Lookahead analysis in exact real arithmetic with logical methods

Nils Köpp

Ludwig-Maximilians Universität, Theresienstr. 39, 80333 München

Helmut Schwichtenberg

Ludwig-Maximilians Universität, Theresienstr. 39, 80333 München

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

Lookahead analysis in exact real arithmetic with logical methods

Nils Köpp

Ludwig-Maximilians Universität, Theresienstr. 39, 80333 München

Helmut Schwichtenberg

Ludwig-Maximilians Universität, Theresienstr. 39, 80333 München

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 explicitly 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. This predicate $\mathbf{L}$ is introduced in Section 2. In Section 2.1 we give a quick overview of the definitions and main theorems from [1] and [2], and in Section 2.2 we present our new definition and prove some basic properties. Then in Section 3 we use this method to get explicit bounds for the lookahead in exact real arithmetic. Section 4 concludes.

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 a 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

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

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

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

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

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

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

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

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.

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

Here ι_I is the algebra of realizers of the inductive predicate I, with constructors of ι_I corresponding to the clauses of I. The n.c. predicate $I^{\mathbf{r}}$ has one more argument (of type ι_I) than I, with $I^{\mathbf{r}}\vec{t}x$ expressing that x realizes $I\vec{t}$.

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 $f \;\mathbf{r}\; \forall_{n \in \mathbb{N}} A := \forall_{n,m}(m \;\mathbf{r}\; (n \in \mathbb{N}) \to (f\, m) \;\mathbf{r}\; A)$. 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.

Implementation in Minlog

For computing extracted terms and verifying the correctness of proofs the proof assistant Minlog [6] is used. Minlog is designed as an implementation of *TCF*. In particular this means that Minlog implements a (simply) typed constructive arithmetic with the following additional features.

- Inductively and coinductively defined predicates are added.
- There is a distinction between computationally relevant (c.r.) and non-computational (n.c.) predicates.

- Realizability predicates are added.
- Partial functionals are allowed. They are defined by equations, which are possibly non-terminating (like corecursion).
- Minimal logic is used, with $\to$, $\forall$ the only primitive logical connectives. Existence, disjunction and conjunction are inductively defined.

In Minlog a soundness proof can be automatically generated for every proof. All the proofs in this paper have been formalized[1] in Minlog. After each proof in the following we state its computational content.

More specifically, in *TCF* all variables are typed. The following table shows which variables have which type.

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

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 the type of real numbers $\mathbb{R}$ is explicitly defined as the type $(\mathbb{N} \to \mathbb{Q}) \times (\mathbb{Z}^+ \to \mathbb{N})$.

An introduction to Minlog can be found in [7] or `doc/tutor.pdf` in the Minlog directory.

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 1.1 (Cauchy representation)**.** We define the algebra $\mathbb{R}$ 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 verification.

[1] In the file `sdind.scm` in the directory `examples/analysis`

2. The inductive predicate L and its properties

2.1. 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 2.1 (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 2.2. Henceforth we will use $\mathbf{Sd}$ for the predicate as well as the algebra given by three nullary constructors. A realiser of ${}^{co}\mathbf{I}x$ is exactly a stream of signed digits. The algebra of streams of signed digits $\mathbb{S}$ is given by the single constructor

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

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

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

Theorem 2.3 (`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 2.4 (`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

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

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

Theorem 2.5 (`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

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

2.2. 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

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

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

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

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 lose anything compared to Theorem 2.3, we just added some control regarding the precision of the input.

Definition 2.6. We define $\mathbf{L}$ as the least predicate satisfying the clauses

$$\begin{aligned}
&\forall_x (|x| \leq 1 \to \mathbf{L}(x, 0)),\\
&\forall_{d \in \mathbf{Sd}, x, y, n} \Big(\mathbf{L}(x, n) \to y = \frac{x+d}{2} \to \mathbf{L}(y, n+1) \Big).
\end{aligned}$$

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

$$\begin{aligned}
&\mathtt{U} : \mathbb{L}\\
&\mathtt{C} : \mathbf{Sd} \to \mathbb{L} \to \mathbb{L},
\end{aligned}$$

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 $\mathbf{L}$ it is 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 2.7. The elimination axiom of $\mathbf{L}$ has the following form. Assume $P \subseteq \mathbb{R} \times \mathbb{N}$. Then if

$$\begin{aligned} &\forall_x (|x| \leq 1 \to Px0), \\ &\forall_{d \in \mathbf{Sd}, x, y, n} \Big(\mathbf{L}xn \to Pxn \to y = \frac{x+d}{2} \to Py(n+1) \Big) \end{aligned}$$

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 $\mathcal{R}_{\mathbb{L}}^{\tau} : \mathbb{L} \to \tau \to (\mathbf{Sd} \to \mathbb{L} \to \tau \to \tau) \to \tau$ with the computation rules

$$\begin{aligned} &\mathcal{R}_{\mathbb{L}}^{\tau}(\mathtt{U}, t_0, f) = t_0, \\ &\mathcal{R}_{\mathbb{L}}^{\tau}(\mathtt{C}dv, t_0, f) = f(d, v, \mathcal{R}_{\mathbb{L}}^{\tau}(v, t_0, f)), \end{aligned}$$

i.e., recursion for lists.

