Lookahead analysis in exact real arithmetic with logical methods

Nils Kpp

Ludwig-Maximilians Universitt, Theresienstr. 39, 80333 Mnchen

Helmut Schwichtenberg

Ludwig-Maximilians Universitt, Theresienstr. 39, 80333 Mnchen

Abstract

Program extraction from proofs can be used to obtain verified algorithms in exact real arithmetic for e.g., the signed-digit code representation. In Minlog this has been done in the past with the use of certain coinductive predicates. In a next step we want to analyze the lookahead of these extracted programs. Doing it by hand is quite cumbersome, so instead we change our definitions. Instead of a coinductive predicate we use an inductive predicate for the representation of reals that already incorporates the lookahead. In this way the lookahead becomes part of the specification which is carried through all the proofs. In the end we extract programs for computations on a signed-digit representation and we can just read off the lookahead from the specification.

Keywords: signed digit code, exact real number computation, lookahead , program extraction, realizability, Minlog

Preprint submitted to Theoretical Computer Science *July 29, 2022*

Lookahead analysis in exact real arithmetic with logical methods

Nils Kpp

Ludwig-Maximilians Universitt, Theresienstr. 39, 80333 Mnchen

Helmut Schwichtenberg

Ludwig-Maximilians Universitt, Theresienstr. 39, 80333 Mnchen

1. Introduction

We continue the work from [1], [2] and [3], where a *coinductive* predicate ${}^{co}\mathbf{I}$ was used to represent exact real numbers, i.e., ${}^{co}\mathbf{I}x \Leftrightarrow$ *x has a signed-digit-code representation.* This technique is based on an approach from [4], although here we use explicitely defined real numbers instead of axioms for abstract real numbers. From proofs that this predicate ${}^{co}\mathbf{I}$ is closed under average, multiplication and division, algorithms for exact real number arithmetic are extracted and translated to Haskell. Furthermore a proof of correctness can be automatically generated. In this paper we refine this method in order to analyze the lookahead of these algorithms, namely we want to track how many input digits are needed to compute a certain number of output digits. To that end we use a new *inductive* predicate $\mathbf{L}$ that already contains the information regarding the lookahead. In Section 1.3 we give a quick overview of the definitions and main theorems from [1] and [2]. In Section 1.4 we give our new definition and prove some basic properties. Then in Section 2 we use this method to get explicit bounds for the lookahead in exact real arithmetic.

1.1. Theory of computable functionals

We use the formal theory of computable functionals *TCF* [5] to formalize statements like "there is an arbitrarily exact representation of x by a list of signed digits". *TCF* is given by a language that consists of the following.

- *Types* consisting of variables, arrow-types and algebras directly given by *constructors.*
- *Terms* given by the algebra-constructors, lambda-abstraction, application and constants defined by computation rules.
- *Predicates* P, X which are either *constants*, *variables* or *comprehension terms* $\{\vec{x} \mid A\}$ of a fixed arity.

- *Formulae* of the form $P\vec{t}$, $A \to B$ and $\forall_x A$.

The logic is *natural deduction* together with a set of axioms for the predicate constants given below. Note that predicates are marked as either *computationally relevant* (c.r.) or *non-computational* (n.c.). Furthermore, for an unary predicate A, we write $t \in A$ for At and $\forall_{t \in A} B$, $\exists_{t \in A} B$ are short for $\forall_t (At \to B)$ and $\exists_t (At \wedge B)$, respectively.

Predicates

The constants are either *inductive* or *coinductive* predicates. An inductive predicate I is given by a list of introductory axioms

$$(I)_i^+ : \forall_{\vec{x}} (\vec{A}_i(I) \to I\vec{t}_i) \qquad (i < k),$$

where X must only appear *strictly positive* in all $A_{ij}(X)$. Furthermore we add a *least-fixed-point-axiom* or *induction-axiom* for I of the form

$$(I)^- : I\vec{x} \to \Big(\forall_{\vec{x}} (\vec{A}_i(I \cap X) \to X\vec{t}_i) \Big)_{i<k} \to X\vec{x}.$$

The logical connectives $\exists, \wedge$ and $\vee$ are all special cases of inductive predicates. A *coinductive predicate* J is introduced by giving one *closure-axiom* of the form

$$(J)^- : \forall_{\vec{x}} (J\vec{x} \to \bigvee_{i<k} \exists_{\vec{y}_i} B_i(J)).$$

Furthermore we add a *greatest-fixed-point-axiom*

$$(J)^+ : X\vec{x} \to \forall_{\vec{x}} (X\vec{x} \to \bigvee_{i<k} \exists_{\vec{y}_i} B_i(J \cup X)) \to J\vec{x}.$$

For any inductive predicate I its *companion prediacte* ${}^{co}I$ is given by setting $B_i = \bigwedge \vec{A}_i$. Examples for inductive predicates are the *totality* predicates. For some type τ the expression $t \in \tau$ is short for $t \in \mathbf{T}_\tau$, where $\mathbf{T}_\tau$ is the totality-predicate for this type, e.g., for a term $ns \colon \mathbb{N} \to \mathbb{N}$ we have $ns \in (\mathbb{N} \to \mathbb{N}) := \forall_{n \in \mathbf{T}_\mathbb{N}} (ns\ n) \in \mathbf{T}_\mathbb{N}$ where $n \in \mathbf{T}_\mathbb{N}$ is the inductive predicate given by the clauses $0 \in \mathbf{T}_\mathbb{N}$ and $n \in \mathbf{T}_\mathbb{N} \to (n+1) \in \mathbf{T}_\mathbb{N}$. The elimination rule of $n \in \mathbb{N}$ is induction over natural numbers.

Realizability and Computational Content

We inductively assign to each c.r. predicate P and formula A a type τ and a predicate $P^{\mathbf{r}}$ of arity $arity(P) \times \tau$ respectively $A^{\mathbf{r}}$ of arity τ.

$$\begin{aligned}
&\tau(I) := \iota_I \\
&\tau(A \to B) := \begin{cases} \tau(A) \to \tau(B), & A\ c.r. \\ \tau(B), & A\ n.c. \end{cases} \\
&\tau(\forall_x A) := \tau(A) \\
&x\mathbf{r}I\vec{t} := I^{\mathbf{r}}\vec{t}x \\
&f\mathbf{r}(A \to B) := \begin{cases} \forall_x(x\mathbf{r}A \to (f\,x)\mathbf{r}B), & A\ c.r. \\ A \to f\mathbf{r}B, & A\ n.c. \end{cases} \\
&x\mathbf{r}\forall_y A := \forall_y x\mathbf{r}A
\end{aligned}$$

Furthermore to each derivation of a c.r. formula we assign its *extracted term.* The interesting cases are the axioms. For an inductive predicate I given as above they are

$$\begin{aligned}
&et((I)_i^+) := \mathtt{C}_i^{\iota_I}, \\
&et((I)^-(P)) := \mathcal{R}_{\iota_I}^{\tau}(P), \\
&et((^{co}I)^-) := \mathtt{D}, \\
&et((^{co}I)^+(P)) := {}^{co}\mathcal{R}^{\iota_I}_{\tau(P)},
\end{aligned}$$

where $\mathcal{R}_{\iota_I}$ and ${}^{co}\mathcal{R}^{\iota_I}$ are the *recursion* and *corecursion* operators associated to the algebra ι_I and $\mathtt{C}_i$ and $\mathtt{D}$ are its constructors and destructor respectively. Note that a non-uniform version of the $\forall$-quantifier can be recovered by abbreviations as above, e.g., for A c.r. $\tau(\forall_{n\in\mathbb{N}}A) := \mathbb{N} \to \tau(A)$ and $f\mathbf{r}\forall_{n\in\mathbb{N}}A := \forall_{n,m}(m\mathbf{r}(n \in \mathbb{N}) \to (f\,m)\mathbf{r}A)$.

