathbb{L}}\ell \to \ell \equiv [] \lor \exists_{s,\ell'}(T_{\mathbb{D}}s \land {}^{\mathrm{co}}T_{\mathbb{L}}\ell' \land \ell \equiv s :: \ell').$$

The introduction (or coinduction or greatest-fixed-point) axiom ${}^{\mathrm{co}}I^+$ of ${}^{\mathrm{co}}I$ says that ${}^{\mathrm{co}}I$ is the greatest predicate such that

$${}^{\mathrm{co}}I^+ \colon \forall_{\vec{x}}(X\vec{x} \to \bigvee_{i<k} \exists_{\vec{x}_i}(\bigwedge_{j<n_i} A_{ij}({}^{\mathrm{co}}I \cup X) \land \vec{x} \equiv \vec{t}_i)) \to X \subseteq {}^{\mathrm{co}}I.$$

This means that any other "competitor" predicate $X$ satisfying the same closure property is below ${}^{\mathrm{co}}I$. For example,

$${}^{\mathrm{co}}T_{\mathbb{L}}^+ \colon \forall_{\ell}(X\ell \to \ell \equiv [] \lor \exists_{s,\ell'}(T_{\mathbb{D}}s \land ({}^{\mathrm{co}}T_{\mathbb{L}}\ell' \lor X\ell') \land \ell \equiv s :: \ell') \to X \subseteq {}^{\mathrm{co}}T_{\mathbb{L}}.$$

1.3. **Computational content.** We are interested in the computational content of proofs, which in the present setting arises from inductively and coinductively defined predicates only: they can be declared to be computationally relevant (c.r.) or non-computational (n.c.). This also applies to predicate variables $X$. The (finitely many) clauses of a c.r. inductive predicate $I$ determine an algebra $\iota_I$. A "witness" that $I\vec{t}$ or ${}^{\mathrm{co}}I\vec{t}$ holds has type $\iota_I$. It will be a total object in the first and a cototal object in the second case. For instance, the type of the c.r. version of $T_{\mathbb{L}}$ or ${}^{\mathrm{co}}T_{\mathbb{L}}$ is $\mathbb{L}$. A formula $A$ built from atomic formulas by $\to$, $\forall$ is c.r. if its final conclusion is. The *type* $\tau(A)$ is defined by

$$\begin{aligned}\tau(P\vec{t}\,) &:= \tau(P),\\ \tau(A\to B) &:= \begin{cases}\tau(A)\to\tau(B) & \text{if } A \text{ is c.r.}\\ \tau(B) & \text{if } A \text{ is n.c.}\end{cases}\\ \tau(\forall_x A) &:= \tau(A).\end{aligned}$$

Recall that $\exists_x A$, $A\wedge B$ and $A\vee B$ technically are inductively defined (nullary) predicates. If they are c.r., their types are $\tau(\exists_x A) := \tau(A)$; $\tau(A\wedge B) := \tau(A)\times\tau(B), \tau(A), \tau(B)$ depending on the c.r./n.c. status of $A, B$; $\tau(A\vee B) := \tau(A)+\tau(B), \tau(A)+\mathbb{U}, \mathbb{U}+\tau(B), \mathbb{B}$ depending on the c.r./n.c. status of $A, B$.

Let $M$ be a proof in TCF of a c.r. formula $A$. We define its *extracted term* $\mathrm{et}(M)$, of type $\tau(A)$, with the aim to express $M$'s computational content. It will be a term built up from variables, constructors, recursion operators, destructors and corecursion operators by $\lambda$-abstraction and application:

$$\begin{aligned}\mathrm{et}(u^A) &:= z_u^{\tau(A)} \quad (z_u^{\tau(A)} \text{ uniquely associated with } u^A),\\ \mathrm{et}((\lambda_{u^A} M^B)^{A\to B}) &:= \begin{cases}\lambda_{z_u}\mathrm{et}(M) & \text{if } A \text{ is c.r.}\\ \mathrm{et}(M) & \text{if } A \text{ is n.c.}\end{cases}\\ \mathrm{et}((M^{A\to B}N^A)^B) &:= \begin{cases}\mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.}\\ \mathrm{et}(M) & \text{if } A \text{ is n.c.}\end{cases}\\ \mathrm{et}((\lambda_x M^A)^{\forall_x A}) &:= \mathrm{et}(M),\\ \mathrm{et}((M^{\forall_x A(x)}t)^{A(t)}) &:= \mathrm{et}(M).\end{aligned}$$

It remains to define extracted terms for the axioms. Consider a (c.r.) inductively defined predicate $I$. For its introduction and elimination axioms define $\mathrm{et}(I_i^+) := \mathrm{C}_i$ and $\mathrm{et}(I^-) := \mathcal{R}$, where both the constructor $\mathrm{C}_i$ and the recursion operator $\mathcal{R}$ refer to the algebra $\iota_I$ associated with $I$. For the closure and greatest-fixed-point axioms of ${}^{\mathrm{co}}I$ define $\mathrm{et}({}^{\mathrm{co}}I^-) := \mathcal{D}$ and $\mathrm{et}({}^{\mathrm{co}}I^+) := {}^{\mathrm{co}}\mathcal{R}$, where both the destructor $\mathcal{D}$ and the corecursion operator ${}^{\mathrm{co}}\mathcal{R}$ again refer to the algebra $\iota_I$ associated with $I$.

1.4. **Real numbers.** Rational numbers $\mathbb{Q}$ are defined by $\mathbb{Q} := \mathbb{Z}\times\mathbb{P}$, where the pair of $k\colon\mathbb{Z}$ and $p\colon\mathbb{P}$ is denoted by $\frac{k}{p}$. We also have the expected notion of equality between rational numbers. In contrast to the positive, natural, rational numbers or the integers, the real numbers are not an algebra. The property to be a real number is a predicate on $(\mathbb{N}\to\mathbb{Q})\times(\mathbb{P}\to\mathbb{N})$. Totality on this type means that both components map total arguments into total values. By a real number $x$ we mean a pair of this type whose first component is

a Cauchy sequence $(a_n)_{n\in\mathbb{N}}$ of rationals with the second component $M$ as modulus, i.e.,

$$|a_n - a_m| \le \frac{1}{2^p} \quad \text{for } n, m \ge M(p).$$

We write $\mathbb{R}$ for this predicate. For real numbers we can define the arithmetic functions $+$, $-$, $\cdot$ and the inverse and also the absolute value $|\cdot|$. Note that the inverse needs a witness that the real number is positive. We will not go into details here. An exact definition of these functions and also of positivity can be found in [Wie17, Wie18] and in the library of Minlog[1].

An important predicate on the real numbers is real equality: we define $x \le y$ by

$$a_{M(p+1)} \le b_{N(p+1)} + \frac{1}{2^p} \quad \text{for all } p \in \mathbb{P}$$

and real equality $x = y$ by $x \le y \land y \le x$. Most of the predicates and functions on real numbers will be compatible with real equality; however, this must be proved in each case.

The rational numbers can be embedded into the real numbers by identifying a rational number $a$ with the pair consisting of the constant Cauchy sequence $(a)_{n\in\mathbb{N}}$ and the constant modulus $(0)_{p\in\mathbb{P}}$.

We use variable names

| | |
|---|---|
| $x, y$ | for real numbers, |
| $a, b, c$ | for rational numbers, |
| $d, e, k, j, i$ | for integers, |
| $s$ | for signed digits, |
| $u, v$ | for lists (and streams) of signed digits. |

Let $T^{\mathrm{nc}}_{\mathbb{Z}}$ be the n.c. version of the totality predicate for $\mathbb{Z}$, and similarly for other types. Let Sd be the (formally inductive) predicate consisting of the integers $1, 0, -1$, i.e., Sd is given by the three clauses $1 \in \mathrm{Sd}$, $0 \in \mathrm{Sd}$ and $-1 \in \mathrm{Sd}$. We require that Sd has computational content (in the three-element algebra $\mathbb{D}$), and write

$$\forall_{d\in\mathrm{Sd}} A(d) \quad \text{for} \quad \forall_{\hat{d}}(\hat{d} \in T^{\mathrm{nc}}_{\mathbb{Z}} \to \hat{d} \in \mathrm{Sd} \to A(\hat{d}))$$

and similarly for $\exists$. In this case the premise $\hat{d} \in T^{\mathrm{nc}}_{\mathbb{Z}}$ is not necessary because it follows from $\hat{d} \in \mathrm{Sd}$. The predicate $\mathbb{R}$ and also $\le, =$ on real numbers are assumed to be n.c. We write (with $x$ ranging over $[-1, 1]$)

$$\forall_x A(x) \quad \text{for} \quad \forall_{\hat{x}}(\hat{x} \in \mathbb{R} \to A(\hat{x}))$$
$$\forall_{x\in P} A(x) \quad \text{for} \quad \forall_{\hat{x}}(\hat{x} \in \mathbb{R} \to \hat{x} \in P \to A(\hat{x}))$$

and again similarly for $\exists$. The totality of $\hat{x}$ is part of $\hat{x} \in \mathbb{R}$.

We inductively define a predicate $I_0$ on reals $s$ such that the generating sequence $d_0, \dots, d_{m-1}$ for $x \in I_0$ means that $x$ is between $\sum_{n<m} \frac{d_n}{2^{n+1}} - \frac{1}{2^m}$ and $\sum_{n<m} \frac{d_n}{2^{n+1}} + \frac{1}{2^m}$. Let $I_0$ be defined by the two clauses

$$\forall_x(x = 0 \to x \in I_0), \quad \forall_{d\in\mathrm{Sd}}\forall_x\forall_{x'\in I_0}\Big(x = \frac{x' + d}{2} \to x \in I_0\Big).$$

Then the induction axiom is

$$\forall_x(x = 0 \to x \in P) \to \forall_{d\in\mathrm{Sd}}\forall_x\forall_{x'\in I_0\cap P}\Big(x = \frac{x + d}{2} \to x \in P\Big) \to I_0 \subseteq P.$$

[1] `http://minlog-system.de`.

