Dividing by Zero – How Bad Is It, Really?[*]

Takayuki Kihara[1] and Arno Pauly[2]

1   Department of Mathematics, University of California, Berkeley, USA
    kihara@math.berkeley.edu
2   Département d'Informatique, Université Libre de Bruxelles, Belgium
    Arno.M.Pauly@gmail.com

Abstract

In computable analysis testing a real number for being zero is a fundamental example of a non-computable task. This causes problems for division: We cannot ensure that the number we want to divide by is not zero. In many cases, any real number would be an acceptable outcome if the divisor is zero - but even this cannot be done in a computable way.

In this note we investigate the strength of the computational problem *Robust division*: Given a pair of real numbers, the first not greater than the other, output their quotient if well-defined and any real number else. The formal framework is provided by Weihrauch reducibility. One particular result is that having later calls to the problem depending on the outcomes of earlier ones is strictly more powerful than performing all calls concurrently. However, having a nesting depths of two already provides the full power. This solves an open problem raised at a recent Dagstuhl meeting on Weihrauch reducibility.

As application for *Robust division*, we show that it suffices to execute Gaussian elimination.

1   Introduction

We cannot divide by zero! is probably the first mathematical impossibility statement everyone encounters. In the setting we see it first, arithmetic of concrete integers, this does not cause any problems: Since it is obvious whether some number is zero or not, we simply refrain from attempting it – and the multiplicative absorption of 0 ensures that we have no reason for an attempt anyway. As our mathematical world expands to include more kinds of numbers and variables, we may have to introduce case distinctions at times in order to avoid this problem[1].

In most practical situations, this may seem unproblematic. However, a fundamental observation by BROUWER in the early development of constructive mathematics was that *we cannot in general decide whether a real number is zero or not*. Thus, a case distinction based on whether our intended denominator is zero or not is not constructive. In a constructive setting, we can only divide by a number we know to be different from zero.

---

[1] Forgetting about these cases has probably caused a lot of anguish to pupils learning the outcome of their exams.

To consider a concrete example where we might want to divide by a number that could be zero, consider $a, b \in \mathbb{R}$ with $0 \le a \le b$, and the linear equation $a = bx$. We know that there is a solution $x_0 \in [0, 1]$: If $b \ne 0$, then $x_0 := \frac{a}{b}$, otherwise $b = a = 0$, and any x works. We see that we do not actually care about whether $b = 0$ or not, and we do not even need any particular outcome of a misguided attempt to calculate $\frac{0}{0}$ – any number would do.

Unfortunately, the algorithm to divide a real number a by a real number b starts with searching for a rational number bounding b away from 0. If no such number exists, there will be no output at all, rather than some arbitrary number. The *robust division* we would like to employ to solve linear equations as above is not actually computable.

In this note, we study the extent of non-computability of robust division in the formal setting of Weihrauch reducibility. Some results had already been obtained in [17]. We will recall that robust division lies strictly in between the traditional non-constructive principles LLPO and LPO and some other basic properties. Our concern then is with the question how multiple uses of robust division interact. We show that sequential uses of robust division cannot be reduced to parallel uses – however, it suffices to have a nesting depths of 2.

In [17], finding the solution to systems of linear inequalities via a modified Fourier-Motzkin elimination, and finding Nash equilibria in bimatrix games were explored as applications of robust division. Here, we shall consider Gaussian elimination as additional example.

An extended version of this article is available as [15].

2   Background

Computability on the reals and other represented spaces

The long history of studying computability on the real numbers presumably goes back to BOREL [2] (see [1] for a detailed historical picture). Here, we follow the school of WEIHRAUCH [28]. Computability is initially introduced over $\{0,1\}^{\mathbb{N}}$ by means of Type-2 machines. These are obtained from the usual Turing machine model via a simple modification: The head on output tape can move to the right only (and in particular does so whenever a symbol is written), and the machines never halt. The restriction on the output tape ensures that as the computation proceeds, longer and longer finite prefixes of the ultimate infinite output are available.

The transfer of computability from $\{0,1\}^{\mathbb{N}}$ to the spaces of actual interest is achieved via the notion of a represented space. For a more detailed introduction to the theory of represented spaces, we refer to [21]. A represented space is a pair $\mathbf{X} = (X, \delta_X)$ of a set X and a partial surjection $\delta_X :\subseteq \{0,1\}^{\mathbb{N}} \to X$ (the representation).

A multi-valued function[2] between represented spaces is a multi-valued function between the underlying sets. For $f :\subseteq \mathbf{X} \rightrightarrows \mathbf{Y}$ and $F :\subseteq \{0,1\}^{\mathbb{N}} \to \{0,1\}^{\mathbb{N}}$, we call F a realizer of f (notation $F \vdash f$), iff $\delta_Y(F(p)) \in f(\delta_X(p))$ for all $p \in \text{dom}(f\delta_X)$.

$$\begin{array}{ccc} \{0,1\}^{\mathbb{N}} & \xrightarrow{\;F\;} & \{0,1\}^{\mathbb{N}} \\ \Big\downarrow{\scriptstyle \delta_{\mathbf{X}}} & & \Big\downarrow{\scriptstyle \delta_{\mathbf{Y}}} \\ \mathbf{X} & \xrightarrow{\;f\;} & \mathbf{Y} \end{array}$$

A map between represented spaces is called computable (continuous), iff it has a computable (continuous) realizer. Note that a priori, the notion of continuity for maps between represented

---

[2] For a discussion of the notion of a multi-valued function, and in particular the difference to the notion of a relation, we refer to [23], [20].

spaces differs from topological continuity. For the admissible represented spaces (in the sense of [24]), the two notions do coincide, if a represented space is equipped with the final topology inherited from Cantor space along the representation. All representations we are concerned with in this note are admissible.

Before we introduce the standard representation of the real numbers, we fix some standard enumeration $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ of the rationals. Now we define $\rho :\subseteq \{0,1\}^{\mathbb{N}} \to \mathbb{R}$ via $\rho(0^{n_0}10^{n_1}1\ldots) = x$ iff $\forall i \in \mathbb{N}\ |\nu_{\mathbb{Q}}(n_i) - x| < 2^{-n}$. Note that using e.g. the binary or decimal expansion would not have worked satisfactorily[3]. The choice of ρ ensures that, informally spoken, every naturally encountered continuous function on the reals will be computable.

The naturals are represented in the obvious way by $\delta_{\mathbb{N}}(0^n 1^{\mathbb{N}}) = n$. The finite spaces $\{0, \ldots, n\}$ are just the corresponding subspaces of $\mathbb{N}$. Likewise, we introduce the represented space $[0, 1]$ as a subspace of $\mathbb{R}$. The space $\mathbb{S}$ has the elements $\{\bot, \top\}$ represented via $\delta_{\mathbb{S}}(0^{\mathbb{N}}) = \bot$ and $\delta_{\mathbb{S}}(p) = \top$ for $p \neq 0^{\mathbb{N}}$.

For any represented space $\mathbf{X}$, there is a canonical definition of the represented space $\mathcal{A}(\mathbf{X})$ of closed subsets of $\mathbf{X}$. We only require this for the specific choices of $\mathbf{X} = [0, 1], \{0, \ldots, n\}$: In the former case, a closed subset is a closed subset in the usual sense, and it is represented by a list of rational open balls exhausting its complement in $[0, 1]$. In the latter, any subset of $\{0, \ldots, n\}$ is an element of $\mathcal{A}(\{0, \ldots, n\})$, and a set A is represented by $p \in \{0, 1\}^{\mathbb{N}}$ iff $01^k 0$ occurs somewhere in p iff $k \notin A$ for any $k \in \{0, \ldots, n\}$.

