Introduction to Real PCF
(Notes)

Martín Hötzel Escardó

Department of Computer Science
Laboratory for the Foundations of Computer Science
University of Edinburgh

e-mail: `M.Escardo@ed.ac.uk`

Abstract

Real PCF is a programming language for exact real number computation, introduced in order to investigate theoretical issues such as semantics, computational adequacy, program correctness and derivation, universality, algorithms for higher-order functions.

1 Introduction

Real PCF is a programming language for higher-order exact real number computation [9]. It was introduced in order to investigate theoretical issues such as semantics and computational adequacy [7], universality [8], program correctness and derivation [12, 13], algorithms for higher-order functions such as definite integration and global maxima determination [5, 4]. Related work by Pietro Di Gianantonio, Abbas Edalat, Peter Potts and the author can be found in [14, 29, 6].

Real PCF has ground types for truth-values, natural numbers and real numbers. The derived types are defined by iterating a function-type construction. Sequences are functions on natural numbers, and predicates are truth-valued functions. Thus, in Real PCF one can define real numbers, functions between real numbers, predicates on real numbers, sequences of real numbers, sequences of sequences of real numbers, sequences of functions, functionals mapping sequences to numbers (such as limiting operators), functionals mapping functions to numbers (such as integration and supremum operators), functionals mapping predicates to truth-values (such as existential and universal quantification operators), and so on. An extension of Real PCF with recursively defined types is considered in [12, 13].

These notes are a complement to the lecture, consisting of background, basic concepts, and references to the literature.

2 The programming language PCF

In order to properly introduce Real PCF we must go back to PCF.

In 1969, Dana Scott introduced a logic of computable functions, later known as LCF [16, 17, 25]. Although Scott's paper was published only in 1993 as [33], the original manuscript was widely circulated and inspired a whole generation of researchers. Later, in 1977, Gordon Plotkin published a seminal paper in which he considered the terms of Scott's logic as a programming language [27], called PCF (programming language for computable functions).

PCF can be seen as a simplified version of modern typed functional programming languages [2] such as Haskell, Miranda, ML [20, 19, 26]. One has ground types for booleans and naturals numbers. More complex types are formed by iterating a function-type construction: if one has types σ and τ, then one can form a new type $(\sigma \to \tau)$. A version of PCF which is much closer to such programming languages was developed in [28]. In addition to ground types and function types, one has product types, sum types, and recursively defined types. This language is referred to as FPC in [18].

Briefly, PCF has the following primitives: constants for truth-values and natural numbers, a conditional construction, a test for zero, successor and predecessor maps, and a recursion construction.

3 Domain theory

In his original paper, Scott introduced a model of LCF which validates the rules of the logic. A subset of the rules defines program evaluation. In this model, types are interpreted as mathematical structures that are nowadays known, rather uninspiredly, as *domains*.

A domain can be seen either as a partially ordered set or, equivalently, as topological space [31]. Good references to domain theory include [1, 28] (many connections of domain theory with general mathematics can be found in [15]). The reason why topology shows up in the mathematical theory of computation is, of course, that computations are approximation processes, and topology is a general theory of approximation [35, 34].

3.1 The domain of total and partial functions

A prototypical domain is the collection of total and partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ ordered by graph inclusion. In this case the Scott topology coincides with the finite-information topology used in recursion theory [30]. A set $U \subseteq [\mathbb{N} \rightharpoonup \mathbb{N}]$ is open iff the following two conditions are met:

1. If $f \in U$ and $f \subseteq g$ then $g \in U$.
2. If $f \in U$ then there is a finite $f' \in U$ with $f' \subseteq f$.

Here a function is identified with its graph and hence is finite iff its graph is finite.

This topology has the property that a functional $F : [\mathbb{N} \rightharpoonup \mathbb{N}] \to [\mathbb{N} \rightharpoonup \mathbb{N}]$ is continuous iff it is monotone and finite amounts of its output depend only of finite amounts of its input. More precisely,

1. If $f \subseteq g$ then $F(f) \subseteq F(g)$.
2. For every finite $h \subseteq F(f)$ there is a finite $f' \subseteq f$ such that already $h \subseteq F(f')$.