The clauses of $I_0$ also determine the dual predicate ${}^{\mathrm{co}}I_0$, which is coinductively given by the closure axiom

$$\forall_{x\in {}^{\mathrm{co}}I_0}\Big(x=0\vee\exists_{d\in\mathrm{Sd}}\exists_{x'\in {}^{\mathrm{co}}I_0}\Big(x=\frac{x'+d}{2}\Big)\Big)$$

and the coinduction axiom

$$\forall_{x\in P}\Big(x=0\vee\exists_{d\in\mathrm{Sd}}\exists_{x'\in {}^{\mathrm{co}}I_0\cup P}\Big(x=\frac{x'+d}{2}\Big)\Big)\to P\subseteq {}^{\mathrm{co}}I_0.$$

We require that both predicates $I_0$ and ${}^{\mathrm{co}}I_0$ have computational content. The data type associated to these clauses is the algebra $\mathbb{L}$ of lists of signed digits. This association is quite canonical: the first constructor $[]\colon\mathbb{L}$ is a witness for the first clause, which does not need any further information. The second constructor :: of type $\mathbb{D}\to\mathbb{L}\to\mathbb{L}$ is a witness for the second clause consisting of a signed digit and a witness that $x'\in I_0$. We refer to [SW12, Section 7.2] for a formal definition of the type of a (co)inductively defined predicate.

The computational content of the clauses are the constructors, of the induction axiom the recursion operator $\mathcal{R}^\tau_{\mathbb{L}}$, of the closure axiom the destructor $\mathcal{D}_{\mathbb{L}}$ and of the coinduction axiom the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^\tau_{\mathbb{L}}$. All these have been explained in Section 1.2.

## 2. Division for stream-represented real numbers

2.1. **Division for signed digit streams.** From a computational point of view, a real number $x$ is best seen as a device producing for a given accuracy $\frac{1}{2^n}$ a rational number $a_n$ being $\frac{1}{2^n}$-close to $x$, i.e., $|a_n-x|\le\frac{1}{2^n}$. For simplicity we again restrict ourselves to real numbers in the interval $[-1,1]$ (rather than $[-2^n,2^n]$), and to dyadic rationals $\sum_{n<m}\frac{d_n}{2^{n+1}}$ ($d_n\in\{1,\bar1\}$).

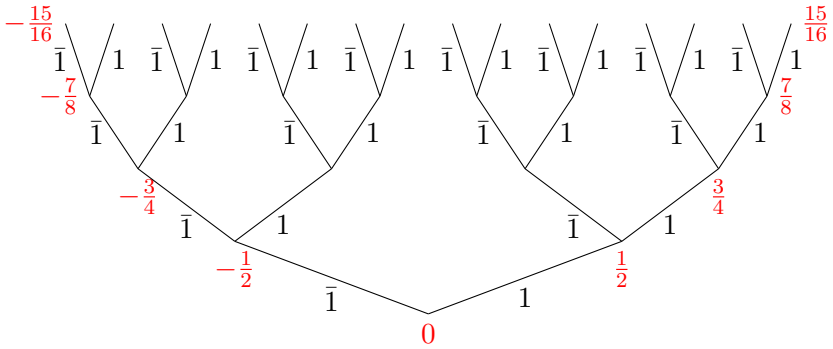



Figure 2: Dyadic rationals.

Clearly we can represent real numbers (in $[-1,1]$) as streams of $1,\bar1$. Now when doing arithmetic operations on such streams the problem of "productivity" arises: suppose we want to add the two streams $\bar1111\ldots$ and $1\bar1\bar1\bar1\ldots$. Then the first digit of the output stream will only be known after we have checked the two input streams long enough, but there is no bound how far we have to look. The well-known cure for this problem is to add a *delay* digit 0 and work with signed digits $\sum_{n<m}\frac{d_n}{2^{n+1}}$, now with $d_n\in\{1,0,\bar1\}$. We have a lot of redundancy here (for instance $\bar11$ and $0\bar1$ both denote $-\frac14$), but this is not a serious problem.

Since 0 as real number is represented by the stream consisting of the digit 0 only, we can simplify our example by removing the nullary clause from the inductive definition of $I_0$, and define $I$ and ${}^{\mathrm{co}}I$ accordingly. We only need ${}^{\mathrm{co}}I$, coinductively defined by the closure axiom

$$\forall_{x\in{}^{\mathrm{co}}I}\exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}I}\Big(x=\frac{x'+d}{2}\Big). \tag{2.1}$$

Therefore, the coinduction axiom is

$$\forall_{x\in P}\exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}I\cup P}\Big(x=\frac{x'+d}{2}\Big)\to P\subseteq{}^{\mathrm{co}}I. \tag{2.2}$$

The canonically associated data type then becomes the algebra $\mathbb{S}$ given by a single binary constructor of type $\mathbb{D}\to\mathbb{S}\to\mathbb{S}$, again denoted by :: (infix). Note that we do not require that $T_{\mathbb{S}}u$ holds for any $u$. Objects $u\in{}^{\mathrm{co}}T_{\mathbb{S}}$ are called *streams* of signed digits.

The computational content of the closure axiom (2.1) now is the destructor $\mathcal{D}_{\mathbb{S}}$ of type $\mathbb{S}\to\mathbb{D}\times\mathbb{S}$, defined by

$$\mathcal{D}_{\mathbb{S}}(s::u)=\langle s,u\rangle.$$

For the coinduction axiom (2.2) we have the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}}$ of type $\tau\to(\tau\to\mathbb{D}\times(\mathbb{S}+\tau))\to\mathbb{S}$. Note that $\mathbb{D}\times(\mathbb{S}+\tau)$ appears since $\mathbb{S}$ has the a single constructor of type $\mathbb{D}\to\mathbb{S}\to\mathbb{S}$. The meaning of ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}}$ is determined by a (non-terminating) computation rule similar to the one for ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{L}}$ (see Section 1.2), with the first case left out.

Our goal in this section is to prove that $[-1,1]$ is closed under division under some conditions, w.r.t. the representation of reals as signed digit streams. For division $\frac{x}{y}$ we clearly need a restriction on the denominator $y$ to stay in the interval $[-1,1]$, and must assume that $|y|$ is strictly positive. For simplicity we assume $\frac{1}{4}\le y$; this can easily be extended to the case $2^{-p}\le y$ (using induction on $p$ and Lemma 2.3 below). The main idea of the algorithm and also of the proof, which is inspired by [CG06], consists in three representations of $\frac{x}{y}$:

$$\frac{x}{y}=\frac{1+\frac{x_1}{y}}{2}=\frac{0+\frac{x_0}{y}}{2}=\frac{-1+\frac{x_{-1}}{y}}{2}$$

where

$$x_1=4\frac{x+\frac{-y}{2}}{2},\quad x_0=2x,\quad x_{-1}=4\frac{x+\frac{y}{2}}{2}.$$

Depending on what we know about $x$ we choose one of these representations of $\frac{x}{y}$ to obtain its first digit. This will give us a corecursive definition of $\frac{x}{y}$.

As we see in the representation of $\frac{x}{y}$, we will use that ${}^{\mathrm{co}}I$ is closed under the average:

**Theorem 2.1.** *The average of two real numbers $x,y$ in ${}^{\mathrm{co}}I$ is in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{x+y}{2}\in{}^{\mathrm{co}}I\Big).$$

*Proof.* The proof in [Ber09, BMST16] can be adapted to the present setting with concrete rather than abstract reals. □

The term extracted from this proof corresponds to an algorithm transforming stream representations of $x$ and $y$ into a stream representation of their average $\frac{x+y}{2}$. For the first $n$ digits in the representation of $\frac{x+y}{2}$ one needs the first $n+1$ digits in the representations of $x$ and $y$.

**Lemma 2.2.** *${}^{\mathrm{co}}I$ is closed under shifting a real $x \le 0$ ($x \ge 0$) by $+1$ ($-1$):*

$$\forall_{x\in{}^{\mathrm{co}}I}(x \le 0 \to x+1 \in {}^{\mathrm{co}}I),$$
$$\forall_{x\in{}^{\mathrm{co}}I}(0 \le x \to x-1 \in {}^{\mathrm{co}}I).$$

*Proof.* We only consider the first claim; the second one is proved similarly. Since we want to prove that something is in ${}^{\mathrm{co}}I$, the proof must use its introduction axiom given in (2.2):

$$\forall_{x\in P}\exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}I\cup P}\Big(x = \frac{x'+d}{2}\Big) \to \forall_x(x \in P \to x \in {}^{\mathrm{co}}I).$$

The crucial point is how to choose the competitor predicate $P$. Looking at the formula we want to prove, we take $P := \{\, x \mid \exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x = y+1)\,\}$ as competitor predicate. Then we have

$$\forall_{x\in P}\exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}I\cup P}\Big(x = \frac{x'+d}{2}\Big) \to \forall_x(\exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x = y+1) \to x \in {}^{\mathrm{co}}I)$$

and since ${}^{\mathrm{co}}I$ is compatible with the real equality, this implies

$$\forall_{x\in P}\exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}I\cup P}\Big(x = \frac{x'+d}{2}\Big) \to \forall_{x\in{}^{\mathrm{co}}I}(x \le 0 \to x+1 \in {}^{\mathrm{co}}I).$$

Therefore it suffices to prove the premise of the formula above. Let $x \in P$ be given. Then by definition of $P$ there is $y$ with $y \in {}^{\mathrm{co}}I$, $y \le 0$ and $x = y+1$. From $y \in {}^{\mathrm{co}}I$ we know $|y| \le 1$ and with $y \le 0$ also $|x| \le 1$. We need $d \in \mathrm{Sd}$ and $x'$ such that

$$(x' \in {}^{\mathrm{co}}I \lor \exists_{y\in{}^{\mathrm{co}}I}(y \le 0 \land x' = y+1)) \land x = \frac{d+x'}{2}.$$

Again from $y \in {}^{\mathrm{co}}I$ we obtain $e \in \mathrm{Sd}$ and $z \in {}^{\mathrm{co}}I$ with $y = \frac{e+z}{2}$. We now distinguish cases on $e \in \mathrm{Sd}$.

*Case* $e = 1$. Then $0 \ge y = \frac{1+z}{2} \ge \frac{1}{2} - \frac{1}{2} = 0$ and hence $x = y+1 = 1$. Picking $d = 1$ and $x' = 1$ gives the claim. Here we need a lemma `CoIOne` stating that 1 is in ${}^{\mathrm{co}}I$. The proof of this lemma (by coinduction) is omitted.