2.3. Basic properties

Lemma 2.8 (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 elimination axiom of $\mathbf{L}$. □

Extracted term (cLSuccToL).

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

Lemma 2.9 (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).

$$\begin{aligned} &\mathtt{cLToLPred}(\mathtt{U}) := \mathtt{U}, \\ &\mathtt{cLToLPred}(d :: u) := u. \end{aligned}$$

Remark 2.10. In the following we will write the extracted term of the previous two lemmas as $\texttt{tl}\colon \mathbb{L} \to \mathbb{L}$, i.e., the usual tail-function. Furthermore, for better readability, we will sometimes make use of the function $\texttt{hd}\colon \mathbb{L} \to \mathbf{Sd}$ defined by

$$\begin{aligned} \texttt{hd}(\texttt{U}) &:= 0, \\ \texttt{hd}(d :: u) &:= d. \end{aligned}$$

Lemma 2.11 (`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 property.

Lemma 2.12 (`LClosure`)**.**

$$x \in \mathbf{L}_{n+1} \to \exists_{d\in\mathbf{Sd},x_0}\Big(x = \frac{x_0 + d}{2} \wedge x_0 \in \mathbf{L}_n\Big).$$

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 will be suppressed 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 2.13. *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 2.14 (`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)**.** $\texttt{cLSd}\colon \mathbb{N} \to \mathbf{Sd} \to \mathbb{L}$

$$\begin{aligned}&\texttt{cLSd}(0,d) := \texttt{U},\\&\texttt{cLSd}(n+1,d) := d :: (\texttt{cLSd}(n,d)).\end{aligned}$$

Lemma 2.15 (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 $d \in \mathbf{Sd}, x, (-x) \in \mathbf{L}_n$ 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)**.** $\texttt{cLUMinus}\colon \mathbb{L} \to \mathbb{L}$ *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 2.16 (LSdTimes)**.**

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

Proof. By induction on $d \in \mathbf{Sd}$. If $d = -1$ we use Lemma 2.15 and if $d = 0$ then Lemma 2.14. Otherwise we are done by compatibility. □

Extracted term (cLSdTimes)**.** $\texttt{cLSdTimes}\colon \mathbb{N} \to \mathbf{Sd} \to \mathbb{L} \to \mathbb{L}$ *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 2.17 (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 2.18 (`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 2.12 and 2.17 with a case distinction on $\mathbf{Sd}$. 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*

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

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

3. Exact real number arithmetic with lookahead

3.1. 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 lookahead 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

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

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

Lemma 3.1 (`LAvToAvc`)**.**

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

Proof. By applying Lemma 2.12 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). cLAvToAvc: $\mathbb{L} \to \mathbb{L} \to \mathbf{Sd}_2 \times \mathbb{L} \times \mathbb{L}$

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

Lemma 3.2 (LAvcSatLCl).

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

Proof. We use Lemma 2.12 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 2.13 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 3.3 (LAvcToL).

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

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

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

and $x, y \in \mathbf{L}_{n+1}$. By Lemma 3.2 there exists $j \in \mathbf{Sd}_2, d \in \mathbf{Sd}$ 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* `cLAvcToL`: $\mathbb{N} \to \mathbf{Sd}_2 \to \mathbb{L} \to \mathbb{L} \to \mathbb{L}$ *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 3.4 (`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 3.1 and 3.3. □

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

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

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

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

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

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

Lemma 3.5 (`LMultToMultc`). *For all* $n \in \mathbb{N}$ *and* $x, y \in \mathbf{L}_{n+3}$ *we have*

$$\exists_{i \in \mathbf{Sd}_2} \exists_{x_1, y_1 \in \mathbf{L}_{n+2}} \exists_{z_1 \in \mathbf{L}_n} \Big(xy = \frac{x_1 y_1 + z_1 + i}{4} \Big).$$