Correctness and implementation in Minlog

The foundation of program extraction is the *soundness-theorem*, namely given a proof M of a formula A we also have a proof of $et(M)\mathbf{r}A$. The proof is by induction on derivations. In Minlog this proof of correctness can be automatically generated for every proof.
In *TCF* all variables are typed. The following table shows which variables have which type.

$m, n : \mathbb{N}$	$a, b : \mathbb{Q}$	$M, N : \mathbb{Z}^+ \to \mathbb{N}$
$d, e, k : \mathbb{Z}$	$x, y : \mathbb{R}$	$as, bs : \mathbb{N} \to \mathbb{Q}$

Here $\mathbb{N}$ are the unary natural numbers, $\mathbb{Z}$ is defined as positive binary numbers and $\mathbb{Q}$ is given by one constructor $\# : \mathbb{Z} \to \mathbb{Z}^+ \to \mathbb{Q}$. In the implementation in Minlog the type of real numbers $\mathbb{R}$ is explicitly defined as the type $(\mathbb{N} \to \mathbb{Q}) \times (\mathbb{Z}^+ \to \mathbb{N})$. For computing the extraced terms and verifying the correctness of the proofs, the proof assistant Minlog [6] is used. An introduction to Minlog can be found in [7] or `doc/tutor.pdf` in the Minlog directory. All the proofs in this paper have been formalized in Minlog. Minlog has the capability to translate terms into Haskell. After each proof in the following we state its computational content.

1.2. Cauchy reals

Formally *real numbers* are defined as pairs of a sequences of rational numbers together with a modulus, roughly as presented in [8].

Definition 1 (Cauchy representation)**.** We define the algebra $\mathbf{R}$ by the single constructor

$$\texttt{RealConstr} : (\mathbb{N} \to \mathbb{Q}) \to (\mathbb{Z}^+ \to \mathbb{N}) \to \mathbb{R}.$$

For $as : \mathbb{N} \to \mathbb{Q}$ and $M : \mathbb{Z}^+ \to \mathbb{N}$ we define

$$\mathbf{Mon}(M) := M \in \mathbf{T}^{nc} \wedge \forall_{p \leq q} \left(Mp \leq Mq\right).$$

Next we define the predicate $\mathbf{R}$, $x = \texttt{RealConstr}\ as\ M \in \mathbf{R}$ by

$$M \in \mathbf{Mon} \wedge as \in \mathbf{T}^{nc} \wedge \forall_{p \in \mathbf{T}^{nc}} \forall_{n,m \geq M(p)} \left(|(as\ n) - (as\ m)| < 2^{-p} \right).$$

In the following $\forall_x A$ will always be an abbreviation for $\forall_x (x \in \mathbf{R} \to A)$ respectively $\forall_{x \in B} A$ abbreviates $\forall_x (x \in \mathbf{R} \to B \to A)$. For the type of $\mathbb{R}$ together with the predicate $\mathbf{R}$ we can define all the usual operation for the real numbers and prove all the properties of a field that constructively hold. For details we refer to [9] and the implementation in Minlog which can be found in the folder `.../git/minlog/lib/rea.scm`. Note that $x \in \mathbf{R}$ by definition does not carry any computational content. Hence the Cauchy-representation of the reals is only used for the verificiation.

1.3. Real numbers represented by streams

A signed-digit-code representation of some real number x is a sequence $(d_i)_i \in \{-1, 0, 1\}^{\mathbb{N}}$ such that

$$x = \sum_{i=1}^{\infty} d_i 2^{-i}.$$

Definition 2 (sd-code representation)**.** We define ${}^{co}\mathbf{I}$ as the greatest predicate satisfying the single clause

$$d \in \mathbf{Sd} \to x \in {}^{co}\mathbf{I} \to x = \frac{y + d}{2} \to y \in {}^{co}\mathbf{I}$$

where $\mathbf{Sd} := \{-1, 0, 1\} \subseteq \mathbb{Z}$ is an inductive predicate.

Remark 1. *Henceforth we will use* ***Sd*** *for the predicate as well as the algebra given by three nullary constructors. A realiser of* ${}^{co}\boldsymbol{I}x$ *is exactly a stream of signed digits. The algebra of streams of signed digits* $\mathbb{S}$ *is given by the single constructor*

$$\mathtt{C} : \mathbf{Sd} \to \mathbb{S} \to \mathbb{S}.$$

The axioms of ${}^{co}\mathbf{I}$ *can be expressed as*

$${}^{co}\mathbf{I}^{-} : \ x \in {}^{co}\mathbf{I} \to \exists_{d \in \mathbf{Sd}, y \in {}^{co}\mathbf{I}}\, x = \frac{y+d}{2}$$

$${}^{co}\mathbf{I}^{+} : \ \forall_{x \in X} \exists_{d \in \mathbf{Sd}} \exists_{y \in X \cup {}^{co}\mathbf{I}}\, x = \frac{y+d}{2} \to X \subseteq {}^{co}\mathbf{I}.$$

Theorem 1 (`CoIAverage`)**.** *For all* $x, y \in {}^{co}\mathbf{I}$

$$\frac{x+y}{2} \in {}^{co}\mathbf{I}$$

Proof. The proof is done by first showing that

$$\left\{ \frac{x+y}{2} \,\middle|\, x, y \in {}^{co}\mathbf{I} \right\} \subseteq \left\{ \frac{x+y+i}{4} \,\middle|\, x, y \in {}^{co}\mathbf{I}, i \in \mathbf{Sd}_2 \right\},$$

where $\mathbf{Sd}_2 := \{-2, -1, 0, 1, 2\}$. Then we show that the second set also satisfies the clause of ${}^{co}\mathbf{I}$. The result follows from the coinduction axiom. □

Theorem 2 (`CoIMult`)**.** *For all* $x, y \in {}^{co}\mathbf{I}$

$$xy \in {}^{co}\mathbf{I}$$

Proof. The idea is similar to the previous proof. First we show

$$\{xy \,|\, x, y \in {}^{co}\mathbf{I}\} \subseteq \left\{ \frac{xy+z+i}{4} \,|\, x, y, z \in {}^{co}\mathbf{I}, i \in \mathbf{Sd}_2 \right\},$$

and then, that the second set fulfills the clause. Coinduction yields the claim. □

Theorem 3 (`CoIDiv`)**.** *For all* $x, y \in {}^{co}\mathbf{I}$ *with* $\frac{1}{4} \leq y$ *and* $|x| \leq y$

$$\frac{x}{y} \in {}^{co}\mathbf{I}$$

Proof. By coinduction we prove

$$\left\{ \frac{x}{y} \,\middle|\, x, y \in {}^{co}\mathbf{I}, |x| \leq y, \frac{1}{4} \leq y \right\} \subseteq {}^{co}\mathbf{I}.$$ □

1.4. Real numbers represented by lists

We want to analyze the lookahead of the algorithms we extracted from the proofs. Going through the proofs or algorithms and tracking the lookahead by hand can be done, albeit it is quite cumbersome. For that reason we want to track the lookahead directly on the logical level as part of the specification. To this end we define a new *inductive* predicate

$$\mathbf{L} \subseteq \mathbb{R} \times \mathbb{N}$$

with the intended meaning

$$\mathbf{L}(x, n) \Leftrightarrow \text{We know the first } n \text{ digits of } x.$$