*Case* $e = 0$. Pick $d = 1$ and $x' = z+1$. Then $z = 2y \le 0$ and hence the r.h.s. of the disjunction above holds with $z$ for $y$. Also $x = 2y = \frac{1+2y+1}{2}$.

*Case* $e = -1$. Pick $d = 1$ and $x' = z$. Then the l.h.s. of the disjunction above holds, and $\frac{d+x'}{2} = \frac{1+z}{2} = \frac{-1+z}{2} + 1 = y + 1 = x$. □

We have formalized this proof in the Minlog[2] proof assistant. Minlog has a tool to extract from the proof a term representing its computational content; it is a term in the theory TCF described in Section 1.2. The extracted term is displayed as

```
[u](CoRec ai=>ai)u
 ([u0][case (DesYprod u0)
     (s pair u1 -> [case s
       (SdR -> SdR pair InL cCoIOne)
       (SdM -> SdR pair InR u1)
       (SdL -> SdR pair InL u1)])])
```

Here `[u]` means lambda abstraction $\lambda_u$, and `(CoRec ai=>ai)` is the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{S}}$ defined above, where $\tau$ is $\mathbb{S}$ again. The type of ${}^{\mathrm{co}}\mathcal{R}^{\mathbb{S}}_{\mathbb{S}}$ is $\mathbb{S} \to (\mathbb{S} \to \mathbb{D} \times (\mathbb{S}+\mathbb{S})) \to \mathbb{S}$. The first argument of the corecursion operator is the abstracted variable `u` of type $\mathbb{S}$, and the second `([u0][case ...])` is the "step" function, which first destructs its argument `u0`

[2]The development described here resides in `minlog/examples/analysis`, file `sddiv.scm`

into a pair of a signed digit `s` and another stream `u1`, and then distinguishes cases on `s`. In the `SdR` case the returned pair of type $\mathbb{D} \times (\mathbb{S} + \mathbb{S})$ has `SdR` again as its left component, and as right component the InL (left embedding into a sum type) of a certain stream denoted `cCoIOne`. This is the computational content (hence the "`c`") of `CoIOne`, essentially an infinite sequence of the digit SdR (written $\vec{1}$).

The algorithm represented by the extracted term can be understood as follows. If $s = \mathrm{SdR}$, then $y$ must be non-negative. Hence $y = 0$, and a stream-representation of $y + 1$ is $\vec{1}$. Here we do not need a corecursive call, and hence the result is $\langle \mathrm{SdR}, \mathrm{InL}(\vec{1}\,)\rangle$. The same happens in case $s = \mathrm{SdL}$. Here $y + 1$ can be determined easily by changing the first digit from SdL to SdR and leaving the tail as it is. Hence the result is $\langle \mathrm{SdR}, \mathrm{InL}(u')\rangle$. Only in case $s = \mathrm{SdM}$ we have a corecursive call. Here we change the first digit from SdM to SdR, which however amounts to adding $\frac{1}{2}$ only. Therefore the procedure continues with the tail. Hence the result is $\langle \mathrm{SdR}, \mathrm{InR}(u')\rangle$. Using the computation rule for ${}^{\mathrm{co}}\mathcal{R}^{\mathbb{S}}_{\mathbb{S}}$ we can now describe the computational content as a function `add1`: $\mathbb{S} \to \mathbb{S}$ defined by

$$\begin{aligned} \mathrm{add1}(\mathrm{SdR} :: u) &:= [\mathrm{SdR}, \mathrm{SdR}, \dots], \\ \mathrm{add1}(\mathrm{SdM} :: u) &:= \mathrm{SdR} :: \mathrm{add1}(u), \\ \mathrm{add1}(\mathrm{SdL} :: u) &:= \mathrm{SdR} :: u. \end{aligned}$$

A similar argument for the second part of the lemma gives sub1: $\mathbb{S} \to \mathbb{S}$ with

$$\begin{aligned} \mathrm{sub1}(\mathrm{SdR} :: u) &:= \mathrm{SdL} :: u, \\ \mathrm{sub1}(\mathrm{SdM} :: u) &:= \mathrm{SdL} :: \mathrm{sub1}(u), \\ \mathrm{sub1}(\mathrm{SdL} :: u) &:= [\mathrm{SdL}, \mathrm{SdL}, \dots]. \end{aligned}$$

**Lemma 2.3.** *For $x$ in ${}^{\mathrm{co}}I$ with $|x| \le \frac{1}{2}$ we have $2x$ in ${}^{\mathrm{co}}I$:*

$$\forall_{x \in {}^{\mathrm{co}}I}\Big(|x| \le \frac{1}{2} \to 2x \in {}^{\mathrm{co}}I\Big).$$

*Proof.* Let $x \in {}^{\mathrm{co}}I$ be given. From the closure axiom (2.1) for ${}^{\mathrm{co}}I$ we obtain $d \in \mathrm{Sd}$ and $x' \in {}^{\mathrm{co}}I$ such that $x = \frac{d+x'}{2}$. We distinguish cases on $d \in \mathrm{Sd}$.

*Case* $d = 1$. Then $x = \frac{1+x'}{2}$ and hence $2x = 1 + x'$. Since $|x| \le \frac{1}{2}$ we have $x' \le 0$. Now the first part of Lemma 2.2 gives the claim.

*Case* $d = 0$. Then $x = \frac{0+x'}{2}$ and hence $2x = x' \in {}^{\mathrm{co}}I$.

*Case* $d = -1$. Then $x = \frac{-1+x'}{2}$ and hence $2x = -1 + x'$. Since $|x| \le \frac{1}{2}$ we have $0 \le x'$. Now the second part of Lemma 2.2 gives the claim. □

Using arguments similar to those in the remark after Lemma 2.2 we can see that the corresponding algorithm can be written as a function Double: $\mathbb{S} \to \mathbb{S}$ with

$$\begin{aligned} \mathrm{Double}(\mathrm{SdR} :: u) &:= \mathrm{add1}(u), \\ \mathrm{Double}(\mathrm{SdM} :: u) &:= u, \\ \mathrm{Double}(\mathrm{SdL} :: u) &:= \mathrm{sub1}(u) \end{aligned}$$

where add1 and sub1 are the functions from this remark.

**Lemma 2.4.** *For $x, y$ in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$ ($x \le 0$) we have $2x - y$ ($2x + y$) in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to 0 \le x \to 4\frac{x + \frac{-y}{2}}{2} \in {}^{\mathrm{co}}I\Big),$$
$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to x \le 0 \to 4\frac{x + \frac{y}{2}}{2} \in {}^{\mathrm{co}}I\Big).$$

*Proof.* In the formulas above instead of $2x \pm y$ we have written $4\frac{x+(\pm y/2)}{2}$ to make Theorem 2.1 applicable. We also use two lemmas stating that ${}^{\mathrm{co}}I$ is closed under $x \mapsto \frac{x}{2}$ and $x \mapsto -x$. To prove the first claim, let $x, y$ in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$. Then clearly $\frac{-y}{2} \in {}^{\mathrm{co}}I$, and $\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}I$ by Theorem 2.1. We have

$$\Big|\frac{x + \frac{-y}{2}}{2}\Big| = \Big|\frac{2x - y}{4}\Big| \le \Big|\frac{2y - y}{4}\Big| = \Big|\frac{y}{4}\Big| \le \frac{1}{4}. \tag{2.3}$$

Hence we can apply Lemma 2.3 twice and obtain $4\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}I$. The proof of the second claim is similar. □

The computational content of the proofs is $\mathrm{AuxL}, \mathrm{AuxR} \colon \mathbb{S} \to \mathbb{S} \to \mathbb{S}$:

$$\mathrm{AuxL}(u, v) := \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(-v)))),$$
$$\mathrm{AuxR}(u, v) := \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(v)))).$$

Here $h, - \colon \mathbb{S} \to \mathbb{S}$ represent the computational content of the two lemmas used in the proof; both are proved by coinduction. The function $h$ prepends SdM to the stream, and $-$ negates all digits:

$$h(u) := \mathrm{SdM} :: u, \qquad \begin{aligned} -(\mathrm{SdR} :: u) &:= \mathrm{SdL} :: (-u), \\ -(\mathrm{SdM} :: u) &:= \mathrm{SdM} :: (-u), \\ -(\mathrm{SdL} :: u) &:= \mathrm{SdR} :: (-u). \end{aligned}$$

**Theorem 2.5.** *For $x, y$ in ${}^{\mathrm{co}}I$ with $\frac{1}{4} \le y$ and $|x| \le y$ we have $\frac{x}{y}$ in ${}^{\mathrm{co}}I$:*

$$\forall_{x,y\in{}^{\mathrm{co}}I}\Big(\frac{1}{4} \le y \to |x| \le y \to \frac{x}{y} \in {}^{\mathrm{co}}I\Big).$$

*Proof.* The proof uses coinduction (2.2) on ${}^{\mathrm{co}}I$ with $P := \{\, z \mid \exists_{x,y\in{}^{\mathrm{co}}I}(|x| \le y \land \frac{1}{4} \le y \land z = \frac{x}{y}) \,\}$. It suffices to prove (2.2)'s premise for this $P$. Let $x, y, z$ be given with $x, y \in {}^{\mathrm{co}}I$, $|x| \le y$, $\frac{1}{4} \le y$ and $z = \frac{x}{y}$. From $|x| \le y$ we have $z \le 1$. By a trifold application of the closure axiom (2.1) to $x$ we obtain $d_1, d_2, d_3 \in \mathrm{Sd}$ and $\tilde{x} \in {}^{\mathrm{co}}I$ such that $x = \frac{4d_1+2d_2+d_3+\tilde{x}}{8}$ or $x = d_1 d_2 d_3 \tilde{x}$ for short. We now distinguish three cases.

If $x = 1 d_2 d_3 \tilde{x}$, $x = 01 d_3 \tilde{x}$ or $x = 001\tilde{x}$, then $0 \le x$. Pick $d = 1$ and $z' = \frac{x'}{y}$ with $x' = 4\frac{x+\frac{-y}{2}}{2}$. Then $x' \in {}^{\mathrm{co}}I$ by Lemma 2.4. From (2.3) we also obtain $|x'| \le y$ and hence $z' \in P$. One can easily check that $z = \frac{1+z'}{2}$.