Proof. By Lemma 2.12 there are $d_1, e_1 \in \mathbf{Sd}$ 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 2.16 and Theorem 3.4 we also know

$$\frac{e_1 x_1 + d_1 y_1}{2} \in \mathbf{L}_{n+1}.$$

So by another application of Lemma 2.12 there exist $z_1 \in \mathbf{L}_n$ and $d_2 \in \mathbf{Sd}$ with $\frac{e_1 x_1 + d_1 y_1}{2} = \frac{z_1 + d_2}{2}$. We compute

$$xy = \frac{x_1 y_1 + d_1 e_1 + e_1 x_1 + d_1 y_1}{4} = \frac{x_1 y_1 + z_1 + (d_2 + d_1 e_1)}{4}. \qquad \square$$

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

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

Proof. We decompose x, z via Lemma 2.12 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 3.3 and 2.16 we know

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

Another two applications of Lemma 2.12 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 2.13 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 3.7 (`LMultcToL`). *For all* $n \in \mathbb{N}, i \in \mathbf{Sd}_2$ *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 $i \in \mathbf{Sd}_2,\ x, z \in \mathbf{L}_{3n+3}$ 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 that there exist $d \in \mathbf{Sd}$ 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 3.6 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 2.8 and 2.9. □

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*

$$\mathtt{cLMultToL}(0, i, u, w, v) := \mathtt{U}$$

$$\mathtt{cLMultToL}(n+1, i, u, w, v) := d :: \mathtt{cLMultToL}(n, \langle i_1, \mathtt{hd}^{(2)}\ u_1, w_1, \mathtt{hd}^{(3)}\ v\rangle),$$

where

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

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

Theorem 3.8 (`LMult`).

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

Proof. By Lemma 3.5 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 2.8 and 2.9 to x and y we can use the previous Lemma 3.7 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*

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

where

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

We proceed with the proof that $\mathbf{L}$ is closed under division. First we 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 3.9 (`LDivSatLClAuxL`,`LDivSatLClAuxR`). *Assume that $n \in \mathbb{N}$ 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

$$0 \leq x \to (2x - y) \in \mathbf{L}_n,$$
$$x \leq 0 \to (2x + y) \in \mathbf{L}_n.$$

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 2.18 for our goal. Now by Theorem 3.4 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 2.15 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 2.15. □

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

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

Lemma 3.10 (`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 2.12 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 ($\mathtt{cTripleCase}$)**.** *The extracted term* $\mathtt{cTripleCase} \colon \mathbb{L} \to \mathbb{U} + \mathbb{U} + \mathbb{U}$ *is directly defined according to the case distinction in the proof, e.g.,*

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

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

Lemma 3.11 ($\mathtt{LDivSatLCl}$)**.** *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

$$\exists_{d_0 \in \mathbf{Sd}} \exists_{x_0 \in \mathbf{L}_n} \Big(|x_0| \leq y \wedge \frac{x}{y} = \frac{\frac{x_0}{y} + d_0}{2} \Big).$$

Proof. We use Lemma 3.10 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 3.9 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 2.18 we get $2x \in \mathbf{L}_{n+2}$. Two applications of 2.8 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 ($\mathtt{cLDivSatLCl}$)**.** *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.*

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

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

Theorem 3.12 (`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 3.11 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 2.8 and 2.9. □

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

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

Remark 3.13. If we compare the algorithms obtained for average, multiplication and division in [2, 1] on stream representations to the algorithm obtained here for list representations, then we can see that they essentially operate in exactly the same way. So although not formally verified we successfully have analysed the lookahead of the former algorithms. In fact this can be made more precise e.g., if we take the extracted term `cCoIAv` $:\mathbb{S}(\mathbf{Sd})\to\mathbb{S}(\mathbf{Sd})\to\mathbb{S}(\mathbf{Sd})$ of theorem 2.3 then its lookahead is bound by $n+1$ exactly if for all streams u_i, v_i and $n\in\mathbb{N}$ we have

$$\texttt{cCoIAv}((u_0|_{n+1})*v_0,(u_1|_{n+1})*v_1)|_n=\texttt{cCoIAv}(u_0,u_1)|_n,$$

where $\cdot|_n:\mathbb{S}(\mathbf{Sd})\to\mathbb{L}(\mathbf{Sd})$ is the initial segment of length n defined by

$$u|_0=[\,],\qquad(d::u)|_{n+1}=d::(u|_n),$$

and $*$ is the concatenation of a list and a stream. Now we also have the extracted term `cLAv` $:\mathbb{L}(\mathbf{Sd})\to\mathbb{L}(\mathbf{Sd})\to\mathbb{L}(\mathbf{Sd})$ of Theorem 3.4 and we consider the following diagram:

$$\begin{tikzcd}\mathbb{L}(\mathbf{Sd})\times\mathbb{L}(\mathbf{Sd}) \arrow[r, "\texttt{cLAv}"] & \mathbb{L}(\mathbf{Sd}) \\ \mathbb{S}(\mathbf{Sd})\times\mathbb{S}(\mathbf{Sd}) \arrow[u, "\cdot|_{n+1}\times\cdot|_{n+1}"] \arrow[r, "\texttt{cCoIAv}"] & \mathbb{S}(\mathbf{Sd}) \arrow[u, "\cdot|_n"']\end{tikzcd}$$