The Myhill-Shepherdson Theorem says that the computable functionals are continuous (and, conversely, that every "effectively" continuous functional is computable) [30].

3.2 Order from topology

In general, the order and topology of a domain are related as follows. Given the topology, one recovers the order by

$x \sqsubseteq y$ iff every neighbourhood of x is a neighbourhood of y.

That is, y is better than x if it has a richer supply of neighbourhoods. This order can be defined in any topological space and is known as the *specialization order* (see e.g. [21]). In the context of domain theory it is known as the *information order.*

The Scott topology is highly non-Hausdorff, as in a Hausdorff space the specialization order is the identity. In fact, it is immediate that a space is T_1 iff its specialization order is the identity. However, many domains which occur in practice have classical Hausdorff spaces as their subspaces of maximal points. For example, the subspace of maximal points of the domain of total and partial functions consists of the total functions, and the relative Scott topology on the total functions makes them into a space homeomorphic to Baire space. Our main example will be the Euclidean real line, which appears as the subspace of maximal points of the interval domain discussed below.

3.3 Topology from order

Conversely, given the order of a domain one recovers the topology by[1]:

U is Scott open iff the following two conditions are met:

1. If $x \in U$ and $x \sqsubseteq y$ then $y \in U$.
2. If the least upper bound of an ascending sequence

$$x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$$

belongs to U, then some term x_i already belongs to U.

[1]This definition works for countably based domains (which are the ones one uses in practice). In general, ascending sequences have to be weakened to directed sets.

For many purposes, a domain can be regarded as a poset with a least element, called *bottom* and denoted by $\perp$, and least upper bounds of ascending sequences—but often one needs more structure not discussed in this note—see [1]. A function $f : D \to E$ is Scott continuous iff it is monotone and preserves least upper bounds of ascending sequences.

3.4 Flat domains

The simplest but not completely trivial domain is the so-called flat domain of natural numbers $\mathbb{N}_\perp = \mathbb{N} \cup \{\perp\}$. The idea is that a partial function $f : \mathbb{N} \to \mathbb{N}$ gets represented by a total function $f_\perp : \mathbb{N}_\perp \to \mathbb{N}_\perp$:

$$f_\perp(x) = \begin{cases} f(x) & \text{if } f(x) \text{ is defined,} \\ \perp & \text{otherwise.} \end{cases}$$

The order on $\mathbb{N}_\perp$ is given by $x \sqsubseteq y$ iff $x = \perp$ or $x = y$. A set U is Scott open iff $\perp \in U$ implies $U = \mathbb{N}_\perp$. A function is continuous iff it is monotone. Similarly, one has a flat domain of truth values $\mathbb{B}_\perp = \{\text{true}, \text{false}, \perp\}$ (cf. Kleene's three-valued logic [22]).

3.5 Fixed points, function spaces and recursive definitions

A *fixed point* of an endomap $f : X \to X$ is a point $x \in X$ such that $f(x) = x$. Every continuous endomap $f : D \to D$ on a domain D has a least fixed-point, given by the least upper bound of the sequence $f^n(\perp)$. Least fixed points are interesting mainly in connection with function spaces and recursive definitions.

Given domains D and E, one constructs a new domain $[D \to E]$ as follows. The elements of $[D \to E]$ are the Scott continuous functions. The order is defined pointwise:

$$f \sqsubseteq g \text{ iff } f(d) \sqsubseteq f(d) \text{ in } E \text{ for all } d \in D.$$

A useful fact is that the functional $[D \to D] \to D$ which maps a continuous function to its least fixed point is itself continuous.

Suppose one has a recursive definition of a function, say $f : \mathbb{N}_\perp \to \mathbb{N}_\perp$, such as

$$f(n) = \text{if } n = 0 \text{ then } 1 \text{ else } n \cdot f(n-1).$$

In this simple case one has just an inductive definition of the factorial function. In general, such a recursive definition can be seen as a functional equation in f. But how does one know that there is a solution? And if there is more than one solution, which one does one select?