Equivalently $\mathbf{L}(x, n)$ should mean that we have a $\frac{1}{2^n}$ approximation of x, i.e., we have n signed digits $d_1 \dots d_n$ such that $\left|x - \sum_{i=1}^{n} \frac{d_i}{2^i}\right| \leq \frac{1}{2^n}$. The formal definition is motivated by the following property that $\mathbf{L}$ should have.

$$\mathbf{L}(x, n+1) \to x = \frac{y+d}{2} \to \mathbf{L}(y, n),$$

i.e., if we know the first $n+1$ digits of x and d is the first digit, then we know the first n digits of $2x-d$. We redo all the proofs previously done with a coinductive predicate, and from a proof of e.g.,

$$\mathbf{L}(x, n+1) \to \mathbf{L}(y, n+1) \to \mathbf{L}(\frac{x+y}{2}, n)$$

we extract a term of type $\mathbb{L} \to \mathbb{L} \to \mathbb{L}$ that computes the first n digits of $\frac{x+y}{2}$, given $n+1$ digits of x and y. Note that we do not loose anything compared to Theorem 1, we just added some control regarding the precision of the input.

Definition 3. We define $\mathbf{L}$ as the least predicate satisfying the clauses

$$\forall_{x,n}(|x| \leq 1 \to \mathbf{L}(x, 0))$$
$$\forall_{x,n,d\in\mathbb{D}}(\mathbf{L}(x, n+1) \to x = \frac{y+d}{2} \to \mathbf{L}(y, n)).$$

Note that the algebra of realizers of $\mathbf{L}$, namely $\mathbb{L}$ is now given by two constructors

$$\mathtt{U} : \mathbb{L}$$
$$\mathtt{C} : \mathbf{Sd} \to \mathbb{L} \to \mathbb{L},$$

where $\mathtt{U}$ is the empty list. The computational content of the two introductory axioms are the two constructors respectively. Note that due to the nullary clause of $\mathbf{L}$ is it unnecessary to use a coniductive predicate, since realizers of $\mathbf{L}xn$ will be total (as long as n is) , i.e., finite lists, in any case. Note that apart from choosing the right natural number, the proofs are very similar to the original proofs. Coinduction will be replaced by an induction over the length of the realizing list. In the following we will use the variable names $u, v : \mathbb{L}$.

Remark 2. *The elimination axiom of* $\mathbf{L}$ *has the following form. Assume* $P \subseteq \mathbb{R} \times \mathbb{N}$*. Then if*

$$\forall_x (|x| \leq 1 \to Px0)$$
$$\forall_{d \in \mathbb{D}, x, y, n} (\mathbf{L}xn \to Pxn \to y = \frac{x+d}{2} \to Py(n+1))$$

we can infer $\mathbf{L} \subseteq P$*. In the following we will use the notation*

$$\mathbf{L}xn := (x \in \mathbf{L}_n)$$

If P *has type* τ *then the computational content of the elimination axiom* $(\mathbf{L})^-[P]$ *is given by the recursion operator* $\mathcal{R}^\tau_{\mathbb{L}} : \mathbb{L} \to \tau \to (\mathbf{Sd} \to \mathbb{L} \to \tau \to \tau) \to \tau$ *with the computation rules*

$$\mathcal{R}^\tau_{\mathbb{L}}(\mathtt{U}, t_0, f) = t_0$$
$$\mathcal{R}^\tau_{\mathbb{L}}(\mathtt{C}dv, t_0, f) = f(d, v, \mathcal{R}^\tau_{\mathbb{L}}(v, t_0, f)),$$

i.e., recursion for lists.

1.5. Basic properties

We prove some basic properties.

Lemma 1 (`LSuccToL`)**.**

$$\forall_{x,n} (x \in \mathbf{L}_{n+1} \to x \in \mathbf{L}_n).$$

Proof. We can immediately prove $\forall_{m,x,n}(x \in \mathbf{L}_m \to m = n+1 \to x \in \mathbf{L}_n)$ with the elemination axiom of $\mathbf{L}$. □

Extracted Term (`cLSuccToL`)**.**

$$\mathtt{cLSuccToL}(d :: u) := u.$$

Lemma 2 (`LToLPred`)**.**

$$\forall_{x,n} (x \in \mathbf{L}_n \to x \in \mathbf{L}_{n \dot{-} 1}).$$

Proof. Immediately by the elimination axiom of $\mathbf{L}$. □

Extracted Term (`cLToToLPred`)**.**

$$\mathtt{cLToLPred}(\mathtt{U}) := \mathtt{U},$$
$$\mathtt{cLToLPred}(d :: u) := u.$$

Remark 3. *In the following we will write the extracted term of the previous two lemmas as* $\mathtt{tl} \colon \mathbb{L} \to \mathbb{L}$*, i.e., the usual tail-function. Furthermore, for better readability, we will sometimes make use of the function* $\mathtt{hd} \colon \mathbb{L} \to \mathbf{Sd}$ *defined by*

$$\mathtt{hd}(\mathtt{U}) := 0,$$
$$\mathtt{hd}(d :: u) := d.$$

Lemma 3 (`LCompat`)**.**

$$\forall_{x,y,n}(x = y \to x \in \mathbf{L}_n \to y \in \mathbf{L}_n)$$

Proof. The proof is by induction on $\mathbf{L}xn$ □

Extracted Term (`cLCompat`)**.** The extracted term is $\texttt{cLCompat}(u) := u$, i.e., the identity. In the following we will omit it.

In order to adapt proofs for ${}^{co}\mathbf{I}$ to the present setting we prove that $\mathbf{L}$ satisfies a certain closure-rule.

Lemma 4 (`LClosure`)**.**

$$x \in \mathbf{L}_{n+1} \to \exists_{d\in\mathbb{D},x_0} x = \frac{x_0 + d}{2} \wedge x_0 \in \mathbf{L}_n.$$

Proof. Immediately by induction. □

Extracted Term (`cLClosure`)**.** The extracted term is of type $\mathbb{L} \to \mathbf{Sd} \times \mathbb{L}$. Let $\langle \cdot, \cdot \rangle$ denote the pair constructor, then

$$\texttt{cLClosure}(s :: v) := \langle s, v \rangle.$$

In the following this term we will be supressed by either supplying enough digits in the input or by using the `hd` function when needed.

In order to avoid long case distinctions in the following proofs we define two functions $J, K \colon \mathbb{Z} \to \mathbb{Z}$ such that the following equality holds:

$$m = K(m) + 4J(m)$$

These functions exist since we can do division with remainder. Furthermore we have the following property:

Lemma 5. *For $m \in \mathbb{Z}$ with $|m| \leq 6$ it holds that $Km \in \mathbf{Sd}_2$ and $Jm \in \mathbf{Sd}$.*

For the proofs in the next section regarding multiplication we need some more properties of $\mathbf{L}$.