As there are canonical tupling functions $\langle \ldots \rangle : (\{0,1\}^{\mathbb{N}})^n \to \{0,1\}^{\mathbb{N}}$ available, we can define products of represented spaces in a straight-forward way. We obtain binary and countable disjoint unions by $(\delta_0 + \delta_1)(0p) = \delta_0(p)$ and $(\delta_0 + \delta_1)(1p) = \delta_1(p)$, and $(\coprod_{i \in \mathbb{N}} \delta_i)(0^n 1 p) = \delta_n(p)$. We will iterate the binary product, starting with the convention $\mathbf{X}^0 = \{0\}$ and setting $\mathbf{X}^{n+1} = \mathbf{X}^n \times \mathbf{X}$. Finally, $\mathbf{X}^*$ is shorthand for $\coprod_{i \in \mathbb{N}} \mathbf{X}^i$.

Weihrauch reducibility

Weihrauch reducibility is a computable many-one reduction comparing multi-valued functions between represented spaces. So $f \leq_{\mathrm{W}} g$ informally means that f could be computed with the help of a single oracle-call to g.
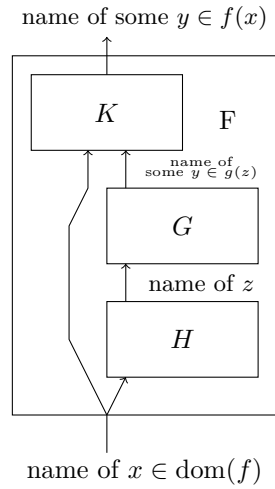
▶ **Definition 1.** Let $f :\subseteq \mathbf{X} \rightrightarrows \mathbf{Y}$ and $g :\subseteq \mathbf{U} \rightrightarrows \mathbf{V}$ be partial multivalued functions between represented spaces. Say that f is Weihrauch reducible to g, in symbols $f \leq_{\mathrm{W}} g$, if there are computable functions $K :\subseteq \{0,1\}^{\mathbb{N}} \times \{0,1\}^{\mathbb{N}} \to \{0,1\}^{\mathbb{N}}$ and $H :\subseteq \{0,1\}^{\mathbb{N}} \to \{0,1\}^{\mathbb{N}}$ such that whenever G is a realizer of g, the function $F := (p \mapsto K(p, G(H(p))))$ is a realizer for f.

Based on earlier work by WEIHRAUCH [26, 27], Weihrauch reducibility was suggested as a framework for computable metamathematics in [5, 4] (see also [12, 17]). We point to the introduction of [6] for a recent overview on the development of the field so far.

We shall denote the set of Weihrauch degrees by $\mathfrak{W}$, and point out some operations on them. As shown in [18], the binary product $\times$, the binary disjoint union $\sqcup$, the countable disjoint union $\coprod$ and the operation * all can be lifted from represented spaces via multivalued functions between represented spaces to Weihrauch degrees. $\mathfrak{W}$ is a distributive lattice, and $\sqcup$ is the join. However, no non-trivial countable suprema exist in $\mathfrak{W}$ as shown in [13]. In particular, $\coprod$ is not the countable join.

Informally, $f \sqcup g$ means that both f and g are available for use, but the user has to decide for each instance on one of the two to call. A call to $f \times g$ means making two independent

---

[3] As already noted by TURING [25].

name of some $y \in f(x)$

K F

name of
some $y \in g(z)$

G

name of z

H

name of $x \in \mathrm{dom}(f)$

Figure 1 Illustrating Definition 1.[4]

calls, one to f and one to g. Using f^* means that we first decide on some number $n \in \mathbb{N}$, and then make n independent calls to f.

We want to use a further operation; corresponding to first making a call to some g and then a call to f depending on the outcome of the call to g. In [9, 7] the operation $\star$ was defined as $f \star g := \max_{\leq_{\mathrm{W}}} \{f' \circ g' \mid f' \leq_{\mathrm{W}} f \wedge g' \leq_{\mathrm{W}} g\}$. Here the maximum is understood to range over all f', g' with types such that $f' \circ g'$ is well-defined. While it is not obvious that this maximum exists, an explicit construction is provided in [11]. Informally, an input to $f \star g$ consists of an input to g together with a multivalued function computing some input to f from an output to g. The output is the output of g together with the output of f.

We iterate both $\times$ and $\star$: $f^0 = f^{(0)} = \mathrm{id}_{\{0,1\}^{\mathbb{N}}}$, and $f^{n+1} = f^n \times f$ and $f^{(n+1)} = f^{(n)} \star f$.

Special Weihrauch degrees

We will refer to a number of well-studied specific Weihrauch degrees in this paper. We shall first recall the degrees LPO and LLPO from [27], the Weihrauch degree counterparts to the Brouwerian counterexamples in intuitionistic mathematics. LPO : $\{0,1\}^{\mathbb{N}} \to \{0,1\}$ maps $0^{\mathbb{N}}$ to 1 and each $p \neq 0^{\mathbb{N}}$ to 0. This map is equivalent to the characteristic function of 0 in $\mathbb{R}$ or $[0,1]$, and we will not distinguish these maps. The map LLPO : $\{0,1\}^{\mathbb{N}} \rightrightarrows \{0,1\}$ outputs 0 if the first 1 in the input occurs at an even position, and 1 if it occurs at an odd position. If the input was $0^{\mathbb{N}}$, both 0 and 1 are valid outputs. An alternative characterization would be LLPO : $\mathbb{R} \times \mathbb{R} \to \{0,1\}$ with $0 \in \mathrm{LLPO}(0,y)$, $1 \in \mathrm{LLPO}(x,0)$ and $0,1 \in \mathrm{LLPO}(x,y)$ if $x \neq 0 \neq y$. It holds that LLPO $<_{\mathrm{W}}$ LPO.

A bountiful source of calibrating principles is found in the closed choice principles and their restrictions:

▶ **Definition 2.** $\mathrm{C_X} :\subseteq \mathcal{A}(\mathbf{X}) \rightrightarrows \mathbf{X}$, $\mathrm{dom}(\mathrm{C_X}) = \{A \in \mathcal{A}(\mathbf{X}) \mid A \neq \emptyset\}$, $x \in \mathrm{C_X}(A) \Leftrightarrow x \in A$.

Thus, closed choice is the task to find a point in a given non-empty closed set. As being non-empty is merely promised rather than constructively witnessed, this task is generally not computable: As long as there is more than one remaining choice, whenever we start

---

[4] This figure was taken from [22].

outputting some point we might learn next that this point is not in the closed set after all (e.g. by reading some rational ball containing it in the name of the closed set).

These principles have been extensively studied [4, 3, 10, 9, 16, 6]. Depending on the topological properties of the space $\mathbf{X}$ and potentially restrictions to certain subsets, these principles have been revealed to be useful in characterizing many other principles.

Most relevant for us are the principles $C_{\{0,\ldots,n\}}$. It was shown in [17] that $C_{\{0,\ldots,n\}} \leq_W C_{\{0,1\}}^n$, and it follows from the independent choice theorem in [3] that $C_{\{0,1\}}^n \leq_W C_{\{0,1\}}^{(n)} \leq_W C_{\{0,\ldots,2^n-1\}}$. It is quite easy to see that $C_{\{0,1\}} \equiv_W$ LLPO.