Then we can formally prove that this diagram commutes for all $n \in \mathbb{N}$ which entails the property above regarding the lookahead of `cCoIAv`. Although these kinds of proofs are possible for all the corresponding pairs of extracted terms from Section 3, these proofs involve unfolding all the extracted terms which quickly becomes unmanageable.

3.2. Squaring

Instead of only tracking the lookahead of essentially known algorithms we are also interested in finding new algorithms with (possibly) optimal lookahead. Now in Theorem 3.8 we have proven that our algorithm for multiplication has lookahead $3n+3$. For the optimal lookahead we consider x, y for which we know n digits of a signed-digit representation respectively, i.e., they are contained in intervals of length $\frac{1}{2^{n-1}}$. Multiplying these intervals we get a new interval I_n with length $|I_n| \leq \frac{2^n-1}{4^{n-1}}$. Then we can estimate

$$|I_{n+3}| \leq \frac{1}{2^{n-1}},$$

which is exactly what we need to determine the first n digits of $x \cdot y$. Hence theoretically the optimal bound for the lookahead of multiplication should be $n+3$.

We will now give a proof that $\mathbf{L}$ is closed under the square. The obtained bound for the lookahead will be $n+4$. We could just use Theorem 3.8 and obtain

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

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 3.14 (`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 2.12 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 3.15. 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

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

Theorem 3.16 (`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 3.14 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 3.4 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 3.4 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 2.16. For the first one we use the introduction axiom and Lemmas 2.8 and 2.16. □

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

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

where

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

Remark 3.17. A very similar decomposition as in Lemma 3.14 can be used for the multiplication. Namely

$$
\frac{\frac{x+d_0}{2} + d_1}{2} \cdot \frac{\frac{y+e_0}{2} + e_1}{2} = \frac{\frac{\frac{\frac{xy+d_1e_1}{2} + d_0e_1}{2} + d_0e_0}{2} + \frac{\frac{\frac{e_1x+d_1y}{2} + d_1e_0}{2} + \frac{e_0x+d_0y}{2}}{2}}{2},
$$

from which we can prove

$$
x, y \in \mathbf{L}_{n+5} \to x \cdot y \in \mathbf{L}_n.
$$

Remark 3.18. Now that we have the squaring algorithm for the list representation we are also interested if the same algorithm is also possible for the stream representation. So following our methodology, the algorithm should be obtained from a proof of

$$
x \in {}^{co}\mathbf{I} \to x^2 \in {}^{co}\mathbf{I}.
$$

While possible, technically such a proof is much harder.

3.3. Haskell translation

The terms extracted from the proofs can be translated to Haskell or Scheme programs. We compare the run-time of the algorithms obtained for multiplication for the stream and list-representations respectively. For this purpose we translate the terms to Haskell and measure the time (with `:set +s`) it takes to compute the first n digits of the square of some stream or list u. This u is a pseudo-random sequence of signed digits generated with the Haskell `System.Random` package. The time (in seconds) in the table below is the average over 10 runs. Furthermore we also compare with the multiplication algorithm from Remark 3.15.

As expected the runtime of `cCoIMult` and `cLMult` is about the same. The small difference is due to the testing setup, since the algorithm for lists spends more time printing to the screen.

digits	`cCoIMult`	`cLMult`	`cLMultNew`
10	.03	.03	.02
50	.12	.10	.05
250	2.34	2.44	.79
1000	22.27	24.67	12.65

Figure 1: runtime test for squaring

4. 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 should be applicable to analyze the lookahead of algorithms that are based on *Gray-code* as in [10]. We leave this for future work.

References

[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, to appear in Mathematics for Computation (M4C) (ed. M. Benini, O. Beyersdorff, M. Rathjen, P. Schuster), World Scientific, Singapore (2022).

[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`.

[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 of Grundlehren der mathematischen Wissenschaften, Springer Verlag, Berlin, Heidelberg, New York, 1985.

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

[10] U. Berger, K. Miyamoto, H. Schwichtenberg, H. Tsuiki, Logic for Gray-code computation, in: D. Probst, P. Schuster (Eds.), Concepts of Proof in Mathematics, Philosophy, and Computer Science, De Gruyter, 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.