Lemma 6 (`LSd`)**.**

$$\forall_{n\in\mathbb{N}}\forall_{d\in\mathbf{Sd}}\, d \in \mathbf{L}_n$$

Proof. By induction on $n \in \mathbb{N}$. □

Extracted Term (`cLSd`)**.** $\texttt{cLSd} \colon \mathbb{N} \to \mathbf{Sd} \to \mathbb{L}$

$$\texttt{cLSd}(0, d) := \texttt{U}$$
$$\texttt{cLSd}(n + 1, d) := d :: (\texttt{cLSd}(n, d))$$

Lemma 7 (`LUMinus`)**.**

$$\forall_n \forall_{x\in\mathbf{L}_n} (-x) \in \mathbf{L}_n$$

Proof. By induction on $x \in \mathbf{L}_n$. In the base case we immediately get $(-x) \in \mathbf{L}_0$. Assume $d \in \mathbb{D}, x, (-x) \in \mathbf{L}_n$ and $y = \frac{x+d}{2}$. We need to prove $(-y) \in \mathbf{L}_{n+1}$ which follows from $-y = \frac{-x-d}{2}$. □

Extracted Term (`cLUMinus`)**.** `cLUMinus`$\colon \mathbb{L} \to \mathbb{L}$ is given by recursion on $\mathbb{L}$:

$$\begin{aligned} &\texttt{cLUMinus}\,\texttt{U} := \texttt{U} \\ &\texttt{cLUMinus}(e :: v) := (-e) :: (\texttt{cLUMinus}\ v) \end{aligned}$$

Lemma 8 (`LSdTimes`)**.**

$$\forall_{n \in \mathbb{N}} \forall_{d \in \mathbb{D}} \forall_{x \in \mathbf{L}_n} (dx) \in \mathbf{L}_n$$

Proof. By induction on $d \in \mathbb{D}$. If $d = -1$ we use Lemma 7 and if $d = 0$ then Lemma 6. Otherwise we are done by compatibility. □

Extracted Term (`cLSdTimes`)**.** `cLSdTimes`$\colon \mathbb{N} \to \mathbf{Sd} \to \mathbb{L} \to \mathbb{L}$ is defined by a case-distiction on $\mathbf{Sd}$:

$$\begin{aligned} &\texttt{cLSdTimes}(n, -1, u) := \texttt{cLUMinus}\ u \\ &\texttt{cLSdTimes}(n, 0, u) := \texttt{cLSd}(n, 0) \\ &\texttt{cLSdTimes}(n, 1, u) := u \end{aligned}$$

For the proof that $\mathbf{L}$ is closed under division we will need the following two Lemmas.

Lemma 9 (`LNegToLPlusOne`,`LPosToLMinusOne`)**.** *For all $n \in \mathbb{N}$ we have*

$$\begin{aligned} &x \in \mathbf{L}_n \to x \leq 0 \to (x+1) \in \mathbf{L}_n, \\ &x \in \mathbf{L}_n \to 0 \leq x \to (x-1) \in \mathbf{L}_n. \end{aligned}$$

Proof. Both statements follow by induction on $n \in \mathbb{N}$. In the step a case-distinction on $\mathbf{Sd}$ is used. □

Extracted Term (`cLNegToPlusOne`,`cLPosToLMinusOne`)**.** The extracted terms are defined by recursion on $\mathbb{N}$ and a case-distinction on $\mathbf{Sd}$ in the step-cases:

$$\texttt{cLNegToPlusOne}(0, u) := \texttt{U}$$

$$\texttt{cLNegToPlusOne}(n+1, d :: v) := \begin{cases} \texttt{cLSd}(n+1, 1), & d = 1 \\ 1 :: \texttt{cLNegToPlusOne}(v), & d = 0 \\ 1 :: v, & d = -1 \end{cases}$$

$$\texttt{cLPosToLMinusOne}(0, u) := \texttt{U}$$

$$\texttt{cLPosToLMinusOne}(n+1, d :: v) := \begin{cases} -1 :: v, & d = 1 \\ -1 :: \texttt{cLPosToLMinusOne}(v), & d = 0 \\ \texttt{cLSd}(n+1, -1), & d = -1 \end{cases}$$

Lemma 10 (LToLDouble,LToLQuad)**.** *For all* $n \in \mathbb{N}$

$$x \in \mathbf{L}_{n+1} \to |x| \leq \frac{1}{2} \to (2x) \in \mathbf{L}_n,$$

$$x \in \mathbf{L}_{n+2} \to |x| \leq \frac{1}{4} \to (4x) \in \mathbf{L}_n.$$

Proof. For the first statement we use Lemmas 4 and 9 with a case-distinction on **Sd**. E.g. if $x = \frac{y+1}{2}$, then $x \leq \frac{1}{2}$ implies $y \leq 0$ and $2x = y + 1 \in \mathbf{L}_n$. The second follows directly from the first. □

Extracted Term (cLToLDouble,cLToLQuad)**.** The extracted terms are

$$\texttt{cLToLDouble}(n, d :: u) := \begin{cases} \texttt{cLNegToLPlusOne}(n, u), & d = 1 \\ u, & d = 0 \\ \texttt{cLPosToLMinusOne}(n, u), & d = -1 \end{cases},$$

$$\texttt{cLToLQuad}(n, u) := \texttt{cLToLDouble}(n, \texttt{cLToLDouble}(n + 1, u)).$$

2. Exact real number arithmetic with lookahead

2.1. Average, multiplication and division

Now we can go through the proofs referenced above with the new predicate $\mathbf{L}$ instead of ${}^{co}\mathbf{I}$. This will give us verified algorithms for exact real number arithmetic with the added bonus that the look-ahead of the algorithms is explicitly part of the specification.

We first show that $\mathbf{L}$ is closed under the average. The proof structure will be very similar, namely we show

$$\left\{\frac{x+y}{2} \mid x, y \in \mathbf{L}_{n+1}\right\} \subseteq \left\{\frac{x+y+i}{4} \mid x, y \in \mathbf{L}_n, i \in \mathbf{Sd}_2\right\} \subseteq \mathbf{L}_n,$$

where the second inclusion is proven by induction on $\mathbb{N}$.

Lemma 11 (LAvToAvc)**.**

$$x, y \in \mathbf{L}_{n+1} \to \exists_{i \in \mathbb{D}_2} \exists_{x_1, y_1 \in \mathbf{L}_n} \frac{x+y}{2} = \frac{x_1 + y_1 + i}{4}$$

Proof. By applying Lemma 4 to $x, y \in \mathbf{L}_{n+1}$ we get

$$\frac{x+y}{2} = \frac{x_1 + y_1 + (d + e)}{4}.$$

□

Extracted Term (cLAvToAvc)**.** $\texttt{cLAvToAvc}\colon \mathbb{L} \to \mathbb{L} \to \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L}$

$$\texttt{cLAvToAvc}(d :: u, e :: v) := \langle d + e, u, v \rangle.$$

Lemma 12 (`LAvcSatLCl`).

$$i \in \mathbf{Sd}_2 \to x, y \in \mathbf{L}_{n+1} \to \exists_{d \in \mathbb{D}} \exists_{j \in \mathbb{D}_2} \exists_{x_1, y_1 \in \mathbf{L}_n} \frac{x+y+i}{4} = \frac{\frac{x_1+y_1+j}{4}+d}{2}.$$

Proof. We use Lemma 4 to decompose $x, y \in \mathbf{L}_{n+1}$ to $d, e \in \mathbf{Sd}, x_0, y_0 \in \mathbf{L}_n$. Then we compute

$$\frac{x+y+i}{4} = \frac{x_0+y_0+(d+e+2i)}{8}.$$

With Lemma 5 we can transform this expression to

$$\frac{\frac{x_0+y_0+K(d+e+2i)}{4}+J(d+e+2i)}{2}. \qquad \square$$

Extracted Term (`cLAvcSatLCl`). `cLAvcSatLCl`: $\mathbf{Sd}_2 \to \mathbb{L} \to \mathbb{L} \to \mathbf{Sd} \times \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L}$

$$\texttt{cLAvcSatLCl}(i, d :: u, e :: v) := \langle K(d+e+2i), J(d+e+2i), u, v \rangle.$$

Lemma 13 (`LAvcToL`).