We make passing references to $C_{\mathbb{N}}$ (and use that $C_{\mathbb{N}} \equiv_W C_{\mathbb{N}} \star C_{\mathbb{N}}$ by the independent choice theorem from [3]), to $CC_{[0,1]}$, the restriction of $C_{[0,1]}$ to connected subsets and to $PCC_{[0,1]}$, the restriction of $C_{[0,1]}$ to connected sets with positive Lebesgue measure. We also mention $\star$-WWKL from [6], which is $\coprod_{n\in\mathbb{N}}(2^{-n})$-WWKL, where ε-WWKL is the restriction of $C_{\{0,1\}^{\mathbb{N}}}$ to sets with measure at least ε. By $UC_{\mathbf{X}}$ we denote the restriction of $C_{\mathbf{X}}$ to singletons [3]. Finally, $C_{\sharp=2}$ and $C_{\sharp\leq2}$ from [16] are the restrictions of $C_{\{0,1\}^{\mathbb{N}}}$ to sets with cardinality 2 and at most 2 respectively.

3 Robust division

We consider two variants of robust division: In one case, we know an upper bound on the result, in the second, we do not. Modulo the rescaling, the first case corresponds to knowing that the denominator is at least as big as the numerator.

▶ **Definition 3** ([17]). Define rDiv $: \mathbb{R} \times \mathbb{R} \rightrightarrows [0,1]$ via $\frac{\min\{|x|,|y|\}}{|y|} \in \text{rDiv}(x,y)$ iff $y \neq 0$ and $z \in \text{rDiv}(x,0)$ for all $x \in \mathbb{R}, z \in [0,1]$.

To simplify notation, we will usually assume that inputs (x,y) for rDiv already satisfy $0 \leq x \leq y$, so that $\frac{\min\{|x|,|y|\}}{|y|} = \frac{x}{y}$ holds.

▶ **Definition 4.** Define ubrDiv $: \mathbb{R}\times\mathbb{R} \rightrightarrows \mathbb{R}$ by $\frac{x}{y} \in \text{ubrDiv}(x,y)$ iff $y \neq 0$ and $z \in \text{ubrDiv}(x,0)$ for all $x, z \in \mathbb{R}$.

It turns out that the case distinction on $y \neq 0$ or $y = 0$ is equivalent to the unbounded case ubrDiv. Thus, we do not need to investigate ubrDiv as an independent basic operation. Note that the following proof also establishes that it makes no difference for the degree of ubrDiv if the result is presumed to be non-negative.

▶ **Proposition 5.** ubrDiv $\equiv_W$ *LPO*.

Proof. The direction ubrDiv $\leq_W$ LPO follows from computability of division where well-defined and the definition.

For the other direction, note that given some $p \in \{0,1\}^{\mathbb{N}}$ we can compute $x, y \in \mathbb{R}$ such that if $p \neq 0^{\mathbb{N}}$, then $\frac{x}{y} = \min\{n \in \mathbb{N} \mid p(n) = 1\}$ and $x = y = 0$ if $p = 0^{\mathbb{N}}$. Furthermore, there is a computable multivalued retract $\tau : \mathbb{R} \rightrightarrows \mathbb{N}$, so we may pretend that the output of ubrDiv(x,y) is a natural number n indicating the position of the first 1 in p, if it exists. Given this number, we can then check whether $p(n)$ is 1 or not, which in turn determines the answer to LPO(p). ◀

The bounded variant of robust division was already established as a *new* degree in [17]. We recall some results on this degree from the literature before continuing its investigation.

▶ **Proposition 6** ([17]).
1. $C_{\{0,1\}} <_W \mathrm{rDiv} <_W LPO$.
2. $\mathrm{rDiv} <_W CC_{[0,1]}$.
3. $\mathrm{rDiv} \not\leq_W C^*_{\{0,1\}}$.

▶ **Proposition 7** ([6, Theorem 16.3, Corollary 16.5 & Theorem 16.6]).
1. $\mathrm{rDiv} <_W PCC_{[0,1]}$.
2. rDiv *is join-irreducible.*
3. $\mathrm{rDiv} \not\leq_W *\text{-}WWKL$.

The preceding results from [6] intuitively state that there is a mechanism to solve rDiv in a probabilistic way with positive probability and error detection. However, there is no way to obtain a positive lower bound on the probability of solving a given instance correctly.

It is a well-known phenomenon in the study of Weihrauch reducibility that closed choice principles make very convenient representatives of Weihrauch degrees (cf. [4, 3, 10, 9, 16]). The case of robust division is no different: For a represented space $\mathbf{X}$ we denote by $\mathrm{AoUC}_{\mathbf{X}}$ the restriction of $C_{\mathbf{X}}$ to $\{A \in \mathcal{A}(\mathbf{X}) \mid |A| = 1\} \cup \{X\}$ following an idea of Brattka. Just by its definition, it is clear that $\mathrm{UC}_{\mathbf{X}} \leq_W \mathrm{AoUC}_{\mathbf{X}} \leq_W C_{\mathbf{X}}$ holds for any space $\mathbf{X}$. In the following we shall focus on $\mathrm{AoUC}_{[0,1]}$.

▶ **Proposition 8.** [5] $\mathrm{rDiv} \equiv_W AoUC_{[0,1]}$.

Proof. The reduction $\mathrm{rDiv} \leq_W \mathrm{AoUC}_{[0,1]}$ is straight-forward: On input $(x, y) \in \mathbb{R}^2$ for rDiv, while the search for a $k \in \mathbb{N}$ with $y > 2^{-k}$ continues, the input to $\mathrm{AoUC}_{[0,1]}$ is kept at $[0, 1]$. If such a k is ever found, one can compute $\frac{x}{y}$, and hence also $\{\frac{x}{y}\}$ as $[0, 1]$ is computably Hausdorff and collapse the unit interval to it.

For the other direction, as long as the input to $\mathrm{AoUC}_{[0,1]}$ has not collapsed, one starts to input $(0, 0)$ to rDiv. If the input of $\mathrm{AoUC}_{[0,1]}$ ever collapsed to $\{z\}$, one can compute z. The input to rDiv can still be chosen from some interval $[0, 2^{-k}] \times [0, 2^{-k}]$. In particular, $x = 2^{-k}z$ and $y = 2^{-k}$ works and forces the correct output.   ◀

4 Sequential versus concurrent uses of rDiv

If multiple uses of some noncomputable principle are needed to solve a particular task, an important distinction is whether these have to be sequential, or can be applied in a concurrent fashion. In the former case, some instances to the principle may depend on outputs obtained from prior invocations. In the latter, each instantiation is independent of the others. For various principles, however, we find that sequential uses can be reduced to concurrent uses.

▶ **Definition 9.** We call f *finitely concurrent*, iff $f^* \equiv_W \coprod_{n \in \mathbb{N}} f^{(n)}$.

▶ **Proposition 10** ([16]). *The following are finitely concurrent:*
1. *LPO*
2. $LLPO \equiv_W C_{\{0,1\}}$
3. $C_{\sharp=2}$
4. $C_{\sharp\leq2}$

---

[5] This result was suggested to the author by Brattka, and has been shown in [19].