If $x = \bar{1} d_2 d_3 \tilde{x}$, $x = 0\bar{1} d_3 \tilde{x}$ or $x = 00\bar{1}\tilde{x}$, then $x \le 0$. Pick $d = -1$ and $z' = \frac{x'}{y}$ with $x' = 4\frac{x+\frac{y}{2}}{2}$. We can then proceed as in the first case.

```
[u,u0](CoRec ai=>ai)u
 ([u1][case (cCoIClosure u1)
   (s pair u2 -> [case s
     (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
     (SdM -> [case (cCoIClosure u2)
       (s0 pair u3 -> [case s0
        (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
        (SdM -> [case (cCoIClosure u3)
         (s1 pair u4 -> [case s1
          (SdR -> SdR pair InR(cCoIDivSatCoIClAuxR u1 u0))
          (SdM -> SdM pair InR(cCoIToCoIDouble u1))
          (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
        (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
     (SdL -> SdL pair InR(cCoIDivSatCoIClAuxL u1 u0))])])
```

Figure 3: Extracted term for Theorem 2.5.

The final case is $x = 000\tilde{x}$. Then $|x| \le \frac{1}{8}$; pick $d = 0$ and $z' = \frac{x'}{y}$ with $x' = 2x$. We obtain $|x'| \le \frac{1}{4} \le y$ and with Lemma 2.3 also $x' \in {}^{\mathrm{co}}I$. Therefore $z' \in P$, and $z = \frac{z'}{2}$ is easily checked. □

The term Minlog extracts is displayed in Figure 3. The three occurrences of `cCoIClosure` correspond to the trifold application of the closure axioms (2.1) to $x$. The seven cases $x = 1d_2d_3\tilde{x}$, $x = 01d_3\tilde{x}$, $x = 001\tilde{x}$, $x = 000\tilde{x}$, $x = \bar{1}d_2d_3\tilde{x}$, $x = 0\bar{1}d_3\tilde{x}$ and $x = 00\bar{1}\tilde{x}$ are clearly visible. In the first three cases `SdR` corresponds to picking $d = 1$ and usage of the AuxR function from Lemma 2.4, and similarly in the last three cases `SdL` corresponds to picking $d = -1$ and usage of AuxL. In the middle case `SdM` corresponds to $d = 0$; there we use the computational content of Lemma 2.3. We can describe the extracted term by a function $\mathrm{Div}\colon \mathbb{S} \to \mathbb{S} \to \mathbb{S}$ corecursively defined by

$$\mathrm{Div}(u,v) := \begin{cases} \mathrm{SdR} :: \mathrm{Div}(\mathrm{AuxR}(u,v),v) & \text{if } u = 1\tilde{u} \vee u = 01\tilde{u} \vee u = 001\tilde{u}, \\ \mathrm{SdM} :: \mathrm{Div}(\mathrm{Double}(u),v) & \text{if } u = 000\tilde{u}, \\ \mathrm{SdL} :: \mathrm{Div}(\mathrm{AuxL}(u,v),v) & \text{if } u = \bar{1}\tilde{u} \vee u = 0\bar{1}\tilde{u} \vee u = 00\bar{1}\tilde{u}. \end{cases}$$

As abbreviation in the case distinction, we have denoted $\mathrm{SdL}, \mathrm{SdM}$ and SdR by $\bar{1}, 0$ and 1 and we have omitted the constructor ::.

We use this description of the extracted term to see how far we have to look into $u$ and $v$ to determine the first $n$ entries of $\mathrm{Div}(u,v)$. To this end we write the above equation as

$$\mathrm{Div}(u,v) = d(u) :: \mathrm{Div}(G(u,v),v),$$

where $d(u)$ depends on the first three digits of $u$, and $G(u,v)$ is one of $\mathrm{AuxR}(u,v)$, $\mathrm{Double}(u)$ or $\mathrm{AuxL}(u,v)$, according to the present case. Recall

$$\mathrm{AuxL}(u,v) := \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u,h(n(v))))),$$
$$\mathrm{AuxR}(u,v) := \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u,h(v)))).$$

By the equations for $-$, $h$, Av and Double we see that the first $n$ entries of

| | |
|---|---|
| $-u$ | need the first $n$ entries of $u$, |
| $h(u)$ | need the first $n-1$ entries of $u$, |
| $\mathrm{Av}(u,v)$ | need the first $n+1$ entries of $u$ and $v$, |
| $\mathrm{Double}(u)$ | need the first $n+1$ entries of $u$. |

Hence $\mathrm{AuxR}(u,v)$, $\mathrm{AuxL}(u,v)$ and $G(u,v)$ all need at most the first $n+3$ entries of $u$ and $n+2$ entries of $v$. Iterating the above equation for $G$ gives for $\mathrm{Div}(u,v)$ the representation

$$d(u) :: d(G(u,v)) :: d(G(G(u,v),v)) :: d(G(G(G(u,v),v),v),v)\ldots$$

Therefore the first $n$ entries of $\mathrm{Div}(u,v)$ depend on at most the first $3n$ entries of $u$ and the first $3n-1$ entries of $v$.

2.2. **Division for binary reflected digit streams.** In the stream representation of real numbers described in Section 2.1 adjacent dyadics of the same length can differ in many digits:

$$\frac{7}{16} \sim 1\bar{1}11, \qquad \frac{9}{16} \sim 11\bar{1}\bar{1}.$$

A possible cure is to *flip* after an occurrence of 1; see Figure 4. The result is called binary reflected (or Gray-) code.

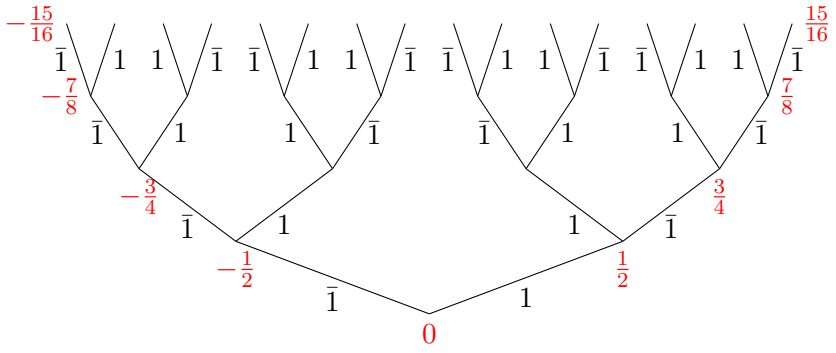



Figure 4: Gray code.

Then two dyadics of the same length are adjacent if and only if they differ in exactly one digit. For instance we now have

$$\frac{7}{16} \sim 111\bar{1}, \qquad \frac{9}{16} \sim 1\bar{1}1\bar{1}.$$

This is a desirable property of stream-coded real numbers.

Again when doing arithmetical operations on Gray code the problem of productivity arises, which is dealt with in a somewhat different way. The idea is to introduce two "modes" when generating the code, and flip from one mode to the other whenever we encounter the digit 1. More precisely, instead of the predicate $I$ we now use two predicates $G, H$ ($G$ for

Gray, $H$ next character) and flip from one to the other after reading 1. They are defined by simultaneous induction, with clauses

$$\forall_{d\in\mathrm{Psd}}\forall_{x\in G}\Big(-d\frac{x-1}{2}\in G\Big),\quad \forall_{x\in H}\Big(\frac{x}{2}\in G\Big),$$
$$\forall_{d\in\mathrm{Psd}}\forall_{x\in G}\Big(d\frac{x+1}{2}\in H\Big),\quad \forall_{x\in H}\Big(\frac{x}{2}\in H\Big).$$

Here Psd (for proper signed digit) is the (inductive) predicate consisting of the integers $1,-1$. We require that Psd has computational content, in the booleans $\mathbb{B}=\{\mathtt{tt},\mathtt{ff}\}$. Note that there are no nullary clauses (as for $I$), since they are not needed. We are only interested in the duals ${}^{\mathrm{co}}G,{}^{\mathrm{co}}H$, coinductively defined by the simultaneous closure axioms

$$\begin{aligned}&\forall_{x\in{}^{\mathrm{co}}G}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G}\Big(x=-d\frac{x'-1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H}\Big(x=\frac{x'}{2}\Big)\Big)\\&\forall_{x\in{}^{\mathrm{co}}H}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G}\Big(x=d\frac{x'+1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H}\Big(x=\frac{x'}{2}\Big)\Big)\end{aligned}\tag{2.4}$$

and the simultaneous coinduction axioms

$$\begin{aligned}&\forall_{x\in P}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x=-d\frac{x'-1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x=\frac{x'}{2}\Big)\Big)\to\\&\forall_{x\in Q}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x=d\frac{x'+1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x=\frac{x'}{2}\Big)\Big)\to\\&P\subseteq{}^{\mathrm{co}}G\qquad\text{and the same with conclusion } Q\subseteq{}^{\mathrm{co}}H.\end{aligned}\tag{2.5}$$

We require that the predicates $G,H$ and therefore ${}^{\mathrm{co}}G,{}^{\mathrm{co}}H$ as well have computational content, in simultaneously defined algebras $\mathbb{G}$ and $\mathbb{H}$ given by the constructors

$$\begin{aligned}&\mathrm{Lr}\colon\mathbb{B}\to\mathbb{G}\to\mathbb{G},\qquad&&\mathrm{U}\colon\mathbb{H}\to\mathbb{G}\qquad\text{for }\mathbb{G},\\&\mathrm{Fin}\colon\mathbb{B}\to\mathbb{G}\to\mathbb{H},\qquad&&\mathrm{D}\colon\mathbb{H}\to\mathbb{H}\qquad\text{for }\mathbb{H}.\end{aligned}$$

We write $\mathrm{Lr}_{\mathtt{tt}}(u)$ for $\mathrm{Lr}(\mathtt{tt},u)$ and $\mathrm{Lr}_{\mathtt{ff}}(u)$ for $\mathrm{Lr}(\mathtt{ff},u)$, and similarly for Fin. Then $\mathrm{Lr}_{\mathtt{tt}/\mathtt{ff}}$ means left/right, $\mathrm{Fin}_{\mathtt{tt}/\mathtt{ff}}$ means finally left/right. The delay constructors are U ("undefined") for $\mathbb{G}$ and D ("delay") for $\mathbb{H}$. Figure 5 indicates how these constructors generate Gray code.