The above definition is equivalent to

$$f = F(f),$$

where $F : [\mathbb{N}_\perp \to \mathbb{N}_\perp] \to [\mathbb{N}_\perp \to \mathbb{N}_\perp]$ is defined by:

$$F(f)(n) = \text{if } n = 0 \text{ then } 1 \text{ else } n \cdot f(n-1)$$

In general, there is no reason why there should be a function f satisfying $f = F(f)$. However, if the functional F is continuous, one knows that there is in fact a least solution to the recursive definition, namely the least fixed point of F. In the case of PCF, computational adequacy will tell us that this is the appropriate solution of the recursive definition.

4 Scott's model of PCF

The ground types of booleans and naturals of PCF are interpreted as the flat domains defined above. Function types are interpreted as continuous function spaces. Recursive definitions are interpreted via function spaces and least fixed-points as discussed above. Since the primitive operations, including the least fixed-point operator, are continuous, and since continuous functions are closed under composition, all PCF-definable functions are continuous.

4.1 Computational adequacy of Scott's model

Computationally, $\perp$ corresponds to non-termination. This is made precise as follows:

> A PCF program M of ground type evaluates to a non-bottom value in finitely many computation steps iff it denotes that value in Scott's model [27].

This is known as *computational adequacy* of Scott's model.

4.2 Universality of Scott's model

It is easy to see that one can define all partial recursive functions in PCF. Perhaps surprisingly, some computable functions in the model fail to be definable. Two of them are a "parallel" conditional such that

$$\text{pif } \perp \text{ then } x \text{ else } x = x$$

and an existential quantifier $\exists : [\mathbb{N}_\perp \to \mathbb{B}_\perp] \to \mathbb{B}_\perp$ defined by

$$\exists(p) = \begin{cases} \text{true} & \text{if } p(n) = \text{true for some } n, \\ \text{false} & \text{if } p(\perp) = \text{false}, \\ \perp & \text{otherwise.} \end{cases}$$

(Notice that, by monotonicity of $p \in [\mathbb{N}_\perp \to \mathbb{B}_\perp]$, the condition $p(\perp) = \text{false}$ is equivalent to $p(x) = \text{false}$ for all $x \in \mathbb{N}_\perp$.) However,

> If one adds these functions to PCF as primitive, then PCF becomes *universal* for Scott's model, in the sense that all computable functions in the model become definable [27].

5 Real PCF

Real PCF is PCF extended with a ground type for real numbers, interpreted as an interval domain that we refer to as the *partial real line*.

5.1 Partial real numbers and universality

The partial real line is the domain of *non-empty closed and bounded* real intervals ordered by reverse inclusion, together with an artificial least element $\perp$. The idea is that singleton intervals represent *total real numbers* and that non-singletons represent *partial real numbers.* This is justified by the fact that the relative Scott topology on the set of total numbers yields a homeomorphic copy of the Euclidean real line. Although the author is not aware of any explicit mention of partial numbers in the literature, the idea is certainly a natural one.

Perhaps the most compelling reason to make partial numbers official is that the set of computable real numbers is *not* recursively enumerable[2], but the larger set of computable partial real numbers *is.* This is analogous to what happens with total and partial computable functions on the natural numbers. For instance, if only total numbers were expressible in Real PCF, a universality result would not be possible (unless the syntax of Real PCF was undecidable).

It has to be stressed that it is not enough to add just a bottom element denoting non-termination, for non-termination of a computation of a real number can take place after some meaningful approximations are produced. In other words, the set computable total numbers enlarged by just a bottom element is still *not* recursively enumerable. Thus, it is not enough to talk about partial *functions* on the reals. One does have to admit a rich supply of partial *reals.* This is implicit in the work of Boehm and Cartwright [3], where the impossibility of universality of any abstract data type for (total) real numbers is proved by diagonalization, even in the presence of partial functions.

5.2 Connections with interval analysis

The interval domain was introduced by Dana Scott as an example in [32]. It is related to the interval space used in interval analysis [24] as follows [11]. The topology of the interval space is induced by the Hausdorff metric: the distance between two intervals is the maximum of the distances of the end-points. A set U of intervals is Scott open iff it is Hausdorff open *and* the conditions $x \in U$ and $x \supseteq y$ together imply $y \in U$ for all intervals x and y. The second condition is the one which forces Scott continuous functions to be monotone. In other words, a Hausdorff continuous *monotone* function is Scott continuous. But in interval analysis one *does* work with monotone functions! Thus, implicitly, interval analysts are actually working with the Scott topology.