Whether rDiv is finitely concurrent in this sense was posed as an open question during the Dagstuhl workshop *Measuring the Complexity of Computational Content* in September 2015 [8]. We can now provide a negative answer[6]:

▶ **Theorem 11.** *$LLPO \star AoUC_{[0,1]} \not\leq_W AoUC_{[0,1]}^k$ for all $k \in \mathbb{N}$.*

Proof. We say that a binary tree $T \subseteq \{0,1\}^*$ is an *a.o.u. tree* if for any height $n \in \mathbb{N}$ either $|T \cap \{0,1\}^n| = 2^n$ or $|T \cap \{0,1\}^n| = 1$. Clearly, one can identify $AoUC_{[0,1]}$ with the partial multi-valued function sending an a.o.u. tree T to all infinite paths through T. We often identify a set $S \subseteq \{0,1\}^*$ with its characteristic function $\chi_S : \{0,1\}^* \to \{0,1\}$. Under this identification, a partial function $t :\subseteq \{0,1\}^* \to \{0,1\}$ is called a *partial tree* if $\text{Tr}(t) := \{\sigma \in \text{dom}(t) : t(\sigma) = 1\}$ forms a subtree of $\{0,1\}^*$. If such t is computable, we call t a *partial computable tree*. Note that a tree is computable if and only if it is of the form $\text{Tr}(t)$ for a partial computable tree t which is *total*, that is, $\text{dom}(t) = \{0,1\}^*$. We say that a partial computable tree t *looks like an a.o.u. tree at* (l,s) if

1. $t(\sigma)[s]$ converges for any binary string σ of length l,
2. for any $n < l$, the cardinality of $\text{Tr}(t) \cap \{0,1\}^n$ is either 2^n or 1,

where $t(\sigma)[s]$ is the result of the computation of $t(\sigma)$ by stage s. Note that given $(l,s) \in \mathbb{N}^2$, we can effectively decide whether t looks like an a.o.u. tree at (l,s) or not. By $\text{aou}(t,s)$ we denote the greatest $l \leq s$ such that t looks like an a.o.u. tree at (l,s).

Consider two partial multi-valued functions $Z_0 = AoUC_{[0,1]} \times \text{id}_\mathbf{X}$ and $Z_1 = (\text{id}_{\{0,1\}^\mathbb{N}} \circ \pi_0, C_{\{0,1\}} \circ \text{eval})$, where $\mathbf{X}$ is the represented space $C(\{0,1\}^\mathbb{N}, \text{dom}(C_{\{0,1\}}))$ of continuous functions from Cantor space $\{0,1\}^\mathbb{N}$ into the hyperspace $\text{dom}(C_{\{0,1\}}) = \mathcal{A}(\{0,1\}) \setminus \{\emptyset\}$ of nonempty closed subsets of $\{0,1\}$. More explicitly, we consider the following two partial multi-valued functions:

$Z_0 : \text{dom}(AoUC_{[0,1]}) \times \mathbf{X} \rightrightarrows \{0,1\}^\mathbb{N} \times \mathbf{X},$

$Z_0(T,S) = AoUC_{[0,1]}(T) \times \{S\},$

$Z_1 : \{0,1\}^\mathbb{N} \times \mathbf{X} \rightrightarrows \{0,1\}^\mathbb{N} \times 2,$

$Z_1(x,S) = \{x\} \times C_{\{0,1\}}(S(x)).$

Clearly, $Z_0 \leq_W AoUC_{[0,1]}$ and $Z_1 \leq_W C_{\{0,1\}}$. We will show that $Z_1 \circ Z_0 \not\leq_W AoUC_{[0,1]}^k$. Let $\{(\mathbf{t}^e, \varphi_e, \psi_e)\}_{e \in \mathbb{N}}$ be an effective enumeration of all triples of k-tuples $\mathbf{t}^e = (t_i^e)_{i<k}$ of partial computable trees, partial computable functions $\varphi_e :\subseteq (\{0,1\}^\mathbb{N})^k \to \{0,1\}^\mathbb{N}$ and $\psi_e :\subseteq (\{0,1\}^\mathbb{N})^k \to \{0,1\}$. Intuitively, $(\mathbf{t}^e, \varphi_e, \psi_e)$ is a triple constructed by the opponent Opp, who tries to show $Z_1 \circ Z_0 \leq_W AoUC_{[0,1]}^k$ for some k. The game proceeds as follows: The proponent Pro of our claim gives an instance (T_r, S_r) of $Z_1 \circ Z_0$, so that $(T_r, S_r) \in \text{dom}(AoUC_{[0,1]}) \times \mathbf{X}$. In particular, T_r is an a.o.u. tree, and S_r is a continuous function from $[T_r]$ into $\mathcal{A}(\{0,1\})$. Then, Opp reacts with an instance $\mathbf{t}^r$ of $AoUC_{[0,1]}^k$, that is, a k-tuple $\mathbf{t}^r = (t_i^r)_{i<k}$ of *total* a.o.u. trees. If Opp wins, Opp has to ensure that if $(p_i)_{i<k}$ is a k-tuple of infinite paths through Opp's a.o.u. trees, that is, $p_i \in [\text{Tr}(t_i^r)]$, then $\varphi_r((p_i)_{i<k}) = x$ is a path through Pro's a.o.u. tree T_r and $\psi_r((p_i)_{i<k})$ chooses an element of Pro's set $S_r(x)$, where Opp can use information on (names of) T_r and S_r to construct φ_r and ψ_r. Our purpose is to prevent Opp's strategy.

Given e, we will introduce the e-th strategy, which works as a proponent Pro of our claim. The e-th strategy Pro will construct a computable a.o.u. tree $T_e \subseteq \{0,1\}^*$ and a computable

---

[6] Which also disproves a claim made in the PhD thesis of the second author [19, Theorem 5.2.1.6].

function $S_e : \{0,1\}^{\mathbb{N}} \to \mathcal{A}(\{0,1\})$ in a computable way uniformly in e. These will prevent Opp's strategy, that is, there is a k-tuple $(p_i)_{i<k}$ of infinite paths through Opp's a.o.u. trees such that if $\varphi_e((p_i)_{i<k}) = x$ chooses a path through Pro's a.o.u. tree T_e then $\psi_e((p_i)_{i<k})$ cannot be an element of Pro's set $S_e(x)$.

We will also define $\mathtt{state}(e, s) \in \{0, \ldots, k\} \cup \{\mathtt{end}\}$. The value $\mathtt{state}(e, s) = q$ for $q \neq \mathtt{end}$ indicates that the e-th strategy Pro believes that by stage s, *at least q many trees* $(t^e_{u(j)})_{j<q}$ *in* Opp*'s k-tuple* $(t^e_i)_{i<k}$ *have been forced not to have more than one infinite path*, that is, $\mathrm{Tr}(t^e_{u(j)})$ for each $j < q$ has a unique infinite path whenever the opponent Opp has a chance of winning this game with the triple $(\mathbf{t}^e, \varphi_e, \psi_e)$. Under this assumption, if $\mathtt{state}(e, s) = q$, the fact that these q many trees has no more than one infinite path will be witnessed at some stage. The value $\mathtt{state}(e, s) = \mathtt{end}$ indicates that the winning of Pro is already witnessed by Pro's action of shrinking Pro's a.o.u. tree T_e to a tree having a unique path which avoids all φ_e-values made by Opp.

By induction on s, we determine the set $T_e \cap \{0,1\}^s$ of strings in T_e of length s and $\mathtt{state}(e, s)$. In the beginning of our construction, we define $\mathtt{state}(e, 0) = 0$. At stage s, we inductively assume that $T_e \cap \{0,1\}^{s-1}$ and $\mathtt{state}(e, s-1)$ have already been defined, say $\mathtt{state}(e, s-1) = q$, and that if $\mathtt{state}(e, s-1) \neq \mathtt{end}$ then $T_e \cap \{0,1\}^{s-1} = \{0,1\}^{s-1}$.