$$\forall_{n \in \mathbb{N}} \forall_{i \in \mathbb{D}_2} \forall_{x,y \in \mathbf{L}_n} \frac{x+y+i}{4} \in \mathbf{L}_n$$

Proof. By induction on $n \in \mathbb{N}$. The base-case is immediate by the first introduction axiom of $\mathbf{L}$, since

$$|x|, |y| \leq 1 \Rightarrow \left| \frac{x+y+i}{4} \right| \leq 1.$$

So assume

$$\forall_{i \in \mathbb{D}_2} \forall_{x,y \in \mathbf{L}_n} \frac{x+y+i}{4} \in \mathbf{L}_n,$$

and $x, y \in \mathbf{L}_{n+1}$. By Lemma 12 there exists $j \in \mathbb{D}_2, d \in \mathbb{D}$ and $x_1, y_1 \in \mathbf{L}_n$ with

$$\frac{x+y+i}{4} = \frac{\frac{x_1+y_1+j}{4}+d}{2}$$

By compatibility of $\mathbf{L}$ it suffices to prove

$$\frac{\frac{x_1+y_1+j}{4}+d}{2} \in \mathbf{L}_{n+1}.$$

By the second introduction axiom of $\mathbf{L}$ this follows from

$$\frac{x_1+y_1+j}{4} \in \mathbf{L}_n$$

which we have by the induction hypothesis. $\square$

Extracted Term (cLAvcToL)**.** The extracted term $\texttt{cLAvcToL}\colon \mathbb{N} \to \mathbf{Sd}_2 \to \mathbb{L} \to \mathbb{L} \to \mathbb{L}$ is defined by recursion on $\mathbb{N}$. It is given by

$$\begin{aligned} &\texttt{cLAvcToL}(0, i, u, v) := \texttt{U} \\ &\texttt{cLAvcToL}(n+1, i, u, v) := d :: \texttt{cLAvcToL}(n, w), \end{aligned}$$

where

$$\langle d, w\rangle \ = \texttt{cLAvcSatLCl}(i, u, v).$$

Theorem 4 (LAverage)**.**

$$\forall_{n\in\mathbb{N}}\forall_{x,y\in\mathbf{L}_{n+1}} \frac{x+y}{2} \in \mathbf{L}_n$$

Proof. Directly by the Lemmas 11 and 13. □

Extracted Term (cLAverage)**.** The extracted term $\texttt{cLAverage}\colon \mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbb{L}$ is defined by

$$\texttt{cLAverage}(n, u, v) := \texttt{cLAvcToL}(n+1, \texttt{cLAvToAvc}(\texttt{u}, \texttt{v})).$$

Next we will prove, that $\mathbf{L}$ is closed under multiplication. Again the structure of the proof is comparable to the coinductive case. We prove

$$\{xy \mid x, y \in \mathbf{L}_{n+3}\} \subseteq \left\{ \frac{xy+z+i}{4} \mid x, y \in \mathbf{L}_{n+2}, z \in \mathbf{L}_n, i \in \mathbf{Sd}_2 \right\},$$

and in a second step, by induction on $\mathbb{N}$,

$$\left\{ \frac{xy+z+i}{4} \mid x, z \in \mathbf{L}_{3n}, y \in \mathbf{L}_{3n \dot{-} 1}, i \in \mathbf{Sd}_2 \right\} \subseteq \mathbf{L}_n.$$

Lemma 14 (LMultToMultc)**.** *For all $n \in \mathbb{N}$ and $x, y \in \mathbf{L}_{n+3}$ we have*

$$\exists_{i\in\mathbb{D}_2}\exists_{x_1,y_1\in\mathbf{L}_{n+2}}\exists_{z_1\in\mathbf{L}_n}\left(xy = \frac{x_1y_1 + z_1 + i}{4}\right)$$

Proof. By Lemma 4 there are $d_1, e_1 \in \mathbb{D}$ and $x_1, y_1 \in \mathbf{L}_{n+2}$ such that

$$x = \frac{x_1 + d_1}{2} \qquad y = \frac{y_1 + e_1}{2}.$$

Then by Lemma 8 and Theorem 4 we also know

$$\frac{e_1x_1 + d_1y_1}{2} \in \mathbf{L}_{n+1}.$$

So by another application of Lemma 4 there exist $z_1 \in \mathbf{L}_n$ and $d_2 \in \mathbf{Sd}$ with $\frac{e_1x_1+d_1y_1}{2} = \frac{z_1+d_2}{2}$. We compute

$$xy = \frac{x_1y_1 + d_1e_1 + e_1x_1 + d_1y_1}{4} = \frac{x_1y_1 + z_1 + (d_2 + d_1e_1)}{4}.$$ □

Extracted Term (`cLMultToMultc`)**.** The extracted term `cLMultToMultc`: $\mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L} \times \mathbb{L}$ is given by

$$\texttt{cLMultToMultc}(n, d_1 :: u, e_1 :: v) := \langle d_2 + d_1 e_1, u, e_1 :: v, w \rangle,$$

where

$$d_2 :: w = \texttt{cLAverage}(\texttt{cLSdTimes}(n+2, e_1, u), \texttt{cLSdTimes}(n+2, d_1, v).$$

Lemma 15 (`LMultcSatLCl`)**.** *For all $i \in \mathbb{D}_2, n \in \mathbb{N}, m, x \in \mathbf{L}_{m+1}, y \in \mathbf{L}_{n+2}$ and $z \in \mathbf{L}_{n+3}$ we have*

$$\exists_{d \in \mathbb{D}} \exists_{j \in \mathbb{D}_2} \exists_{x_1 \in \mathbf{L}_m} \exists_{z_1 \in \mathbf{L}_n} \frac{xy + z + i}{4} = \frac{\frac{x_1 y + z_1 + j}{4} + d}{2}.$$

Proof. We decompose x, z via Lemma 4 into

$$x = \frac{x_1 + d_1}{2} \qquad z = \frac{z_0 + d_0}{2},$$

where $x_1 \in \mathbf{L}_m$ and $z_0 \in \mathbf{L}_{n+2}$. Then by Lemmas 13 and 8 we know

$$\frac{z_0 + d_1 y + i}{4} \in \mathbf{L}_{n+2}.$$

Another two applications of Lemma 4 to this yields

$$\frac{z_0 + d_1 y + i}{4} = \frac{z_1 + e_1 + 2e_0}{4},$$

where $z_1 \in \mathbf{L}_n$ and $e, e_0 \in \mathbf{Sd}$. Finally we compute

$$\frac{xy + z + i}{4} = \frac{x_1 y + (z_0 + d_1 y + i) + d_0 + i}{8} = \frac{x_1 y + (z_1 + e_1 + 2e_0) + d_0 + i}{8}.$$

With $m := e_1 + 2e_0 + d_0 + i$ and Lemma 5 we get

$$= \frac{\frac{x_1 y + z_1 + J(m)}{4} + K(m)}{2}.$$

□

Extracted Term (`cLMultcSatLCl`)**.** The extracted term `cLMultcSatLCl`: $\mathbf{Sd}_2 \to \mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbb{L} \to \mathbf{Sd} \times \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L}$ is given by

$$\texttt{cLMultcSatLCl}(i, n, d_1 :: u, v, d_0 :: w) := \langle K(m), J(m), u, w_0 \rangle,$$

where $m := e_1 + 2e_0 + d_0 + i$ and

$$e_0 :: e_1 :: w_0 := \texttt{cLAvcToL}(n+2, w, \texttt{cLSdTimes}(n+2, d_1, v)).$$