As pointed out by Mike Smyth (personal communication), this indicates that the Hausdorff topology is actually the Lawson topology in the case of the interval domain. In general, the Lawson topology is a refinement of the Scott topology which, for many classes of well-behaved domains, is compact Hausdorff [15, 1]. In our case we have an artificial least element in the interval domain, so the above remarks are slightly inaccurate. Strictly speaking, the interval domain endowed with the Lawson topology is the one-point compactification of the interval space endowed with the Hausdorff metric.

[2]See e.g. [23], where computability on the reals is expressed in terms of the interval domain.

5.3 Primitive operations and universality

For simplicity and without essential loss of generality, here we consider Real PCF extended with a type for the unit interval $[0,1] \subseteq \mathbb{R}$ instead of the whole real line. Notice that the interval domain over $[0,1]$ has a natural bottom element, namely $\bot = [0,1]$.

It is interesting that, from the point of view of universality, the following five numerical operations are enough, provided the parallel conditional and the existential quantifier discussed above are also available:

$$x \mapsto x/2, \qquad x \mapsto (x+1)/2, \qquad x \mapsto \min(2x,1), \qquad x \mapsto \max(0,2x-1)$$

and

$$(x <_\bot y) = \begin{cases} \text{true} & \text{if } x < y, \\ \text{false} & \text{if } x > y, \\ \bot & \text{otherwise.} \end{cases}$$

Strictly speaking, one considers Scott continuous extensions of these functions on the reals to the interval domain (the general extension problem is investigated in detail in [10]). See [8] for the proof of universality of Real PCF and [12, 13] for universality of Real PCF extended with recursive types.

5.4 Primitive operations and evaluation

From the point of view of evaluation, one needs more. Instead of considering only the two linear maps

$$x \mapsto x/2, \qquad x \mapsto (x+1)/2,$$

which have image $[0,1/2]$ and $[1/2,1]$, one considers more generally for every interval $a = [\underline{a}, \overline{a}] \subseteq [0,1]$ the unique increasing affine map $x \mapsto px+q : [0,1] \to [0,1]$ with image a, namely

$$\mathrm{cons}_a(x) = (\overline{a} - \underline{a})x + \underline{a}.$$

In practice a is assumed to have *rational* end-points so that one has countably many primitive operations. The idea is that

> If M is a Real PCF program of real number type, then one knows that the value of $\mathrm{cons}_a(M)$ is contained in the interval a, even if one doesn't know anything about the value of M.

Thus, the expression $\mathrm{cons}_a(M)$ can be regarded as a *partially evaluated program* with *partial result* a. Then the following evaluation scheme is applied:

1. If one starts with $M = M_0$, then the Real PCF machinery tries to reduce M_0 to a partially evaluated program $\mathrm{cons}_{a_1}(M_1)$.

2. This attempt may engage into an infinite loop.

3. If it doesn't and the interval a_1 is small enough, evaluation is successful and stops.

4. Otherwise, the Real PCF machinery applies the same scheme to M_1.
5. The partial result produced by M_1, if any, is combined with that of M_0, giving rise to a better result.
6. And so on.

Partial results are combined as follows. First, one defines a composition operations on the unit interval domain by

$$a \circ b = \mathrm{cons}_a(b).$$

This makes the interval domain over $[0, 1]$ into a monoid (=semigroup) with neutral element $\perp = [0, 1]$. Associativity can be expressed as $\mathrm{cons}_a \circ \mathrm{cons}_b(x) = \mathrm{cons}_{a \circ b}(x)$. Moreover, the information order of the interval domain is recovered from composition by the following refinement property:

> $a \supseteq c$ iff there is some b with $a \circ b = c$.