At stage s, the e-th strategy Pro acts as follows:

1. Ask whether there exists $l \leq s$ such that t^e_i for each $i < k$ looks like an a.o.u. tree at (l, s), and at least q-many trees among Opp's k-tuple $(t^e_i)_{i<k}$ have no more than one node above height l. In other words, for $\mathrm{aou}(e, s) := \min_{i<k} \mathrm{aou}(t^e_i, s)$, ask whether there are at least q many $i < k$ such that $|\mathrm{Tr}(t^e_i) \cap \{0,1\}^{\mathrm{aou}(e,s)}| = 1$.
   a. If no, we go to the next stage $s + 1$ after setting $\mathtt{state}(e, s) = \mathtt{state}(e, s-1)$ and $T_e \cap \{0,1\}^s = \{0,1\}^s$.
   b. If yes, go to item (2).

2. Ask whether $\varphi_e((p_i)_{i<k})$ already computes a node (of Pro's a.o.u. tree T_e) of length at least $q + 1$ for any k-tuple of paths p_i through Opp's a.o.u. trees t^e_i, that is,

$$(\forall i < k)(\forall (\sigma_i)_{i<k}) \left[((\forall i < k) \, \sigma_i \in \mathrm{Tr}(t^e_i) \cap \{0,1\}^{\mathrm{aou}(e,s)}) \right.$$
$$\left. \to \; (\forall m \leq q) \, \varphi_e((\sigma_i)_{i<k})(m)[s] \downarrow \right].$$

Here, by effective continuity, the Type-2 computation $\varphi_e : (\{0,1\}^{\mathbb{N}})^k \to \{0,1\}^{\mathbb{N}}$ is approximated by a Type-1 computation $\tilde{\varphi}_e : (\{0,1\}^*)^k \to \{0,1\}^*$ (e.g., consider the Type-2 Turing machine model). We always identify φ_e with $\tilde{\varphi}_e$, and therefore, the notation $\varphi_e((\sigma_i)_{i<k})(m)$ makes sense, that is, by $\varphi_e((\sigma_i)_{i<k})(m)[s] \downarrow$, we mean that the computation of $\tilde{\varphi}_e((\sigma_i)_{i<k})(m)$ halts by stage s.
   a. If no, we go to the next stage $s + 1$ after setting $\mathtt{state}(e, s) = \mathtt{state}(e, s-1)$ and $T_e \cap \{0,1\}^s = \{0,1\}^s$.
   b. If yes, go to item (3).

3. Ask whether the image of the product of k many closed sets generated by Opp's k-tuple $\mathbf{t}^e$ under the map φ_e covers the whole space $\{0,1\}^{\mathbb{N}}$. Formally speaking, let us consider the following set:

$$\varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^l] \restriction p := \{\varphi_e((\sigma_i)_{i<k}) \restriction p : (\forall i < k) \, \sigma_i \in \mathrm{Tr}(t^e_i) \cap \{0,1\}^l\},$$

and then ask whether $\tau \in \varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}] \restriction q + 1$ for all $\tau \in \{0,1\}^{q+1}$.
   a. If no, choose a witness τ, and we finish the construction by setting $\mathtt{state}(e, s) = \mathtt{end}$ after defining T_e as a tree having a unique infinite path $\tau^\frown 0^\infty := \tau^\frown 000\ldots$.
   b. If yes, go to item (4).

4. Ask whether $\psi_e((p_i)_{i<k})$ already computes some value $j \in \{0, 1\}$ for any k-tuple of paths p_i through Opp's a.o.u. trees t_i^e, that is,

$$(\forall i < k)(\forall(\sigma_i)_{i<k})\left[((\forall i < k)\ \sigma_i \in \mathrm{Tr}(t_i^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}) \ \rightarrow \ \psi_e(\sigma)[s]\downarrow\right].$$

  a. If no, go to the next stage $s + 1$ after setting $\mathtt{state}(e, s) = \mathtt{state}(e, s - 1)$ and $T_e \cap \{0,1\}^s = \{0,1\}^s$.

  b. If yes, let $D_{e,s} = \{\sigma \in \mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)} : \varphi_e(\sigma) \succeq 0^q 1\}$. Note that $D_{e,s} \neq \emptyset$ since we answered yes in item (3); therefore $\varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}] \restriction q + 1 = \{0,1\}^{q+1}$. If $i \notin \psi_e[D_{e,s}]$ for some $i \in \{0, 1\}$, then remove $1 - i$ from $S_e(0^q 1)$ (hence, we have $S_e(0^q 1) = \{i\}$). If $\psi_e[D_{e,s}] = \{0,1\}$ then remove 0 from $S_e(0^q 1)$. In these cases, set $\mathtt{state}(e, s) = q + 1$ and $T_e \cap \{0,1\}^s = \{0,1\}^s$.

Eventually, T_e is constructed as an a.o.u. tree, and $S_e(x) \in \mathrm{dom}(\mathsf{C}_2)$.

▶ **Claim.** *Assume that $(t_i^e)_{i<k}$ determines a k-tuple of a.o.u. trees. Then, there is a realizer G of $\mathrm{AoUC}_{[0,1]}^k$ such that $(\varphi_e \circ G((t_i^e)_{i<k}), \psi_e \circ G((t_i^e)_{i<k}))$ is not a solution to $Z_1 \circ Z_0(T_e, S_e)$, that is, $\varphi_e \circ G((t_i^e)_{i<k}) \notin [T_e]$ or otherwise $\psi_e \circ G((t_i^e)_{i<k}) \notin S_e \circ \varphi_e \circ G((t_i^e)_{i<k})$, where $[T_e]$ denotes the set of all infinite paths through T_e.*

Proof. Assume that $\mathbf{t}^e = (t_i^e)_{i<k}$ determines a k-tuple of a.o.u. trees. In this case, t_i^e is a total tree for each $i < k$. Suppose for the sake of contradiction that the conclusion fails (that is, Opp wins). Then φ_e and ψ_e are defined on all tuples of infinite paths through $\mathrm{Tr}(t_i^e)$, $i < k$. Since $q = 0$ at first, the condition in item (1) is automatically fulfilled. Note that since t_i^e is a total a.o.u. tree, the value $\mathrm{aou}(e, s)$ tends to infinity as $s \to \infty$. Therefore, since φ_e is defined on all paths of Opp's trees, by compactness, the condition in item (2) is also satisfied at some stage s. If $\tau \notin \varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}] \restriction q + 1$, then T_e has a unique infinite path $\tau^\frown 0^\infty$; therefore $\varphi_e \circ G(\mathbf{t}^e) \notin [T_e]$ for any realizer G, which contradicts our assumption. Therefore, $\varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}] \restriction q + 1 = \{0,1\}^{q+1}$. By compactness, the condition in item (4) is eventually satisfied. In any cases, for some $\sigma = (\sigma_i)_{i<k} \in D_{e,s}$, $\psi_e(\sigma) \notin S_e(\varphi_e(\sigma))$ by our construction. In order for Opp to win this game, Opp has to declare that σ_i for some $i < k$ is not extendible to an infinite path through t_i^e. Consequently, under our assumption that Opp wins, Pro's strategy forces such t_i^e not to be the full binary tree; therefore t_i^e has only one path since t_i^e is an a.o.u. tree. Then we continue the same argument with $q = 1$. We can still satisfy the condition in item (1) at some stage since we know at most one tree t_i^e has only one path. Eventually, this construction forces that any of t_i^e has only one path. Then, however, it is impossible to satisfy $\varphi_e[\mathrm{Tr}(\mathbf{t}^e) \cap \{0,1\}^{\mathrm{aou}(e,s)}] \restriction q + 1 = \{0,1\}^{q+1}$. ◀