Lemma 16 (`LMultcToL`)**.** *For all $n \in \mathbb{N}, i \in \mathbb{D}_2$ and $x, z \in \mathbf{L}_{3n}, y \in \mathbf{L}_{3n \dot{-} 1}$ we have*

$$\frac{xy + z + i}{4} \in \mathbf{L}_n$$

Proof. By induction on $n \in \mathbb{N}$. In the step case assume $i \in \mathbb{D}_2$, $x, z \in \mathbf{L}_{3n+3}$ and $y \in \mathbf{L}_{3n+2}$. We need to prove

$$\frac{xy + z + i}{4} \in \mathbf{L}_{n+1}.$$

To that end we show there exist $d \in \mathbb{D}$ and $x' \in \mathbf{L}_n$ with

$$\frac{xy + z + i}{4} = \frac{x' + d}{2}.$$

Then the claim follows from the second introduction axiom. From Lemma 15 we obtain $d \in \mathbf{Sd}$, $j \in \mathbf{Sd}_2$, $x_1 \in \mathbf{L}_{3n+2}$ and $z_1 \in \mathbf{L}_{3n}$ with

$$\frac{xy + z + i}{4} = \frac{\frac{x_1 y + z_1 + j}{4} + d}{2}$$

Hence it suffices to show that $\frac{x_1 y + z_1 + j}{4} \in \mathbf{L}_n$ for which we use the induction hypothesis. This requires that $x_1, z_1 \in \mathbf{L}_{3n}$ and $y \in \mathbf{L}_{3n \dot{-} 1}$ which we either have directly or by applications of Lemmas 1 and 2. □

Extracted Term (`cLMultToL`)**.** The extracted term `cLMultToL`: $\mathbb{N} \to \mathbb{L} \times \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L} \to \mathbb{L}$ is given by recursion, namely

$$\begin{aligned}
&\texttt{cLMultToL}(0, i, u, w, v) := \texttt{U} \\
&\texttt{cLMultToL}(n+1, i, u, w, v) := d :: \texttt{cLMultToL}(n, \langle i_1, \texttt{hd}^{(2)}\ u_1, w_1, \texttt{hd}^{(3)}\ v \rangle),
\end{aligned}$$

where

$$\langle d, i_1, u_1, w_1 \rangle = \texttt{cLMultcSatLCl}(i, 3n, u, v, w).$$

Now we have all the parts to finalize the proof, that $\mathbf{L}$ is closed under multiplication.

Theorem 5 (`LMult`)**.**

$$n \in \mathbb{N} \to x, y \in \mathbf{L}_{3n+3} \to xy \in \mathbf{L}_n$$

Proof. By Lemma 14 there exist $x_1, y_1 \in \mathbf{L}_{3n+2}, z \in \mathbf{L}_{3n}$ and $i \in \mathbf{Sd}_2$ with

$$xy = \frac{x_1 y_1 + z_1 + i}{4}.$$

Now with two respectively three applications of Lemmas 1 and 2 to x and y we can use the previous Lemma 16 to get $\frac{x_1 y_1 + z_1 + i}{4} \in \mathbf{L}_n$. □

Extracted Term (`cLMult`)**.** The extracted term `cLMult`: $\mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbb{L}$ is defined by

$$\texttt{cLMult}(n, u, v) := \texttt{cLMultcToL}(n, i, \texttt{hd}^{(2)}\ u_1, \texttt{hd}^{(3)}\ v_1, w),$$

where

$$\langle i, u_1, v_1, w \rangle := \texttt{cLMultToMultc}(3n, u, v).$$

We proceed with the proof that $\mathbf{L}$ is closed under division. We first prove

$$\left\{ \frac{x}{y} \,\middle|\, x \in \mathbf{L}_{n+3}, y \in \mathbf{L}_{n+2}, \frac{1}{4} \leq y, |x| \leq y \right\} \subseteq$$
$$\left\{ \frac{\frac{x}{y} + d}{2} \,\middle|\, d \in \mathbf{Sd}, x \in \mathbf{L}_{n}, y \in \mathbf{L}_{n+2}, \frac{1}{4} \leq y, |x| \leq y \right\}.$$

Then in a second step, by induction on $\mathbb{N}$ this directly gives us

$$\left\{ \frac{x}{y} \,\middle|\, x \in \mathbf{L}_{3n}, y \in \mathbf{L}_{3n \dot{-} 1}, \frac{1}{4} \leq y, |x| \leq y \right\} \subseteq \mathbf{L}_n.$$

Lemma 17 (`LDivSatLClAuxL`,`LDivSatLClAuxR`). *Assume $n \in \mathbb{N}$ and $x \in \mathbf{L}_{n+3}, y \in \mathbf{L}_{n+2}$ with*

$$\frac{1}{4} \leq y \wedge |x| \leq y.$$

Then we have

$$\begin{aligned} 0 \leq x &\to (2x - y) \in \mathbf{L}_n \\ x \leq 0 &\to (2x + y) \in \mathbf{L}_n \end{aligned}$$

Proof. First assume $0 \leq x$. Using the assumptions we can estimate

$$\left. \begin{array}{l} 2x - y \leq 2y - y \leq |y| \leq 1 \\ y - 2x \leq y \leq |y| \leq 1 \end{array} \right\} \Rightarrow \left| \frac{x - \frac{y}{2}}{2} \right| \leq \frac{1}{4}.$$

Hence we can apply Lemma 10 for our goal. Now by Theorem 4 it remains to prove $x, -\frac{y}{2} \in \mathbf{L}_{n+3}$. The first we have by assumption and the latter follows from Lemma 7 and an application of the second introduction axiom using $\frac{y}{2} = \frac{y+0}{2}$ and the assumption $y \in \mathbf{L}_{n+2}$. The second formula is proven in a similar fashion but with an application of Lemma 7. □

Extracted Term (`cLDivSatLClAuxL`,`cLDivSatLClAuxR`). The extracted terms, both of type $\mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbb{L}$, are given by

$$\begin{aligned} &\texttt{cLDivSatLClAuxL}(n, u, v) := \texttt{cLToLQuad}(n, \texttt{cLAverage}(n + 2, u, \texttt{cLUMinus}(\texttt{SdM} :: v))), \\ &\texttt{cLDivSatLClAuxR}(n, u, v) := \texttt{cLToLQuad}(n, \texttt{cLAverage}(n + 2, u, \texttt{SdM} :: v)). \end{aligned}$$

Lemma 18 (`TripleCase`).

$$3 \leq n \to x \in \mathbf{L}_n \to 0 \leq x \vee |x| \leq \frac{1}{8} \vee x \leq 0$$

Proof. Three applications of Lemma 4 give $d_0, d_1, d_2 \in \mathbf{Sd}$ and some $y \in \mathbf{L}_n$ with

$$x = \frac{4d_0 + 2d_1 + d_2 + y}{8}.$$

By case-distinctions on $d_i \in \mathbf{Sd}$ we get

$$\left.\begin{array}{r} d_0 = 1 \\ d_0 = 0, d_1 = 1 \\ d_0 = d_1 = 0, d_2 = 1 \end{array}\right\} \Rightarrow 0 \leq x \qquad \left.\begin{array}{r} d_0 = d_1 = 0, d_2 = -1 \\ d_0 = 0, d_1 = -1 \\ d_0 - 1 \end{array}\right\} \Rightarrow x \leq 0$$

and

$$d_0 = d_1 = d_2 = 0 \Rightarrow |x| \leq \frac{1}{8}. \qquad \square$$

Extracted Term (`cTripleCase`)**.** The extracted term `cTripleCase`$: \mathbb{L} \to \mathbb{U} + \mathbb{U} + \mathbb{U}$ is directly defined according to the case-distinction in the proof, e.g.,

$$\texttt{cTripleCase}(0 :: 0 :: 0 :: u) = \texttt{InL}(\texttt{InR}).$$

In the following we will omit it and just write out the case-distinction.