(If a is not a singleton then such an interval b is necessarily unique.) Hence, if one starts with a Real PCF program $M = M_0$ of real number type and gets partial results

$$M_0 \to \mathrm{cons}_{a_1}(M_1) \qquad M_1 \to \mathrm{cons}_{a_2}(M_2) \quad \ldots \quad M_{n-1} \to \mathrm{cons}_{a_n}(M_n) \ldots$$

then one has the refined partial evaluation

$$M \to \mathrm{cons}_{a_1}(\mathrm{cons}_{a_2}(\ldots(\mathrm{cons}_{a_n}(M_n)\ldots))) \to \mathrm{cons}_{a_1 \circ a_2 \circ \cdots \circ a_n}(M_n).$$

Therefore one gets a nested sequence of better and better partial results

$$a_1 \supseteq a_1 \circ a_2 \supseteq \cdots \supseteq a_1 \circ a_2 \circ \cdots \circ a_n \supseteq \cdots.$$

5.5 Computational adequacy

The Real PCF machinery is defined in such a way that computational adequacy holds in the following form: For any Real PCF program of real number type, one can get as close to the value of M as one wishes, in finitely many computation steps. More precisely,

> If a Real PCF program M denotes a (total or partial) real number $x \neq \perp$, then for every *open* interval $a \supseteq x$, as small as one pleases, $M \to \mathrm{cons}_b(N)$ for some b and N with $a \supseteq b \supseteq x$.

The proof uses, among other things, the fact that the primitive operations are Scott continuous and the fact that a set U of intervals is Scott open iff the following two conditions are met:

1. If $x \in U$ and $x \supseteq y$ then $y \in U$.
2. If $x \in U$ then there is an interval $x' \in U$ whose interior contains x.

Here the interior is, of course, taken in the Euclidean topology of the real line. It follows continuity of endomaps on the partial real line is characterized by

> A monotone map f is continuous iff for every partial number y whose interior contains $f(x)$, there is a partial number x' whose interior contains x and one already has that the interior of y contains $f(x')$.

5.6 Primitive operations and efficiency

If one generalizes from linear maps to Möbius transformations, then nothing is gained in terms of expressibility, but a lot is gained in terms of efficiency [29]. For instance, many elementary functions can be defined via Möbius transformations without the use of the parallel conditional or the existential quantifier. This is based on the work of Vuillemin [36]. The Möbius transformations also form a monoid, and a similar refinement property is satisfied, but here one uses the base interval $[0, \infty]$ instead of the unit interval—see [6]. This makes Möbius transformations suitable for program evaluation as above, as it is shown in [29].

References

[1] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D.M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, Oxford, 1994.

[2] R. Bird and P. Wadler. *Introduction to Functional Programming.* Prentice-Hall, New York, 1988.

[3] H.J. Boehm and R. Cartwright. Exact real arithmetic: Formulating real numbers as functions. In Turner. D., editor, *Research Topics in Functional Programming*, pages 43–64. Addison-Wesley, 1990.

[4] A. Edalat and M.H. Escardó. Integration in Real PCF. *Information and Computation.* To appear (available at http://www.dcs.ed.ac.uk/home/mhe/papers.html).

[5] A. Edalat and M.H. Escardó. Integration in Real PCF (extended abstract). In *Proceedings of the Eleventh Annual IEEE Symposium on Logic In Computer Science*, pages 382–393, New Brunswick, New Jersey, USA, July 1996.

[6] A. Edalat and P.J. Potts. A new representation for exact real numbers. *Electronic Notes in Theoretical Computer Science*, 6, 1997.

[7] M.H. Escardó. PCF extended with real numbers. *Theoretical Computer Science*, 162(1):79–115, August 1996.

[8] M.H. Escardó. Real PCF extended with $\exists$ is universal. In A. Edalat, S. Jourdan, and G. McCusker, editors, *Advances in Theory and Formal Methods of Computing: Proceedings of the Third Imperial College Workshop, April 1996*, pages 13–24, Christ Church, Oxford, 1996. IC Press.

[9] M.H. Escardó. PCF extended with real numbers: A domain-theoretic approach to higher-order exact real number computation. Technical Report ECS-LFCS-97-374 (http://www.dcs.ed.ac.uk/lfcsreps/EXPORT/97/ECS-LFCS-97-374/index.html), Department of Computer Science, University of Edinburgh, December 1997. PhD Thesis at Imperial College of the University of London, 1996.