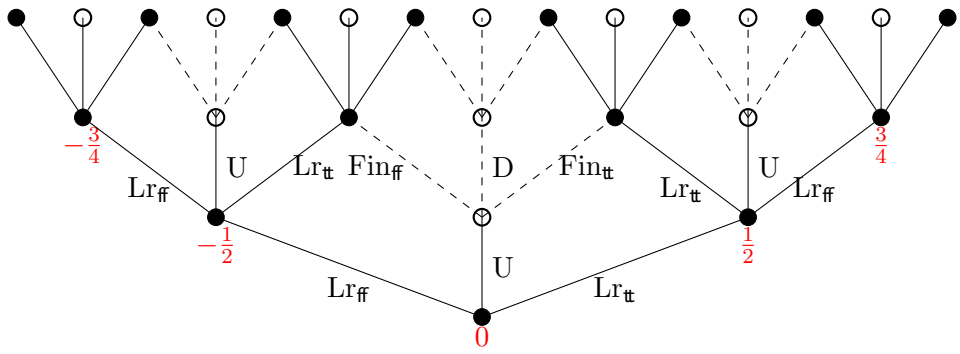



Figure 5: Gray code with delay.

Totality $T_{\mathbb{G}}$, $T_{\mathbb{H}}$ and cototality ${}^{\mathrm{co}}T_{\mathbb{G}}$, ${}^{\mathrm{co}}T_{\mathbb{H}}$ can be defined as for $\mathbb{S}$ above. Objects $u\in{}^{\mathrm{co}}T_{\mathbb{G}}$ are called *streams* in Gray code.

The computational content of the closure axioms (2.4) are the destructors $\mathcal{D}_{\mathbb{G}}\colon \mathbb{G} \to (\mathbb{B}\times\mathbb{G})+\mathbb{H}$ and $\mathcal{D}_{\mathbb{H}}\colon \mathbb{H} \to (\mathbb{B}\times\mathbb{G})+\mathbb{H}$ defined by

$$\begin{aligned} \mathcal{D}_{\mathbb{G}}(\mathrm{Lr}_b(u)) &= \mathrm{InL}\langle b,u\rangle, \quad & \mathcal{D}_{\mathbb{G}}(\mathrm{U}v) &= \mathrm{InR}(v),\\ \mathcal{D}_{\mathbb{H}}(\mathrm{Fin}_b(u)) &= \mathrm{InL}\langle b,u\rangle, \quad & \mathcal{D}_{\mathbb{H}}(\mathrm{D}v) &= \mathrm{InR}(v). \end{aligned}$$

The computational content of the coinduction axioms (2.5) are instances of the simultaneous corecursion operators

$$\begin{aligned} {}^{\mathrm{co}}\mathcal{R}_{\mathbb{G}}^{(\mathbb{G},\mathbb{H}),(\sigma,\tau)} &\colon \sigma\to\delta_{\mathbb{G}}\to\delta_{\mathbb{H}}\to\mathbb{G}\\ {}^{\mathrm{co}}\mathcal{R}_{\mathbb{H}}^{(\mathbb{G},\mathbb{H}),(\sigma,\tau)} &\colon \tau\to\delta_{\mathbb{G}}\to\delta_{\mathbb{H}}\to\mathbb{H} \end{aligned} \tag{2.6}$$

with step types

$$\begin{aligned} \delta_{\mathbb{G}} &:= \sigma\to\mathbb{B}\times(\mathbb{G}+\sigma)+(\mathbb{H}+\tau),\\ \delta_{\mathbb{H}} &:= \tau\to\mathbb{B}\times(\mathbb{G}+\sigma)+(\mathbb{H}+\tau). \end{aligned}$$

The type $\mathbb{B}\times(\mathbb{G}+\sigma)+(\mathbb{H}+\tau)$ appears since $\mathbb{G}$ has the two constructors $\mathrm{Lr}\colon \mathbb{B}\to\mathbb{G}\to\mathbb{G}$ and $\mathrm{U}\colon \mathbb{H}\to\mathbb{G}$, and $\mathbb{H}$ has the two constructors $\mathrm{Fin}\colon \mathbb{B}\to\mathbb{G}\to\mathbb{H}$ and $\mathrm{D}\colon \mathbb{H}\to\mathbb{H}$. Omitting the upper indices of ${}^{\mathrm{co}}\mathcal{R}$ the computation rules for the terms ${}^{\mathrm{co}}\mathcal{R}_{\mathbb{G}}sgh$ and ${}^{\mathrm{co}}\mathcal{R}_{\mathbb{H}}tgh$ are

$$\begin{aligned} {}^{\mathrm{co}}\mathcal{R}_{\mathbb{G}}sgh &= \begin{cases} \mathrm{Lr}(b,u) & \text{if } g(s)\equiv \mathrm{InL}\langle b,\mathrm{InL}_{\mathbb{G}+\sigma}u\rangle\\ \mathrm{Lr}(b,{}^{\mathrm{co}}\mathcal{R}_{\mathbb{G}}s'gh) & \text{if } g(s)\equiv \mathrm{InL}\langle b,\mathrm{InR}_{\mathbb{G}+\sigma}s'\rangle\\ \mathrm{U}(v) & \text{if } g(s)\equiv \mathrm{InR}(\mathrm{InL}_{\mathbb{H}+\tau}v)\\ \mathrm{U}({}^{\mathrm{co}}\mathcal{R}_{\mathbb{H}}tgh) & \text{if } g(s)\equiv \mathrm{InR}(\mathrm{InR}_{\mathbb{H}+\tau}t) \end{cases}\\ {}^{\mathrm{co}}\mathcal{R}_{\mathbb{H}}tgh &= \begin{cases} \mathrm{Fin}(b,u) & \text{if } h(t)\equiv \mathrm{InL}\langle b,\mathrm{InL}_{\mathbb{G}+\sigma}u\rangle\\ \mathrm{Fin}(b,{}^{\mathrm{co}}\mathcal{R}_{\mathbb{G}}sgh) & \text{if } h(t)\equiv \mathrm{InL}\langle b,\mathrm{InR}_{\mathbb{G}+\sigma}s\rangle\\ \mathrm{D}(v) & \text{if } h(t)\equiv \mathrm{InR}(\mathrm{InL}_{\mathbb{H}+\tau}v)\\ \mathrm{D}({}^{\mathrm{co}}\mathcal{R}_{\mathbb{H}}t'gh) & \text{if } h(t)\equiv \mathrm{InR}(\mathrm{InR}_{\mathbb{H}+\tau}t') \end{cases} \end{aligned} \tag{2.7}$$

with $s$ of type $\sigma$ and $t$ of type $\tau$.

Our goal in this section is to prove that $[-1,1]$ is closed under division $\frac{x}{y}$ (for $|x|\le|y|>0$), w.r.t. the representation of reals in Gray code. We proceed essentially as for the signed digit case. The main difference is that the simultaneous definition of ${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$ makes it necessary to use simultaneous coinduction.

**Theorem 2.6.** *The average of two real numbers $x,y$ in ${}^{\mathrm{co}}G$ is in ${}^{\mathrm{co}}G$:*

$$\forall_{x,y\in{}^{\mathrm{co}}G}\Big(\frac{x+y}{2}\in{}^{\mathrm{co}}G\Big).$$

*Proof.* The proof in [BMST16] can be adapted to the present setting with concrete rather than abstract reals. □

**Lemma 2.7.** *${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$ are closed under minus:*

$$\forall_{x\in{}^{\mathrm{co}}G}(-x\in{}^{\mathrm{co}}G),\qquad \forall_{x\in{}^{\mathrm{co}}H}(-x\in{}^{\mathrm{co}}H).$$

*Proof.* We prove both claims simultaneously, by coinduction:

$$\forall_{x\in P}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x=-d\frac{x'-1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x=\frac{x'}{2}\Big)\Big)\to$$
$$\forall_{x\in Q}\Big(\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x=d\frac{x'+1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x=\frac{x'}{2}\Big)\Big)\to$$
$$\forall_x(x\in P\to x\in{}^{\mathrm{co}}G)\wedge\forall_x(x\in Q\to x\in{}^{\mathrm{co}}H).$$

We choose $P:=\{\,x\mid -x\in{}^{\mathrm{co}}G\,\}$ and $Q:=\{\,x\mid -x\in{}^{\mathrm{co}}H\,\}$. Since ${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$ are invariant under real equality (also proven by coinduction) we have to prove the two premises. We only consider the first one; the second is proved similarly. Let $x\in P$ be given. The goal is

$$\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x=-d\frac{x'-1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x=\frac{x'}{2}\Big).$$

From $x\in P$ we get $-x\in{}^{\mathrm{co}}G$ and therefore

$$\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G}\Big(-x=-d\frac{x'-1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H}\Big(-x=\frac{x'}{2}\Big)$$

by the closure axiom. If the right hand side of this disjunction holds, we are done by definition of $Q$. If the left hand side holds, we have $x=d\frac{x'-1}{2}$ for some $d\in\mathrm{Psd}$ and $x'\in{}^{\mathrm{co}}G$. To get the left hand side of the disjunction in the goal formula, we take $d:=-d$ and $x':=x'$. □

The function we get from the proof takes a Gray code and flips all $\mathtt{tt}$ to $\mathtt{ff}$ and the other way around. We denote this function by $-$. Then the computation rules are:

$$-(\mathrm{Lr}_{\mathtt{tt}}(u))=\mathrm{Lr}_{\mathtt{ff}}(-u),\quad -(\mathrm{Fin}_{\mathtt{tt}}(u))=\mathrm{Fin}_{\mathtt{ff}}(-u),$$
$$-(\mathrm{Lr}_{\mathtt{ff}}(u))=\mathrm{Lr}_{\mathtt{tt}}(-u),\quad -(\mathrm{Fin}_{\mathtt{ff}}(u))=\mathrm{Fin}_{\mathtt{tt}}(-u)),$$
$$-(\mathrm{U}(v))=\mathrm{U}(-v),\qquad -(\mathrm{D}(v))=\mathrm{D}(-v).$$

**Lemma 2.8.** *${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$ are equivalent:*

$$\forall_{x\in{}^{\mathrm{co}}G}(x\in{}^{\mathrm{co}}H),\qquad \forall_{x\in{}^{\mathrm{co}}H}(x\in{}^{\mathrm{co}}G).$$