Lemma 19 (`LDivSatLCl`)**.** *Assume $n \in \mathbb{N}$, $x \in \mathbf{L}_{n+3}$, $y \in \mathbf{L}_{n+2}$ with*

$$\frac{1}{4} \leq y \wedge |x| \leq y.$$

Then

$$\exists_{d_0 \in \mathbb{D}} \exists_{x_0 \in \mathbf{L}_n} |x_0| \leq y \wedge \frac{x}{y} = \frac{\frac{x_0}{y} + d_0}{2}.$$

Proof. We use Lemma 18 and distinguish cases. In the first case $0 \leq x$ and we define $d_0 = 1$ and

$$x_0 = 2x - y = 4\frac{x - \frac{y}{2}}{2}.$$

An application of Lemma 17 directly yields $x_0 \in \mathbf{L}_n$ and we compute

$$\frac{x}{y} = \frac{\frac{x_0 + y}{2}}{y} = \frac{\frac{x_0}{2} + 1}{2}.$$

In case $x \leq 0$ we proceed in a similar fashion with $d_0 = -1$ and $x_0 = 2x + y$. The remaining case is $|x| \leq \frac{1}{8}$. Here we define $d_0 = 0$ and $x_0 = 2x$. Since $x \in \mathbf{L}_{n+3}$ and $|x| \leq \frac{1}{8} \leq \frac{1}{2}$ by Lemma 10 we get $2x \in \mathbf{L}_{n+2}$. Two applications of 1 yield $x_0 \in \mathbf{L}_n$. Futhermore

$$\frac{x}{y} = \frac{\frac{2x}{y}}{2} = \frac{\frac{x_0}{y} + 0}{2}. \qquad \square$$

Extracted Term (`cLDivSatLCl`)**.** By unfolding the case-distinction the extracted term of type $\mathbb{N} \to \mathbb{L} \to \mathbb{L} \to \mathbf{Sd} \times \mathbb{L}$ can be represented in the following way.

$$\texttt{cLDivSatLCl}(n, u, v) := \begin{cases} \langle 1, \texttt{cLDivSatLClAuxL}(u, v) \rangle, & 0 \leq x \\ \langle 0, \texttt{hd}^{(2)}(\texttt{cLToDouble}(n + 2, u))) \rangle, & |x| \leq \frac{1}{8} \\ \langle -1, \texttt{cLDivSatLClAuxR}(u, v) \rangle, & x \leq 0 \end{cases}$$

Now we can finally prove that $\mathbf{L}$ is closed under division:

Theorem 6 (`LDiv`)**.**

$$\forall_{n\in\mathbb{N}}\forall_{x\in\mathbf{L}_{3n}}\forall_{y\in\mathbf{L}_{3n\dot{-}1}}\left(\frac{1}{4}\leq y\to|x|\leq y\to\frac{x}{y}\in\mathbf{L}_n\right)$$

Proof. By induction on $n\in\mathbb{N}$. In the base case we have $\left|\frac{x}{y}\right|\leq 1$ and are done. So assume $x\in\mathbf{L}_{3n+3}$, $y\in\mathbf{L}_{3n+2}$. By the previous Lemma 19 we get $d_0\in\mathbf{Sd}, x_0\in\mathbf{L}_{3n}$ with $|x_0|\leq y$ and

$$\frac{x}{y}=\frac{\frac{x_0}{y}+d_0}{2}.$$

By the second introduction axiom we need to prove $\frac{x_0}{y}\in\mathbf{L}_n$. With the induction hypothesis it remains to prove $x_0\in\mathbf{L}_{3n}$ and $y\in\mathbf{L}_{3n\dot{-}1}$. The former we have and the latter follows with applications of Lemmas 1 and 2. □

Extracted Term (`cLDiv`)**.** The extracted term `cLDiv`: $\mathbb{N}\to\mathbb{L}\to\mathbb{L}\to\mathbb{L}$ is given by recursion on $n\in\mathbb{N}$.

$$\begin{aligned}&\texttt{cLDiv}(0,u,v)=\texttt{U},\\&\texttt{cLDiv}(n+1,u,v)=d::\texttt{cLDiv}(n,u_1,\texttt{hd}^{(3)}\ v),\end{aligned}$$

where

$$\langle d,u_1\rangle:=\texttt{cLDivSatLCl}(3n,u,v).$$

Next we turn our attention the computation of the square. We could just use Theorem 5 and obtain

$$x\in\mathbf{L}_{3n+3}\to x^2\in G_n$$

but there is a way to obtain an algorithm with a lower bound for the lookahead. The computation is based on the following decomposition.

Lemma 20 (`LSquareAux`)**.** *If* $x\in\mathbf{L}_{n+2}$ *then there exist* $d_0,d_1\in\mathbf{Sd}$ *and* $y\in\mathbf{L}_n$ *with*

$$x^2=\frac{\frac{y^2+d_1^2+2d_0^2+4d_0d_1}{8}+\frac{d_1y+d_0^2+2d_0y}{4}}{2}.$$

Proof. By two applications of Lemma 4 we get $y\in\mathbf{L}_n$ and $d_0,d_1\in\mathbf{Sd}$ with

$$x=\frac{y+d_1+2d_0}{4}.$$

The rest is obtained by elementary arithmetic. □

Extracted Term (`cLSquareAux`)**.** The extracted term is given by

$$\texttt{cLSquareAux}(d_0::d_1::u)=\langle d_0,d_1,u\rangle.$$

Remark 4. *In the proof of the following Theorem we avoid an unnecessary use of course-of-values-induction by using induction over $\mathbb{N}$ in the following way.*

$$A0 \to A1 \to \forall_{n\in\mathbb{N}}(An \to A(n+2)) \to \mathbb{N} \subseteq A$$

If τ is the type associated to A, the realizer $f\colon \tau \to \tau \to (\mathbb{N} \to \tau \to \tau) \to \mathbb{N} \to \tau$ is given by

$$\begin{aligned}
&f(t_0, t_1, H, 0) = t_0,\\
&f(t_0, t_1, H, 1) = t_1,\\
&f(t_0, t_1, H, n+2) = H(n, f(t_0, t_1, n)).
\end{aligned}$$

Theorem 7 (`LSquare`)**.**

$$\forall_{n\in\mathbb{N}}(x \in \mathbf{L}_{n+4} \to x^2 \in \mathbf{L}_n)$$

Proof. By the remark above we prove the three following statements which then imply the claim.

$$\forall_x(x \in \mathbf{L}_4 \to x^2 \in \mathbf{L}_0), \qquad \forall_x(x \in \mathbf{L}_5 \to x^2 \in \mathbf{L}_1),$$
$$\forall_{n\in\mathbb{N}}\forall_x(x \in \mathbf{L}_{n+4} \to x^2 \in \mathbf{L}_n) \to \forall_{n\in\mathbb{N}}\forall_x(x \in \mathbf{L}_{n+6} \to x^2 \in \mathbf{L}_{n+2})$$