Suppose for the sake of contradiction that $Z_1 \circ Z_0 \leq_{\mathrm{W}} \mathrm{AoUC}_{[0,1]}^k$ holds via computable H and $K = \langle K_0, K_1 \rangle$, i.e., given (T, S), for any k-tuple $\mathbf{p}$ of infinite paths through trees $H(T, S) = \{H_i(T, S)\}_{i<k}$, $(K_0(\mathbf{p}, T, S), K_1(\mathbf{p}, T, S)) \in Z_1 \circ Z_0(T, S)$, that is, $K_0(\mathbf{p}, T, S) \in [T]$ and $K_1(\mathbf{p}, T, S) \in S(K_0(\mathbf{p}, T, S))$. Choose a computable function f such that $\mathbf{t}^{f(e)} = H(T_e, S_e)$, $\varphi_{f(e)} = \lambda \mathbf{p}.K_0(\mathbf{p}, T_e, S_e)$, and $\psi_{f(e)} = \lambda \mathbf{p}.K_1(\mathbf{p}, T_e, S_e)$. By Kleene's recursion theorem, there is r such that $(\mathbf{t}^{f(r)}, \varphi_{f(r)}, \psi_{f(r)}) = (\mathbf{t}^r, \varphi_r, \psi_r)$. This triple clearly satisfies the premise of the above claim. The realizer G in the claim witnesses the failure of $Z_1 \circ Z_0 \leq_{\mathrm{W}} \mathrm{AoUC}_{[0,1]}^k$ via H and K. Consequently, $\mathsf{LLPO} \star \mathrm{AoUC}_{[0,1]} \not\leq_{\mathrm{W}} \mathrm{AoUC}_{[0,1]}^k$ for all $k \in \mathbb{N}$. ◀

We point out that the preceding theorem relativizes, i.e. even provides a separation w.r.t. continuous Weihrauch reductions. The same holds for all other separation results in this article.

▶ **Corollary 12.** *$\mathsf{LLPO} \star \mathrm{AoUC}_{[0,1]} \not\leq_W \mathrm{AoUC}_{[0,1]}^*$.*

Proof. Assume to the contrary that $C_{\{0,1\}} \star AoUC_{[0,1]} \leq_W AoUC^*_{[0,1]}$. Consider as input to $LLPO \star AoUC_{[0,1]}$ the set $[0,1]$ together with the constant function $h : [0,1] \to \mathcal{A}(\{0,1\})$, $x \mapsto \{0,1\}$. The latter can be represented in such a way that it shares arbitrarily long prefixes with names for any other continuous function of that type. The reduction has to chose some $k \in \mathbb{N}$ eventually that serves as the first component of the derived input to $AoUC^*_{[0,1]}$. But since the original input can still be altered to any other suitable input, this would imply $LLPO \star AoUC_{[0,1]} \leq_W AoUC^k_{[0,1]}$, thus contradicting Theorem 11. ◄

▶ **Corollary 13.** rDiv *is not finitely concurrent.*

Proof. By Proposition 8, $rDiv \equiv_W AoUC_{[0,1]}$ and by Proposition 6 $LLPO \equiv_W C_{\{0,1\}} \leq_W rDiv$. Thus, Corollary 12 implies $rDiv \star rDiv \not\leq_W rDiv^*$. ◄

We will find next that rDiv only barely fails being finitely concurrent: While some amount of nesting is required to obtain the full power of finitely many uses of rDiv, nesting depths 2 already suffices. This result will be proven via a number of individual technical contributions.

Let $\mathcal{O}(\mathbb{N})$ denote subsets of $\mathbb{N}$ represented via an enumeration of their elements. Call a set $A \subseteq \mathcal{O}(\mathbb{N})$ *nice*, if $\emptyset \in A$ and A contains a computable dense sequence $(a_n)_{n \in \mathbb{N}}$.

▶ **Proposition 14.** *Let* $f :\subseteq \mathcal{O}(\mathbb{N}) \rightrightarrows \mathbf{X}$ *have a nice domain and a computable closed graph. Then* $f \star AoUC^k_{[0,1]} \leq_W C_{\{1,\dots,2^k\}} \star \left(f^{2^k} \times AoUC^k_{[0,1]} \right)$.

Proof. For any subset $I \subseteq \{1,\dots,k\}$ we compute an input to f under the assumption that the components $i \in I$ for $AoUC^k_{[0,1]}$ are singletons, and the components $i \notin I$ are the whole interval. We start with providing a name for $\emptyset \in \mathrm{dom}(f)$ and wait until all components $i \in I$ have started to collapse. Then we can compute the actual values in those singletons, and can attempt to compute the input to f associated with those values, together with $0 \in [0,1]$ for those components $i \notin I$. Before actually fixing any values, we make sure that there is some element a_n of the dense sequence extending the current finite prefix. If we ever find that some component $i \notin I$ is starting to collapse, we abandon the attempt to find the correct input to f, and just extend the current prefix to some suitable a_n. By the assumption that $\mathrm{dom}(f)$ is nice, this is guaranteed to produce a valid input to f, and if I was indeed the correct choice, will be the correct input.

Now we consider the output of f on each of these values, together with the output$(x_1,\dots,x_k)$ of $AoUC^k_{[0,1]}$ on the original input. We replace those x_i with $i \notin I$ with 0, and ask whether this is still a correct output. As the graph of f is a computable closed set, we can ask whether this output matches the input to f obtained from the so modified output of $AoUC^k_{[0,1]}$. We can compute a truth value $t_I \in \mathbb{S}$ which is false iff both questions answer to true. If I was indeed correct, the corresponding t_I will be false. If t_I is false, then the combined outputs of f and $AoUC^k_{[0,1]}$ allow us to solve the original question to $f \star AoUC^k_{[0,1]}$. We can use $C_{\{1,\dots,2^k\}}$ to pick some false t_I. ◄

▶ **Corollary 15.** $AoUC^l_{[0,1]} \star AoUC^m_{[0,1]} \leq_W C_{\{1,\dots,2^m\}} \star AoUC^{l2^m+m}_{[0,1]}$.

Proof. We just need to argue that $AoUC_{[0,1]}$ is equivalent to some $f :\subseteq \mathcal{O}(\mathbb{N}) \rightrightarrows \mathbf{X}$ satisfying the criteria of Proposition 14. Recall that $A \in \mathcal{A}([0,1]) \supseteq \mathrm{dom}(AoUC_{[0,1]})$ is represented by enumerating rational open balls exhausting the complement of A. By letting f be equal to $AoUC_{[0,1]}$, but acting on the enumerations rather than the sets themselves, we have found the required candidate. ◄

▶ **Proposition 16.** *Let* $f :\subseteq \mathcal{O}(\mathbb{N}) \rightrightarrows \mathbf{X}$ *have a nice domain. Then* $f \star C_{\{1,\dots,n\}} \leq_W f^n \times C_{\{1,\dots,n\}}$.

Proof. For each $i \in \{1, \ldots, n\}$ we attempt to compute the suitable input to f if i were the output provided by $C_{\{1,\ldots,n\}}$. We only actually write a finite prefix of the output once we have found an element a_n extending it. If we ever learn that i is not a correct output of $C_{\{1,\ldots,n\}}$, we abandon the attempt and simply extend the current input to f to some a_n. The nice domain of f ensures that this procedure results in a valid input for f. If we do this for all choices of i in parallel, and also compute a suitable i, we can then read of a correct output to $f \star C_{\{1,\ldots,n\}}$. ◄

▶ **Corollary 17.** $AoUC_{[0,1]}^l \star C_{\{1,\ldots,m\}} \leq_W AoUC_{[0,1]}^{lm+m-1}$.