*Proof.* In the coinduction axiom we choose $P:={}^{\mathrm{co}}H$ and $Q:={}^{\mathrm{co}}G$ and the conclusion in the coinduction axiom is our goal. Therefore we have to prove the premises of the coinduction axiom. We only consider the first one; the secound one is proved similarly. Let $x\in{}^{\mathrm{co}}H$ be given. By the closure axiom of ${}^{\mathrm{co}}H$ we have

$$\exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G}\Big(x=d\frac{x'+1}{2}\Big)\vee\exists_{x'\in{}^{\mathrm{co}}H}\Big(x=\frac{x'}{2}\Big)$$

If the second part of this disjunction holds, we get directly the second part of the first premise. If the first part of the disjunction holds, we have $x=d\frac{x'+1}{2}$ for some $d\in\mathrm{Psd}$ and $x'\in{}^{\mathrm{co}}G$. Then $x=-d\frac{(-x')-1}{2}$ and using Lemma 2.7 we get the first part of disjunction in the premise. □

As computational content we have two functions $\mathrm{ToCoH}\colon\mathbb{G}\to\mathbb{H}$ and $\mathrm{ToCoG}\colon\mathbb{H}\to\mathbb{G}$, with computation rules

$$\mathrm{ToCoH}(\mathrm{Lr}_b(u))=\mathrm{Fin}_b(-u),\quad \mathrm{ToCoG}(\mathrm{Fin}_b(u))=\mathrm{Lr}_b(-u),$$
$$\mathrm{ToCoH}(\mathrm{U}(v))=\mathrm{D}(v),\qquad \mathrm{ToCoG}(\mathrm{D}(v))=\mathrm{U}(v).$$

Note that no corecursive call is involved. We denote $\mathrm{ToCoG}(v)$, $\mathrm{ToCoH}(u)$ by $\tilde{v}$, $\tilde{u}$, respectively.

We show that ${}^{\mathrm{co}}G$ is closed under shifting a real $x \le 0$ ($x \ge 0$) by $+1$ ($-1$):

$$\forall_{x\in{}^{\mathrm{co}}G}((x \le 0 \to x+1 \in {}^{\mathrm{co}}G) \land (0 \le x \to x-1 \in {}^{\mathrm{co}}G)).$$

The proof is by coinduction, simultaneously with the same proposition for ${}^{\mathrm{co}}H$. For the coinduction proof it is helpful to reformulate the claim in such a way that the conclusion has the form $x \in {}^{\mathrm{co}}G$:

**Lemma 2.9.**

$$\forall_x(\exists_{y\in{}^{\mathrm{co}}G}(y \le 0 \land (x = y+1 \lor x = -(y+1))) \to x \in {}^{\mathrm{co}}G).$$

*Proof.* We use the introduction axiom for ${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$, and simultaneously prove

$$\forall_x(\exists_{y\in{}^{\mathrm{co}}H}(y \le 0 \land (x = y+1 \lor x = -(y+1))) \to x \in {}^{\mathrm{co}}H).$$

The competitor predicates are

$$P := \{\, x \mid \exists_{y\in{}^{\mathrm{co}}G}(y \le 0 \land (x = y+1 \lor x = -(y+1))) \,\},$$
$$Q := \{\, x \mid \exists_{y\in{}^{\mathrm{co}}H}(y \le 0 \land (x = y+1 \lor x = -(y+1))) \,\}.$$

It suffices to prove the premises of the coinduction axiom for these $P$, $Q$. We only consider the first part; the second is proved similarly. Let $x \in P$ be given. Then we have $y$ with $y \in {}^{\mathrm{co}}G$, $y \le 0$ and $x = y+1 \lor x = -(y+1)$. From $y \in {}^{\mathrm{co}}G$ we know $|y| \le 1$ and with $y \le 0$ also $|x| \le 1$. We need to prove the disjunction

$$D := \exists_{d\in\mathrm{Psd}}\exists_{x'\in{}^{\mathrm{co}}G\cup P}\Big(x = -d\frac{x'-1}{2}\Big) \lor \exists_{x'\in{}^{\mathrm{co}}H\cup Q}\Big(x = \frac{x'}{2}\Big).$$

From $y \in {}^{\mathrm{co}}G$ we obtain either (i) $e \in \mathrm{Psd}$, $z \in {}^{\mathrm{co}}G$ with $y = -e\frac{z-1}{2}$ or else (ii) $z \in {}^{\mathrm{co}}H$ with $y = \frac{z}{2}$. In case (i) we have $|z| \le 1$ since $z \in {}^{\mathrm{co}}G$. We distinguish cases on $e \in \mathrm{Psd}$, and use Lemma 2.7.

*Case* $e = 1$. Then $0 \ge y = -\frac{z-1}{2} \ge -\frac{1}{2} + \frac{1}{2} = 0$, hence $y = 0$. If $x = y+1$, take $x = 1$ and go for the l.h.s. of the disjunction $D$. Picking $d = 1$ and $x' = -1$ gives the claim (since $-1 \in {}^{\mathrm{co}}G$). If $x = -(y+1)$, take $x = -1$ and go for the l.h.s. of the disjunction $D$. Picking $d = -1$ and $x' = -1$ gives the claim (since $-1 \in {}^{\mathrm{co}}G$).

*Case* $e = -1$. If $x = y+1$, go for the l.h.s. of the disjunction $D$, picking $d = 1$ and $x' = -z$. Then $x = y+1 = \frac{z-1}{2}+1 = \frac{z+1}{2} = -\frac{-z-1}{2}$. If $x = -(y+1)$, go for the l.h.s. of the disjunction $D$, picking $d = -1$ and $x' = -z$. Then $x = -(y+1) = -\frac{z-1}{2} - 1 = \frac{-z+1}{2} - 1 = \frac{-z-1}{2}$.

In case (ii) we have $|z| \le 1$ (since $z \in {}^{\mathrm{co}}H$) and $z \le 0$ (since $y = \frac{z}{2}$ and $y \le 0$). Hence $|z+1| \le 1$. If $x = y+1$, go for the l.h.s. of the disjunction $D$, picking $d = 1$ and $x' = -(z+1)$. Then $-(z+1) \in P$ (since also $z \in {}^{\mathrm{co}}G$ and $z \le 0$), and $x = y+1 = \frac{z}{2}+1 = \frac{z+2}{2} = -\frac{-(z+1)-1}{2}$. If $x = -(y+1)$, go for the l.h.s. of the disjunction $D$, with $d = -1$ and $x' = -(z+1)$. Then again $-(z+1) \in P$ and $x = -(y+1) = \frac{-(z+1)-1}{2}$. □

As computational content of a formalization of this proof Minlog returns a term involving simultaneous corecursion. The corresponding algorithm can be described as a pair of two functions $\mathrm{shG}\colon \mathbb{G} \to \mathbb{B} \to \mathbb{G}$ and $\mathrm{shH}\colon \mathbb{H} \to \mathbb{B} \to \mathbb{H}$ defined by

$$\mathrm{shG}(\mathrm{Lr}_{\mathtt{tt}}(u), b) = \mathrm{Lr}_b(-1), \qquad \mathrm{shH}(\mathrm{Fin}_{\mathtt{tt}}(u), b) = \mathrm{Fin}_b(1),$$
$$\mathrm{shG}(\mathrm{Lr}_{\mathtt{ff}}(u), b) = \mathrm{Lr}_b(-u), \qquad \mathrm{shH}(\mathrm{Fin}_{\mathtt{ff}}(u), b) = \mathrm{Fin}_b(-u)),$$
$$\mathrm{shG}(\mathrm{U}(v), b) = \mathrm{Lr}_b(\mathrm{shH}(\tilde{v}, \mathtt{ff})), \qquad \mathrm{shH}(\mathrm{D}(v), b) = \mathrm{Fin}_b(\mathrm{shG}(\tilde{v}, \mathtt{tt})).$$

Recall that $v \mapsto \tilde{v}$ is the function extracted from the proof of Lemma 2.8 stating that ${}^{\mathrm{co}}\mathcal{G}$ and ${}^{\mathrm{co}}H$ are equivalent. If $b = \mathtt{tt}$, we have computational content of the first part of the lemma. If $b = \mathtt{ff}$, we have the second part.

**Lemma 2.10.** *For $x$ in ${}^{\mathrm{co}}\mathcal{G}$ with $|x| \le \frac{1}{2}$ we have $2x$ in ${}^{\mathrm{co}}\mathcal{G}$:*

$$\forall_{x \in {}^{\mathrm{co}}\mathcal{G}}\Big(|x| \le \frac{1}{2} \to 2x \in {}^{\mathrm{co}}\mathcal{G}\Big).$$

*Proof.* Let $x \in {}^{\mathrm{co}}\mathcal{G}$ be given. From the closure axiom for ${}^{\mathrm{co}}\mathcal{G}$ we obtain either (i) $d \in \mathrm{Psd}$, $x' \in {}^{\mathrm{co}}\mathcal{G}$ with $x = -d\frac{x'-1}{2}$, or else (ii) $x' \in {}^{\mathrm{co}}H$ with $x = \frac{x'}{2}$. In case (i) we distinguish cases on $d \in \mathrm{Psd}$. In case $d = 1$ we obtain $x = \frac{-x'+1}{2}$ and hence $2x = -x' + 1$. Since $|x| \le \frac{1}{2}$ we have $x' \le 0$. Now the first part of Lemma 2.9 gives the claim. The case $d = -1$ is similar, using the second part of Lemma 2.9. In case (ii) we have $2x = x'$, hence the goal $2x \in {}^{\mathrm{co}}\mathcal{G}$ follows from the equivalence of ${}^{\mathrm{co}}\mathcal{G}$ and ${}^{\mathrm{co}}H$. □

Using arguments similar to those in the remark after Lemma 2.9 we can see that the corresponding algorithm can be written as a function $\mathrm{Double}\colon \mathbb{G} \to \mathbb{G}$ with

$$\begin{aligned}\mathrm{Double}(\mathrm{Lr}_{\mathtt{tt}}(u)) &:= \mathrm{shG}(-u, \mathtt{tt}),\\ \mathrm{Double}(\mathrm{Lr}_{\mathtt{ff}}(u)) &:= \mathrm{shG}(-u, \mathtt{ff}),\\ \mathrm{Double}(\mathrm{U}(v)) &:= \tilde{v}\end{aligned}$$

where shG and $v \mapsto \tilde{v}$ are the functions from this remark.