The first one is immediate since $|x| \leq 1$ implies $|x^2| \leq 1$.
The second one holds, since $|x^2| \geq 0$, namely we decompose $x = \frac{y+d}{2}$, compute

$$x^2 = \frac{\frac{(y+d)^2-2}{2} + 1}{2}$$

and $\left|(y+d)^2 - 2\right| \leq 2$. Hence the second introduction axiom of $\mathbf{L}$ can be applied with $d = 1$.
For the third assume $\forall_{n\in\mathbb{N}}\forall_x(x \in \mathbf{L}_{n+4} \to x^2 \in \mathbf{L}_n)$ and $x \in \mathbf{L}_{n+6}$. We use Lemma 20 and get $d_0, d_1 \in \mathbf{Sd}$ and $y \in \mathbf{L}_{n+4}$ with

$$x^2 = \frac{\frac{y^2+d_1^2+2d_0^2+4d_0d_1}{8} + \frac{d_1y+d_0^2+2d_0y}{4}}{2}.$$

By an application of Theorem 4 we need to show that the two summands of the numerator are in $\mathbf{L}_{n+3}$. The left one can be rewritten as

$$\frac{\frac{\frac{y^2+d_1^2}{2}+d_0^2}{2} + d_0d_1}{2},$$

so we can apply the introduction-rule three times and it remains to prove $y^2 \in \mathbf{L}_n$, which follows from the assumption since we have $y \in \mathbf{L}_{n+4}$.
The right side can be written as

$$\frac{\frac{d_1y+d_0^2}{2} + d_0y}{2}.$$

By an application of Theorem 4 we need to provide $\frac{d_1y+d_0^2}{2} \in \mathbf{L}_{n+4}$ and $d_0y \in \mathbf{L}_{n+4}$. The second follows directly from Lemma 8. For the first one we use the introduction axiom and Lemmas 1 and 8. □

Extracted Term (`cLSquare`)**.** The extracted term `cLSquare`: $\mathbb{N} \to \mathbb{L} \to \mathbb{L}$ is given by

$$\begin{aligned}
&\mathtt{cLSquare}(0, u) = \mathtt{U}, \\
&\mathtt{cLSquare}(1, u) = 1 :: \mathtt{U}, \\
&\mathtt{cLSquare}(n+2, d_0 :: d_1 :: u) = \mathtt{cLAverage}(n+2, v_0, v_1),
\end{aligned}$$

where

$$\begin{aligned}
&v_0 = d_0 d_1 :: d_0^2 :: d_1^2 :: \mathtt{cLSquare(n,u)}, \\
&v_1 = \mathtt{cLAverage}(n+3, d_0^2 :: (\mathtt{tl}(\mathtt{cLSdTimes}(n+4, d_1, u))), \mathtt{cLSdTimes}(n+4, d_0, u)).
\end{aligned}$$

2.2. Composition

In this section we want to analyze the look-ahead of compound expressions. We can immediately prove the following

Theorem 8 (`LComp`)**.** *Let $f, g \colon \mathbb{R} \to \mathbb{R}$ and $r, s \in \mathbb{N} \to \mathbb{N}$ total functions. Further assume that we have*

$$\begin{aligned}
&\forall_{x,n}(x \in \mathbf{L}_{r(n)} \to (f\,x) \in G_n), \\
&\forall_{x,n}(x \in \mathbf{L}_{s(n)} \to (g\,x) \in G_n),
\end{aligned}$$

then

$$\forall_{x,n}(x \in G_{(s \circ r)(n)} \to (f \circ g)(x) \in G_n)$$

Proof. Immediately from the two assumptions. □

Extracted Term (`cLComp`)**.** Assuming that r, s only occur non-computational the extracted term `cLComp`: $(\mathbb{L} \to \mathbb{L}) \to (\mathbb{L} \to \mathbb{L}) \to \mathbb{L} \to \mathbb{L}$ is simply given by

$$\mathtt{cLComp}(f_0, f_1, u) := f_0(f_1 u).$$

If r, s are computationally relevant, then `cLComp`: $(\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{L} \to \mathbb{L}) \to (\mathbb{N} \to \mathbb{L} \to \mathbb{L}) \to \mathbb{N} \to \mathbb{L} \to \mathbb{L}$ is defined by

$$\mathtt{cLComp}(r_0, r_1, H_0, H_1, n) = H_0(n, H_1(r_0\ n), u).$$

Using this we can e.g., prove the following

Lemma 21. *For all $n \in \mathbb{N}$ we have*

$$\begin{aligned}
&x, y \in \mathbf{L}_{9n+12} \to z \in \mathbf{L}_{3n+3} \to xyz \in \mathbf{L}_n, \\
&x, y, z \in \mathbf{L}_{3n+6} \to \frac{xy + xz}{2} \in \mathbf{L}_n.
\end{aligned}$$

3. Conclusion and further work

We presented a formal method for extracting verified algorithms for exact real number arithmetic based on stream representations. The verification is based on an explicit representation of real numbers by Cauchy-sequences of rational numbers, which ultimately do not appear in the extracted terms. Hence the only axioms needed are the introduction and elimination axioms for the predicates. The novelty of this approach is that the lookahead of these extracted programs is directly part of the specification. All the proofs have been carried out in the proof assistant Minlog and correctness proofs were automatically generated.

The same methodology can be applied to analyze the lookahead of algorithms that are based on *gray-code* as in [10]. Furthermore it would be interesting to formalize the connection between the two predicates ${}^{co}\mathbf{I}$ and $\mathbf{L}$. Namely for certain programs extracted from a proof involving $\mathbf{L}$ there is a corresponding program for ${}^{co}\mathbf{I}$. The latter will then exactly have the lookahead specified in the former.

[1] H. Schwichtenberg, F. Wiesnet, Logic for exact real arithmetic, Logical Methods in Computer Science 17:2, 2021.

[2] H. Schwichtenberg, Logic for exact real arithmetic: Multiplication, in: D. Bridges, H. Ishihara, H. Schwichtenberg, M. Rathjen (Eds.), Handbook of constructive mathematics, Cambridge University Press, 2022, to appear.

[3] K. Miyamoto, H. Schwichtenberg, Program extraction in exact real arithmetic, Mathematical Structures in Computer Science 25 (Special issue 8) (2015) 1692–1704.

[4] U. Berger, From coinductive proofs to exact real arithmetic: theory and applications, Log. Methods Comput. Sci. 7 (1). `doi:10.2168/LMCS-7(1:8)2011`.
URL `https://doi.org/10.2168/LMCS-7(1:8)2011`

[5] H. Schwichtenberg, S. S. Wainer, Proofs and Computations, Perspectives in Logic, Association for Symbolic Logic and Cambridge University Press, 2012.

[6] K. Miyamoto, The Minlog System, `http://www.mathematik.uni-muenchen.de/~logik/minlog/index.php`, [Online; accessed 29-January-2017] (2017).

[7] F. Wiesnet, Introduction to Minlog, in: K. Mainzer, P. Schuster, H. Schwichtenberg (Eds.), Proof and Computation, World Scientific, 2018, pp. 233–288.

[8] E. Bishop, D. Bridges, Constructive analysis, Vol. 279, Springer Science & Business Media, 2012.

[9] H. Schwichtenberg, Constructive analysis with witnesses, in: Schwichtenberg and Spies [11], pp. 323–353.

[10] U. Berger, K. Miyamoto, H. Tsuiki, H. Schwichtenberg, Logic for Gray-code Computation, In D. Probst and P. Schuster: Concepts of Proof in Mathematics, Philosophy, and Computer Science, 2016, pp. 69–110.

[11] H. Schwichtenberg, K. Spies (Eds.), Proof Technology and Computation. Proc. NATO Advanced Study Institute, Marktoberdorf, 2003, Vol. 200 of Series III: Computer and Systems Sciences, IOS Press, 2006.