[10] M.H. Escardó. Properly injective spaces and function spaces. *Topology and Its Applications*, 1997. To appear (available at http://www.dcs.ed.ac.uk/home/mhe/papers.html).

[11] M.H. Escardó and D.M. Claudio. Scott domain theory as a foundation for interval analysis. Technical Report 218, UFRGS/II, Porto Alegre, Brazil, 1993.

[12] M.H. Escardó and T. Streicher. Induction and recursion on the partial real line via biquotients of bifree algebras. In *Proceedings of the Twelveth Annual IEEE Symposium on Logic In Computer Science*, Warsaw, Polland, Jun 1997.

[13] M.H. Escardó and T. Streicher. Induction and recursion on the partial real line with applications to Real PCF. *Theoretical Computer Science*, January 1999. To appear (available at http://www.dcs.ed.ac.uk/home/mhe/papers.html).

[14] P. Di Gianantonio. *A functional approach to computability on real numbers.* PhD thesis, University of Pisa, 1993. Technical Report TD 6/93.

[15] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, 1980.

[16] M. J. C. Gordon, R. Milner, and C. Wadsworth. Edinburgh LCF, a mechanical logic of computation. Report csr-11-77 (in 2 parts), Department of Computer Science, University of Edinburgh, 1977.

[17] Michael J. Gordon, Arthur J. Milner, and Christopher P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *Lecture Notes in Computer Science.* Springer-Verlag, 1979.

[18] C. A. Gunter. *Semantics of Programming Languages – Structures and Techniques.* The MIT Press, London, 1992.

[19] I. Holyer. *Functional Programming with Miranda.* Pitman, 1991.

[20] Paul Hudak and Joseph H. Fasel. A gentle introduction to Haskell. *ACM SIGPLAN Notices*, 27(5):T1–T53, may 1992.

[21] P.T. Johnstone. *Stone Spaces.* Cambridge University Press, Cambridge, 1982.

[22] S.C. Kleene. *Introduction to Metamathematics.* North-Holland, Amsterdam, 1952.

[23] P. Martin-Löf. *Notes on Constructive Mathematics.* Almqvist & Wiksell, Stockholm, 1970.

[24] R.E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, 1966.

[25] L.C. Paulson. *Logic and Computation: Interactive Proof with LCF.* Cambridge University Press, Cambridge, 1987.

[26] L.C. Paulson. *ML for the working programmer.* Cambridge University Press, Cambridge, 1991.

[27] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(1):223–255, 1977.

[28] G. Plotkin. Domains. Post-graduate Lectures in advanced domain theory, University of Edinburgh, Department of Computer Science. Available at http://hypatia.dcs.qmw.ac.uk/sites/other/domain.notes.other, 1983.

[29] P.J. Potts, A. Edalat, and M.H. Escardó. Semantics of exact real arithmetic. In *Proceedings of the Twelveth Annual IEEE Symposium on Logic In Computer Science*, Warsaw, Polland, Jun 1997.

[30] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[31] D. S. Scott. Continuous lattices. In F.W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lectures Notes in Mathematics*, pages 97–136. Springer-Verlag, 1972.

[32] D. S. Scott. Lattice theory, data types and semantics. In *Formal semantics of programming languages*, pages 66–106, Englewood Cliffs, 1972. Prentice-Hall.

[33] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM and OWHY. *Theoretical Computer Science*, 121:411–440, 1993. Reprint of a manuscript produced in 1969.

[34] M.B. Smyth. Power domains and predicate transformers: a topological view. In J. Diaz, editor, *Automata, Languages and Programming*, pages 662–675. Springer-Verlag, 1983. LNCS 154.

[35] M.B. Smyth. Topology. In S. Abramsky, D. M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 641–761. Clarendon Press, Oxford, 1992.

[36] J. Vuillemin. Exact real computer arithmetic with continued fractions. *IEEE Transactions on Computers*, 39(8):1087–1105, 1990.