Parallel to Lemma 2.4 we can now prove

**Lemma 2.11.** *For $x, y$ in ${}^{\mathrm{co}}\mathcal{G}$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$ ($x \le 0$) we have $2x - y$ ($2x + y$) in ${}^{\mathrm{co}}\mathcal{G}$:*

$$\begin{aligned}&\forall_{x,y \in {}^{\mathrm{co}}\mathcal{G}}\Big(\frac{1}{4} \le y \to |x| \le y \to 0 \le x \to 4\frac{x + \frac{-y}{2}}{2} \in {}^{\mathrm{co}}\mathcal{G}\Big),\\ &\forall_{x,y \in {}^{\mathrm{co}}\mathcal{G}}\Big(\frac{1}{4} \le y \to |x| \le y \to x \le 0 \to 4\frac{x + \frac{y}{2}}{2} \in {}^{\mathrm{co}}\mathcal{G}\Big).\end{aligned}$$

*Proof.* We only consider the first claim, and again use Theorem 2.6 on the average. In the formulas above instead of $2x \pm y$ we have written $4\frac{x+(\pm y/2)}{2}$ to make Theorem 2.6 applicable. One can see easily that ${}^{\mathrm{co}}\mathcal{G}$ is closed under $x \mapsto \frac{x}{2}$. Formally, this is proved by coinduction, and the corresponding algorithm is given by a function $h\colon \mathbb{G} \to \mathbb{G}$. To prove the first claim, let $x, y$ in ${}^{\mathrm{co}}\mathcal{G}$ with $\frac{1}{4} \le y$, $|x| \le y$ and $0 \le x$. Then clearly $\frac{-y}{2} \in {}^{\mathrm{co}}\mathcal{G}$, and $\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}\mathcal{G}$ by Theorem 2.6. We have the same estimate (2.3) as in the proof of Lemma 2.4; hence we can apply Lemma 2.10 twice and obtain $4\frac{x+\frac{-y}{2}}{2} \in {}^{\mathrm{co}}\mathcal{G}$. The proof of the second claim is similar. □

The computational content are functions $\mathrm{AuxL}, \mathrm{AuxR}\colon \mathbb{G} \to \mathbb{G} \to \mathbb{G}$ with

$$\begin{aligned}\mathrm{AuxL}(u, u') &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(-u')))),\\ \mathrm{AuxR}(u, u') &:= \mathrm{Double}(\mathrm{Double}(\mathrm{Av}(u, h(u')))).\end{aligned}$$

Parallel to the essential part of Theorem 2.5 we can now prove that division by $y$ satisfies to closure axiom for ${}^{\mathrm{co}}I$:

**Theorem 2.12.** *For $x, y$ in ${}^{\mathrm{co}}G$ with $\frac{1}{4} \le y$ and $|x| \le y$ we find a signed digit $d$ such that $2x = x' + yd$ for some $x' \in {}^{\mathrm{co}}G$ with $|x'| \le y$:*

$$\forall_{x,y\in{}^{\mathrm{co}}G}\Big(\frac{1}{4} \le y \to |x| \le y \to \exists_{d\in\mathrm{Sd}}\exists_{x'\in{}^{\mathrm{co}}G}\Big(|x'| \le y \wedge \frac{x}{y} = \frac{\frac{x'}{y} + d}{2}\Big)\Big).$$

*Proof.* We proceed as for Theorem 2.5, now using Lemma 2.11. The proof uses a trifold application of the coinduction axioms for ${}^{\mathrm{co}}G$ and ${}^{\mathrm{co}}H$ to obtain the first three digits. □

The computational content is a function $f\colon \mathbb{G} \to \mathbb{G} \to \mathbb{D} \times \mathbb{G}$ defined by

$$\begin{aligned}
f(\mathrm{Lr}_{\mathtt{tt}}(u), u') &= (1, \mathrm{AuxR}(\mathrm{Lr}_{\mathtt{tt}}(u), u'))\\
f(\mathrm{Lr}_{\mathtt{ff}}(u), u') &= (-1, \mathrm{AuxL}(\mathrm{Lr}_{\mathtt{ff}}(u), u'))\\
f(\mathrm{U}(v), \mathrm{Fin}_{\mathtt{tt}}(u)) &= (1, \mathrm{AuxR}(\mathrm{U}(v), \mathrm{Fin}_{\mathtt{tt}}(u)))\\
f(\mathrm{U}(v), \mathrm{Fin}_{\mathtt{ff}}(u)) &= (-1, \mathrm{AuxL}(\mathrm{U}(v), \mathrm{Fin}_{\mathtt{ff}}(u)))\\
f(\mathrm{U}(v), \mathrm{D}(\mathrm{Fin}_{\mathtt{tt}}(u))) &= (1, \mathrm{AuxR}(\mathrm{U}(v), \mathrm{D}(\mathrm{Fin}_{\mathtt{tt}}(u))))\\
f(\mathrm{U}(v), \mathrm{D}(\mathrm{Fin}_{\mathtt{ff}}(u))) &= (-1, \mathrm{AuxL}(\mathrm{U}(v), \mathrm{D}(\mathrm{Fin}_{\mathtt{ff}}(u))))\\
f(\mathrm{U}(v), \mathrm{D}(\mathrm{D}(v'))) &= (0, \mathrm{D}(\mathrm{U}(v)))
\end{aligned}$$

**Corollary 2.13.** *For $x, y$ in ${}^{\mathrm{co}}G$ with $\frac{1}{4} \le y$ and $|x| \le y$ we have $\frac{x}{y}$ in ${}^{\mathrm{co}}G$:*

$$\forall_{x,y\in{}^{\mathrm{co}}G}\Big(\frac{1}{4} \le y \to |x| \le y \to \frac{x}{y} \in {}^{\mathrm{co}}G\Big).$$

*Proof.* By coinduction, simultaneously with the same formula where ${}^{\mathrm{co}}G$ is replaced by ${}^{\mathrm{co}}H$. Theorem 2.12 is used in both cases of the simultaneous coinduction. The extracted term could be analyzed in a similar way as in the remark after Theorem 2.5. □

2.3. **Translation to Haskell.** The terms extracted from Theorem 2.5 and Corollary 2.13 can be translated into Scheme or Haskell programs. Because of the presence of corecursion operators in the extracted terms the lazy evaluation of Haskell is more appropriate. As tests we have run (in `ghci` with time measuring by `:set +s`) the signed digit division files[3] on approximations of $\frac{1}{3}$ and $\frac{1}{2}$. To return the first 19 digits of the result of dividing $\frac{1001}{3001}$ by $\frac{10001}{20001}$ took about 0.04 seconds in the signed digit case and about 0.06 seconds in the Gray case.

In the following table we see how the runtime increases if we increase the number of signed digits in the output. Of course, this depends on the used computer, but instead of the concrete numbers we are interested in the scale of the runtime.

In the remark after Theorem 2.5 we have shown that the look-ahead of the input is linear in the digits of the output, i.e., we need at most the first $3n$ entries of $u$ and $v$ to compute the first $n$ entries of $\mathrm{Div}(u, v)$. But here we see that the runtime is clearly not linear. This is not surprising because in this remark we had the representation

$$d(u) :: d(G(u, v)) :: d(G(G(u, v), v)) :: d(G(G(G(u, v), v), v), v) \ldots$$

[3] `sddiv.scm` and `graydiv.scm` residing in `minlog/examples/analysis`

| number of digits | runtime in seconds |
|---|---|
| 10 | 0.01 |
| 25 | 0.05 |
| 50 | 0.14 |
| 75 | 0.26 |
| 100 | 0.46 |
| 250 | 2.69 |
| 500 | 10.11 |
| 750 | 23.90 |
| 1000 | 42.50 |
| 2000 | 182.92 |
| 10000 | 4567.74 |

Figure 6: Runtime

of $\mathrm{Div}(u,v)$. Therefore, for the first $n$ digits, we have to compute

$$G(u,v), G(G(u,v),v), \ldots, \underbrace{G(\ldots G}_{n-1}(u,v)\ldots)$$

and we have to read the first $3n$ digits of $u$ and $v$ and operate on them $n-1$ times. We see that $n$ occurs twice in the calculation and hence the runtime has to be at least quadratic in the numbers of digits of the output. We also see this in the table above:

If we compare, for example, the runtimes for 100, 1000 and 10000 digits, we see that a multiplication of the numbers of digits by 10 causes a multiplication of the runtime by approximately 100. Therefore the runtime seems to be approximately quadratic in the number of computed digits.

## 3. Soundness

We have extracted terms from the two proofs of Theorem 2.5 and Corollary 2.13 and tested them on numerical examples. Now we want to prove that these terms are correct, in the sense that they "satisfy their specification". We interpret this statement in the sense of Kolmogorov [Kol32]: a (c.r.) formula should be viewed as a problem asking for a solution. But what is a solution of a formula/problem $A$? We use Kreisel's notion of (modified) realizability and understand "the term $t$ is a solution of $A$" as $t\ \mathbf{r}\ A$ ($t$ realizes $A$). This definition is reviewed in Section 3.1. Then we go on and prove in Section 3.2 that for every proof $M$ of a c.r. formula $A$ its extracted term $\mathrm{et}(M)$ realizes $A$, i.e., $\mathrm{et}(M)\ \mathbf{r}\ A$. In this proof we make use of "invariance axioms"[4] stating that every c.r. formula is invariant under realizability, formally $A \leftrightarrow \exists_z(z\ \mathbf{r}\ A)$. Finally in Section 3.3 we report on a Minlog tool automatically generating such soundness proofs.

[4]They are called (A-r) "to assert is to realize" in [Fef79].

3.1. **Realizers.** Recall that computational content arises from (co)inductive predicates only. Therefore we begin with a definition of what it means to be a realizer of such a predicate. We restrict ourselves to define realizers for special instances only; a general treatment can be found in [SW12, p.334].