Proof. To argue that we may use $AoUC_{[0,1]}$ in place of f in Proposition 16, we argue as we did to obtain Corollary 15 from Proposition 14. Now $C_{\{1,\ldots,m\}} \leq_W C_{\{0,1\}}^{m-1}$ and $C_{\{0,1\}} \leq_W AoUC_{[0,1]}$ from [17] complete the argument. ◄

So we do find that 3 (or more) consecutive applications of powers of $AoUC_{[0,1]}$ do reduce to 2:

▶ **Corollary 18.** $AoUC_{[0,1]}^l \star AoUC_{[0,1]}^m \star AoUC_{[0,1]}^k \leq_W AoUC_{[0,1]}^{(l+1)2^k-1} \star AoUC_{[0,1]}^{m2^k+k}$

▶ **Proposition 19.** Let $f : \mathbf{X} \rightrightarrows \mathbb{N}$ be such that $n \in f(x) \land m > n \Rightarrow m \in f(x)$. Then if $f \leq_W C_{\{0,1\}^{\mathbb{N}}}$, f is already computable.

▶ **Corollary 20.**

$$AoUC_{[0,1]}^* \star AoUC_{[0,1]}^* \qquad\qquad \equiv_W \coprod_{n \in \mathbb{N}} (AoUC_{[0,1]}^n \star AoUC_{[0,1]}^n)$$

$$\equiv_W \coprod_{n \in \mathbb{N}} AoUC_{[0,1]}^{(n)} \equiv_W C_{\{0,1\}}^* \star AoUC_{[0,1]}^* \qquad \equiv_W \left(AoUC_{[0,1]}^*\right)^{(n+1)}$$

Proof. By Proposition 19 it follows that in e.g. $AoUC_{[0,1]}^* \star AoUC_{[0,1]}^*$ the number of oracle calls made in the second round can be bounded in advance. The equivalences now follow from the uniform versions of Corollaries 15, 18. ◄

5 Gaussian Elimination

Most work on algorithms in linear algebra assumes equality to be decidable, and is thus applicable to computability over the rational or algebraic numbers, but not to computability over the real numbers. In the latter setting, computability of some basic questions (rank, eigenvectors,...) was studied in [30], with some additional results in [29, 10]. Here, we shall consider LU-decomposition and Gaussian elimination.

Gaussian elimination is one of the basic algorithms in linear algebra, used in particular to compute the LU-decomposition of matrices. The goal is to transform a given matrix into row echelon form by means of swapping rows (and maybe columns) and adding multiples of one row to another. Sometimes the leading non-zero coefficients in each row are required to be 1, however, as this is easily seen to require equality testing, we shall not include this requirement.

▶ **Definition 21.** LU-Decomp$_{P,Q}$ takes as input a matrix A, and outputs permutation matrices P, Q, a matrix U in upper echelon form and a matrix L in lower echelon form with all diagonal elements being 1 such that $PAQ = LU$. By LU-Decomp$_Q$ we denote the extension where P is required to be the identity matrix.

▶ **Theorem 22.** $LU\text{-}Decomp_{P,Q} \equiv_W \mathrm{rDiv}^*$ *and* $\mathrm{rDiv}^* \leq_W LU\text{-}Decomp_Q \leq_W \mathrm{rDiv}^* \star \mathrm{rDiv}^*$.

The proof of the preceding theorem follows in form of some lemmata. We point out that the upper bounds are proven via variants of Gaussian elimination. In the case of LU-Decomp$_{P,Q}$ and its matching lower bound, this shows that Gaussian elimination exhibits no more incomputability than inherent in the problem it solves. It is consistent with the classifications that the extra freedom in choosing the pivot elements in solving LU-Decomp$_{P,Q}$ compared to solving LU-Decomp$_Q$ makes the problem less incomputable. Resolving the precise degree of LU-Decomp$_Q$ seems to be beyond the reach of our current methods though.

▶ **Lemma 23.** $LU\text{-}Decomp_{P,Q} \equiv_W LU\text{-}Decomp_{P,Q}^*$ *and* $LU\text{-}Decomp_Q \equiv_W LU\text{-}Decomp_Q^*$.

Proof. An LU-decomposition of $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$ gives rise to LU-decompositions of both A and B. ◀

▶ **Lemma 24.** $LU\text{-}Decomp_{P,Q} \leq_W \mathrm{rDiv}^*$.

Proof. Initially, we rearrange all the matrix elements such that at each step the pivot element chosen has the largest absolute value amongst the remaining elements, and obtain their signs. This can be achieved by $\mathrm{C}_{\{0,\ldots,k\}}$ for suitable k. Then we can compute all the relevant divisions simultaneously, using some rDiv^l. Corollary 17 then shows that this reduces to rDiv^*. ◀

▶ **Lemma 25.** $LU\text{-}Decomp_Q \leq_W \mathrm{rDiv}^* \star \mathrm{rDiv}^*$.

Proof. Given some real matrix $(a_{ij})_{i\leq n, j\leq m}$ we can use $\mathrm{C}_{\{0,\ldots,n-1\}}$ to pick some i_0 such that $|a_{i_0,1}| = \max_{i\leq n} |a_{i,1}|$, and permute the rows to move the i_0-th row to the top. We can use $C_{\{0,1\}}^n$ to figure out for each i whether $|a_{i,1}|$ is non-negative or non-positive. For each $i \neq i_0$ we compute $\mathrm{rDiv}(|a_{i,1}|, |a_{i_0,1}|)$, pick the sign depending on the putative signs on $a_{i,1}$ and $a_{i_0,1}$ and then subtract the corresponding multiple of the i_0-th row from the i-th row. By choice of i_0, either all $a_{i,1}$ are 0 anyway, or $a_{i_0,1} \neq 0$ – in both cases, this ensures that in all rows but the i_0-th the first entry is zero after the operation.

The procedure so far made use of $\mathrm{rDiv}^{n-1} \star \left(\mathrm{C}_{\{0,\ldots,n-1\}} \times C_{\{0,1\}}^n \right)$.

After the first round, the now first row is fixed. Amongst the remaining ones, we pick one with the largest absolute value in the second column (using $\mathrm{C}_{\{0,\ldots,n-2\}}$), determine the signs of entries in the second column (using $C_{\{0,1\}}^{n-2}$) and again use rDiv to compute the coefficients for subtracting the second row from the lower ones.

This is repeated until each row has been dealt with. Overall, we use $n-1$ rounds, so the procedure is reducible to $\left[\mathrm{rDiv}^{n-1} \star \left(\mathrm{C}_{\{0,\ldots,n-1\}} \times C_{\{0,1\}}^n \right)\right]^{(n)}$. By repeated application of Corollaries 17,18 this reduces to $\mathrm{AoUC}_{[0,1]}^k \star \mathrm{AoUC}_{[0,1]}^k$ for sufficiently big k (depending effectively on n). ◀

▶ **Lemma 26.** $\mathrm{rDiv} \leq_W LU\text{-}Decomp_{P,Q}$.

Proof. We consider the computable function $B : [0,1] \to \mathbb{R}^{2\times 2}$ of matrices defined via:

$$B(\varepsilon) = \exp(-\varepsilon^{-2}) \begin{pmatrix} \cos(\varepsilon^{-1}) & \sin(\varepsilon^{-1}) \\ -\sin(\varepsilon^{-1}) & \cos(\varepsilon^{-1}) \end{pmatrix} \text{ for } \varepsilon > 0 \quad B(0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

This is based on a counterexample due to RELLICH (cmp. [14, II.5.3], [30, Example 18]). If $\varepsilon \neq 0$, then the lower-left corner of L in an LU-decomposition of $B(\varepsilon)$ will be one of $\tan \varepsilon^{-1}$,

$\cot \varepsilon^{-1}$ or $-\cot \varepsilon^{-1}$. The relevant case can be obtained from P and Q. As arctan and arccot are total, we can apply the relevant inverse even if $\varepsilon = 0$, and thus the lower-left corner of L is an arbitrary real number. Let x'_ε be the result, and $x_\varepsilon = \max\{0, \min\{1, x'_\varepsilon\}\}$.