Consider the definition of the inductive predicate $I_0$ in Section 1.4. By another (n.c.) inductive predicate $I_0^{\mathbf{r}}$ of arity $(\mathbb{R},\mathbb{L})$ we can express that a list $u$ witnesses ("realizes") that the real $x$ is in $I_0$. We write $u\ \mathbf{r}\ I_0 x$ ($u$ is a realizer of $x \in I_0$) for $(x,u) \in I_0^{\mathbf{r}}$. The predicate $I_0^{\mathbf{r}}$ is required to be non-computational, since in $(x,u) \in I_0^{\mathbf{r}}$ we already have a realizer $u$. $I_0^{\mathbf{r}}$ is inductively defined by the two clauses

$$(0,[]) \in I_0^{\mathbf{r}}, \quad \forall_{d\in \mathrm{Sd}} \forall_{(x,u)\in I_0^{\mathbf{r}}} \Big( \Big( \frac{x+d}{2}, s_d :: u \Big) \in I_0^{\mathbf{r}} \Big)$$

and the induction axiom

$$(0,[]) \in Q \to \forall_{d\in \mathrm{Sd}} \forall_{x\in I_0^{\mathbf{r}}\cap Q} \Big( \Big( \frac{x+d}{2}, s_d :: u \Big) \in Q \Big) \to I_0^{\mathbf{r}} \subseteq Q.$$

where $s_d$ is the signed digit corresponding to $d \in \mathrm{Sd}$ [SW12, p.334]). We write $u\ \mathbf{r}\ I_0 x$ ($u$ is a realizer of $x \in I_0$) for $(x,u) \in I_0^{\mathbf{r}}$. The predicate $I_0^{\mathbf{r}}$ is required to be non-computational, since in $(x,u) \in I_0^{\mathbf{r}}$ we already have a realizer $u$. $I_0^{\mathbf{r}}$ is inductively defined by the two clauses

$$(0,[]) \in I_0^{\mathbf{r}}, \quad \forall_{d\in \mathrm{Sd}} \forall_{(x,u)\in I_0^{\mathbf{r}}} \Big( \Big( \frac{x+d}{2}, s_d :: u \Big) \in I_0^{\mathbf{r}} \Big)$$

and the induction axiom

$$(0,[]) \in Q \to \forall_{d\in \mathrm{Sd}} \forall_{x\in I_0^{\mathbf{r}}\cap Q} \Big( \Big( \frac{x+d}{2}, s_d :: u \Big) \in Q \Big) \to I_0^{\mathbf{r}} \subseteq Q.$$

where $s_d$ is the signed digit corresponding to $d \in \mathrm{Sd}$. Similarly we coinductively define the n.c. predicate $({}^{\mathrm{co}}I_0)^{\mathbf{r}}$ of arity $(\mathbb{R},\mathbb{L})$ to express that a list $u$ witnesses ("realizes") that the real $x$ is in ${}^{\mathrm{co}}I_0$. We write $u\ \mathbf{r}\ {}^{\mathrm{co}}I_0 x$ ($u$ is a realizer of $x \in {}^{\mathrm{co}}I_0$) for $(x,u) \in {}^{\mathrm{co}}I_0^{\mathbf{r}}$. The closure axiom is

$$\forall_{(x,u)\in({}^{\mathrm{co}}I_0)^{\mathbf{r}}} \Big( (x{=}0 \land u{=}[]) \lor \exists_{d\in \mathrm{Sd}} \exists_{(x',u')\in({}^{\mathrm{co}}I_0)^{\mathbf{r}}} \Big( x = \frac{x'+d}{2} \land u = s_d :: u' \Big) \Big)$$

and the coinduction axiom

$$\forall_{(x,u)\in Q} \Big( (x = 0 \land u = []) \lor \exists_{d\in \mathrm{Sd}} \exists_{(x',u')\in({}^{\mathrm{co}}I_0)^{\mathbf{r}}\cup Q} \Big( x = \frac{x'+d}{2} \land u = s_d :: u' \Big) \Big) \to Q \subseteq ({}^{\mathrm{co}}I_0)^{\mathbf{r}}.$$

3.2. **Soundness theorem.** A formula or predicate is called **r**-free if it does not contain any of the $I^{\mathbf{r}}$- or ${}^{\mathrm{co}}I^{\mathbf{r}}$-predicates. A proof $M$ is called **r**-free if it contains **r**-free formulas only.

**Theorem 3.1** (Soundness)**.** *Let $M$ be an* **r***-free proof of a formula $A$ from assumptions $u_i \colon C_i$ $(i < n)$. Then we can derive*

$$\begin{cases} \mathrm{et}(M)\ \mathbf{r}\ A & \text{if } A \text{ is c.r.} \\ A & \text{if } A \text{ is n.c.} \end{cases}$$

*from assumptions*

$$\begin{cases} z_{u_i}\ \mathbf{r}\ C_i & \text{if } C_i \text{ is c.r.} \\ C_i & \text{if } C_i \text{ is n.c.} \end{cases}$$

*Proof.* By induction on $M$. In the base case we have to prove that the extracted terms of $I^{\pm}$, ${}^{\mathrm{co}}I^{\pm}$ realize the respective axioms. The proofs in [SW12, Sections 7.2.8 and 7.2.10] can be adapted to the present situation. The step cases are easier; we only consider the ones where invariance axioms are used.

*Case* $(\lambda_{u^A} M^B)^{A\to B}$ with $A$ c.r. and $B$ n.c. We need a derivation of $A \to B$. By induction hypothesis we have a derivation of $B$ from $z\ \mathbf{r}\ A$. Using the invariance axiom $A \to \exists_z(z\ \mathbf{r}\ A)$ we obtain the required derivation of $B$ from $A$ as follows.

$$\dfrac{\dfrac{A \to \exists_z(z\ \mathbf{r}\ A) \qquad A}{\exists_z(z\ \mathbf{r}\ A)} \qquad \begin{array}{c}[z\ \mathbf{r}\ A]\\ \mid \mathrm{IH}\\ B\end{array}}{B}\ \exists^-$$

*Case* $(M^{A\to B} N^A)^B$ with $A$ c.r. and $B$ n.c. The goal is to find a derivation of $B$. By induction hypothesis we have derivations of $A \to B$ and of $\mathrm{et}(N)\ \mathbf{r}\ A$. Now using the invariance axiom $\forall_z(z\ \mathbf{r}\ A \to A)$ we obtain the required derivation of $B$ by $\to^-$ from the derivation of $A \to B$ and

$$\dfrac{\dfrac{\forall_z(z\ \mathbf{r}\ A \to A) \qquad \mathrm{et}(N)}{\mathrm{et}(N)\ \mathbf{r}\ A \to A} \qquad \begin{array}{c}\\ \mid \mathrm{IH}\\ \mathrm{et}(N)\ \mathbf{r}\ A\end{array}}{A}$$

□

3.3. **Formal soundness proofs for real division algorithms.** In the files `sddiv.scm`, `graydiv.scm` the Minlog command `add-sound` has been applied repeatedly to c.r. theorems to automatically generate the corresponding soundness proofs.

## 4. Conclusion

Recall that our goal was the extraction of computational content from proofs in constructive analysis. Although these proofs work with some standard representation of real numbers, the technique used here makes it possible to extract from such proofs terms describing algorithms which operate on different representations of real numbers, for instance streams of signed digits or Gray code. Moreover, formal proofs of their correctness can be generated automatically.

As future work it is planned to extend the present approach to problems involving e.g. the exponential function and more generally power series. A promising application area would be Euler's existence proof of solutions for ordinary differential equations satisfying a Lipschitz condition.

## Acknowledgment

We thank Tatsuji Kawai, Nils Köpp, Kenji Miyamoto, Hideki Tsuiki and three anonymous referees for helpful comments.

## References


[Ber93] Ulrich Berger. Program extraction from normalization proofs. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 91–106. Springer Verlag, Berlin, Heidelberg, New York, 1993.

[Ber09] Ulrich Berger. From coinductive proofs to exact real arithmetic. In E. Grädel and R. Kahle, editors, *Computer Science Logic*, volume 5771 of *LNCS*, pages 132–146. Springer Verlag, Berlin, Heidelberg, New York, 2009.

[Bis67] Errett Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, New York, 1967.

[BMST16] Ulrich Berger, Kenji Miyamoto, Helmut Schwichtenberg, and Hideki Tsuiki. Logic for Gray-code computation. In D. Probst and P. Schuster, editors, *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 69–110. De Gruyter, 2016.

[CG06] Alberto Ciaffaglione and Pietro Di Gianantonio. A certified, corecursive implementation of exact real numbers. *Theoretical Computer Science*, 351:39–51, 2006.

[Ers77] Yuri L. Ershov. Model $C$ of partial continuous functionals. In R. Gandy and M. Hyland, editors, *Logic Colloquium 1976*, pages 455–467. North-Holland, Amsterdam, 1977.

[Fef79] Solomon Feferman. Constructive theories of functions and classes. In K. McAloon M. Boffa, D. van Dalen, editor, *Logic Colloquium 78*, volume 97 of *Studies in Logic and the Foundations of Mathematics*, pages 159–224. North-Holland, Amsterdam, 1979.

[Gia99] Pietro Di Gianantonio. An abstract data type for real numbers. *Theoretical Computer Science*, 221(1-2):295–326, 1999.

[Kol32] Andrey N. Kolmogorov. Zur Deutung der intuitionistischen Logik. *Math. Zeitschr.*, 35:58–65, 1932.

[MS15] Kenji Miyamoto and Helmut Schwichtenberg. Program extraction in exact real arithmetic. *Mathematical Structures in Computer Science*, 25:1692–1704, 2015.

[Sco70] Dana Scott. Outline of a mathematical theory of computation. Technical Monograph PRG–2, Oxford University Computing Laboratory, 1970.

[SW12] Helmut Schwichtenberg and Stanley S. Wainer. *Proofs and Computations*. Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.

[Tsu02] Hideki Tsuiki. Real number computation through Gray code embedding. *Theoretical Computer Science*, 284:467–485, 2002.

[Wie17] Franziskus Wiesnet. Konstruktive Analysis mit exakten reellen Zahlen. Master's thesis, Mathematisches Institut der Universität München, 2017.

[Wie18] Franziskus Wiesnet. Introduction to Minlog. In K. Mainzer, P. Schuster, and H. Schwichtenberg, editors, *Proof and Computation*, pages 233–288. World Scientific, 2018.