We want to show that $\text{AoUC}_{[0,1]} \leq_{\text{W}} \text{LU-Decomp}_{P,Q}$ (which is equivalent to the claim by Proposition 8). Given $A \in \text{dom}(\text{AoUC}_{[0,1]})$, we show how to compute some $\varepsilon \in [0,1]$ such that $x_\varepsilon \in A$. As long as $A = [0,1]$ is consistent with our knowledge of the input, we specify that $\varepsilon \in [0, 2^{-t}]$ for smaller and smaller $t \in \mathbb{N}$. If we learn at time t that $A \neq [0,1]$, we compute y such that $A = \{y\}$ and choose $k \in \mathbb{N}$ such that $(2k\pi)^{-k} \leq 2^{-t}$. We can then specify $\varepsilon = (2k\pi + y)^{-1}$. But now $x_\varepsilon = y$. If $A = [0,1]$, then $x_\varepsilon \in A$ anyway by definition. ◄

Based on the preceding lemma and [30, Example 18], we can also see that rDiv is reducible to finding eigenvectors of a matrix.

—— **References** ——

1 Jeremy Avigad and Vasco Brattka. Computability and analysis: the legacy of Alan Turing. In Rod Downey, editor, *Turing's Legacy*, volume 42 of *Lecture Notes in Logic*, pages 1–47. Cambridge University Press, 2014. available at http://arxiv.org/abs/1206.3431.

2 Émile Borel. Le calcul des intégrales définies. *Journal de Mathématiques pures et appliquées*, 6(8):159–210, 1912.

3 Vasco Brattka, Matthew de Brecht, and Arno Pauly. Closed choice and a uniform low basis theorem. *Annals of Pure and Applied Logic*, 163(8):968–1008, 2012. `doi:10.1016/j.apal.2011.12.020`.

4 Vasco Brattka and Guido Gherardi. Effective choice and boundedness principles in computable analysis. *Bulletin of Symbolic Logic*, 1:73–117, 2011. arXiv:0905.4685. `doi:10.2178/bsl/1294186663`.

5 Vasco Brattka and Guido Gherardi. Weihrauch degrees, omniscience principles and weak computability. *Journal of Symbolic Logic*, 76:143–176, 2011. arXiv:0905.4679.

6 Vasco Brattka, Guido Gherardi, and Rupert Hölzl. Probabilistic computability and choice. *Information and Computation*, 242:249–286, 2015. arXiv 1312.7305. `doi:10.1016/j.ic.2015.03.005`.

7 Vasco Brattka, Guido Gherardi, and Alberto Marcone. The Bolzano-Weierstrass Theorem is the jump of Weak König's Lemma. *Annals of Pure and Applied Logic*, 163(6):623–625, 2012. also arXiv:1101.0792. `doi:10.1016/j.apal.2011.10.006`.

8 Vasco Brattka, Akitoshi Kawamura, Alberto Marcone, and Arno Pauly. Measuring the Complexity of Computational Content (Dagstuhl Seminar 15392). *Dagstuhl Reports*, 5(9):77–104, 2016. `doi:10.4230/DagRep.5.9.77`.

9 Vasco Brattka, Stéphane Le Roux, and Arno Pauly. On the computational content of the Brouwer fixed point theorem. In S.Barry Cooper, Anuj Dawar, and Benedikt Löwe, editors, *How the World Computes*, volume 7318 of *Lecture Notes in Computer Science*, pages 56–67. Springer Berlin Heidelberg, 2012. `doi:10.1007/978-3-642-30870-3_7`.

10 Vasco Brattka and Arno Pauly. Computation with advice. *Electronic Proceedings in Theoretical Computer Science*, 24, 2010. CCA 2010. URL: `http://arxiv.org/html/1006.0551`.

11 Vasco Brattka and Arno Pauly. On the algebraic structure of Weihrauch degrees. arXiv 1604.08348, 2016. URL: `http://arxiv.org/abs/1604.08348`.

12 Guido Gherardi and Alberto Marcone. How incomputable is the separable Hahn-Banach theorem? *Notre Dame Journal of Formal Logic*, 50(4):393–425, 2009. `doi:10.1215/00294527-2009-018`.

13 Kojiro Higuchi and Arno Pauly. The degree-structure of Weihrauch-reducibility. *Logical Methods in Computer Science*, 9(2), 2013. `doi:10.2168/LMCS-9(2:2)2013`.

14 Tosio Kato. *Pertubation Theory for Linear Operators*. Springer, 1976.

15   Takayuki Kihara and Arno Pauly.   Dividing by zero – how bad is it, really?
      arXiv:1606.04126, 2016.

16   Stéphane Le Roux and Arno Pauly. Finite choice, convex choice and finding roots. *Logical
      Methods in Computer Science*, 2015.   URL: `http://arxiv.org/abs/1302.0380`, `doi:10.`
      `2168/LMCS-11(4:6)2015`.

17   Arno Pauly. How incomputable is finding Nash equilibria? *Journal of Universal Computer
      Science*, 16(18):2686–2710, 2010. `doi:10.3217/jucs-016-18-2686`.

18   Arno Pauly. On the (semi)lattices induced by continuous reducibilities. *Mathematical Logic
      Quarterly*, 56(5):488–502, 2010. `doi:10.1002/malq.200910104`.

19   Arno Pauly. *Computable Metamathematics and its Application to Game Theory*. PhD
      thesis, University of Cambridge, 2012.

20   Arno Pauly. Many-one reductions and the category of multivalued functions. *Math-
      ematical Structures in Computer Science*, 2015. available at: arXiv 1102.3151. `doi:`
      `10.1017/S0960129515000262`.

21   Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*,
      2016. accepted for publication, available at http://arxiv.org/abs/1204.3763. `doi:10.3233/`
      `COM-150049`.

22   Arno Pauly and Florian Steinberg. Representations of analytic functions and weihrauch
      degrees. In *Proceedings of* Computer Science Russia *(CSR)*, volume 9691 of *LNCS*, pages
      367–381, 2016. `doi:10.1007/978-3-319-34171-2_26`.

23   Arno Pauly and Martin Ziegler. Relative computability and uniform continuity of relations.
      Journal of Logic and Analysis, 5, 2013.

24   Matthias Schröder. Extended admissibility. *Theoretical Computer Science*, 284(2):519–538,
      2002. `doi:10.1016/S0304-3975(01)00109-8`.

25   Alan Turing. On computable numbers, with an application to the Entscheidungsproblem:
      Corrections. *Proceedings of the LMS*, 2(43):544–546, 1937.

26   Klaus Weihrauch. The degrees of discontinuity of some translators between representations
      of the real numbers. Informatik Berichte 129, FernUniversität Hagen, Hagen, 1992.

27   Klaus Weihrauch. The TTE-interpretation of three hierarchies of omniscience principles.
      Informatik Berichte 130, FernUniversität Hagen, Hagen, 1992.

28   Klaus Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

29   Martin Ziegler. Real computation with least discrete advice: A complexity theory of nonuni-
      form computability with applications to effective linear algebra. *Annals of Pure and Applied
      Logic*, 163(8):1108–1139, 2012. `doi:10.1016/j.apal.2011.12.030`.

30   Martin Ziegler and Vasco Brattka. Computability in linear algebra. *Theoretical Computer
      Science*, 326:187–211, 2004.