# New ways to multiply $3 \times 3$-matrices[1]

Marijn J.H. Heule

*Computer Science Department, Carnegie Mellon University, Pittsburgh PA, USA*

Manuel Kauers

*Institute for Algebra, J. Kepler University Linz, Austria*

Martina Seidl

*Institute for Formal Models and Verification, J. Kepler University Linz, Austria*

**Abstract**

It is known since the 1970s that no more than 23 multiplications are required for computing the product of two $3 \times 3$-matrices. For non-commutative coefficient rings, it is not known whether it can also be done with fewer multiplications. However, there are several mutually inequivalent ways of doing the job with 23 multiplications. In this article, we extend this list considerably by providing more than 17,000 new and mutually inequivalent schemes for multiplying $3 \times 3$-matrices using 23 multiplications. Moreover, we show that the set of all these schemes is a manifold of dimension at least 17.

*Keywords:* bilinear complexity, matrix multiplication, Laderman's algorithm, SAT solving.

## 1. Introduction

The classical algorithm for multiplying two $n \times n$ matrices performs $2n^3 - n^2$ additions and multiplications. Strassen's algorithm (Strassen, 1969) does the job with only $O(n^{\log_2 7})$ arithmetic operations, by recursively applying a certain scheme for computing the product of two $2 \times 2$-matrices with only 7 instead of the usual 8 multiplications. The discovery of Strassen's algorithm has initiated substantial work during the past 50 years on finding the smallest exponent $\omega$ such that matrix multiplication costs $O(n^\omega)$ operations in the coefficient ring. The current record is $\omega \leq 2.3728639$ and was obtained by Le Gall (2014). It improves the previous record of Vassilevska Williams (2012) by just $3 \cdot 10^{-7}$. Extensive background in this direction is available in text books (Bürgisser et al., 2013; Landsberg, 2017) and survey articles (Bläser, 2013; Pan, 2018). Contrary to wide-spread belief, Strassen's algorithm is not only efficient in theory

[1]M.J.H. Heule was supported by NSF under grant CCF-2006363. M. Kauers was supported by the Austrian FWF grants P31571-N32 and F5004. M. Seidl was supported by the Austrian FWF grant NFN S11408-N23 and the LIT AI Lab funded by the State of Upper Austria.

*Email addresses:* marijn@cmu.edu (Marijn J.H. Heule), manuel.kauers@jku.at (Manuel Kauers), martina.seidl@jku.at (Martina Seidl)

but also in practice. Special purpose software for exact linear algebra, such as the FFLAS and FFPACK packages (Dumas et al., 2008), have been using it since long, and there are also reports that its performance in a numerical context is not as bad as its reputation (Huang et al., 2016).

Besides the quest for the smallest exponent, which only concerns the asymptotic complexity for asymptotically large $n$, it is also interesting to know how many multiplications are needed for a specific (small) $n$ to compute the product of two $n \times n$-matrices. Thanks to Strassen, we know that the answer is at most 7 for $n = 2$, and it can be shown (Winograd, 1971) that there is no way to do it with 6 multiplications. It can further be shown that, in a certain sense, Strassen's scheme is the only way of doing it with 7 multiplications (de Groote, 1978).

Already for $n = 3$, the situation is not completely understood. Laderman (1976) showed that 23 multiplications suffice, and Bläser (2003) showed that at least 19 multiplications are needed. There is also a way to do it with only 22 multiplications (**?**), but this scheme requires the assumption that the coefficient ring is commutative, which prevents it from being applied recursively. In this paper, we restrict the attention to schemes that work over non-commutative coefficient domains, and for this case, no method using less than 23 is currently known. For larger sizes as well as rectangular matrices, many people have been searching for new schemes using fewer and fewer coefficient multiplications. For $n = 4$, the best we know is to apply Strassen's scheme recursively, which requires 49 multiplications. For $n = 5$, the record of 100 multiplications was held Makarov (1987) for 30 years until it was improved, first to 99 by Sedoglavic (2017b), and shortly later to 98 (see, e.g., Sedoglavic, 2019). For $n = 6$, there is a recent scheme by Smirnov (2013) which needs only 160 multiplications. For $n = 7$, Sedoglavic (2017c) found a way to compute the product with 250 multiplications. For larger sizes and rectangular matrices, see the extensive tables compiled by Smirnov (2013, 2017) and Sedoglavic (2019). Many of the schemes for larger matrix sizes are obtained by combining multiplication schemes for smaller matrices (Drevet et al., 2011).

Although nobody knows whether there is a scheme using only 22 multiplications for $n = 3$ (in an exact and non-commutative setting), 23 multiplications can be achieved in many different ways. Johnson and McLoughlin (1986) have in fact found infinitely many ways. They presented a family of schemes involving three free parameters. However, their families involve fractional coefficients and therefore do not apply to arbitrary coefficient rings $K$. Many others have reported isolated schemes with fractional or approximate coefficients. Such schemes can be constructed for example by numerically solving a certain optimization problem, or by genetic algorithms. In Laderman's multiplication scheme, all coefficients are $+1$, $-1$, or 0, which has the nice feature that it works for any coefficient ring. As far as we know, there are so far only three other schemes with this additional property, they are due to Smirnov (2013), Oh et al. (2013), and Courtois et al. (2011), respectively. We add more than 17,000 new schemes to this list.

The isolated scheme presented by Courtois et al. was not found numerically but with the help of a SAT solver. SAT (Biere et al., 2009) refers to the decision problem of propositional logic: given a Boolean formula in conjunctive normal form, is there an assignment of the Boolean variables such that the formula evaluates to true under this assignment? Although SAT is a prototypical example of an NP-complete problem, modern SAT solvers are able to solve very large instances. In addition to various industrial applications, they have recently also contributed to the solution of difficult mathematical problems, see Heule et al. (2016) and Heule (2018) for two examples. SAT solvers also play a central role in our approach. As explained in Section 3, we first use a SAT solver to find multiplication schemes for the coefficient ring $\mathbb{Z}_2$, starting from some known solutions. In a second step, explained in Section 4, we discard solutions that are equivalent to solutions found earlier. Next, we simplify the new solutions (Sect. 5), and use them

as starting points for a new round of searching. Altogether about 35 years of computation time were spent in several iterations of this process. These computations were distributed on several nodes of the Lonestar 5 cluster in Austin, Texas, each of which consists of a Xeon E5-2690 v3 chip with 24 cores running at 2.6 GHz and 64 GB memory, as well as several nodes of the Mach cluster in Linz, Austria, each of which consists of a Xeon E7-8837 chip with 8 cores running at 2.6 GHz and 64 GB memory, as well as several smaller stand-alone machines with altogether around 400 cores and 2TB memory. In the end, we lifted the solutions from $\mathbb{Z}_2$ to arbitrary coefficient rings (Sect. 6), and we extracted families with up to 17 free parameters from them (Sect. 7). Our 17,000 isolated schemes and our parameterized families are provided in various formats on our website (Heule et al., 2019b).

## 2. The Brent Equations

The general pattern of a matrix multiplication scheme consists of two sections. In the first section, several auxiliary quantities $M_1, M_2, \ldots, M_m$ are computed, each of which is a product of a certain linear combination of the entries of the first matrix with a certain linear combination of the entries of the second matrix. In the second section, the entries of the resulting matrix are obtained as certain linear combinations of the auxiliary quantities $M_1, M_2, \ldots, M_m$.

For example, writing

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}, \quad B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}, \quad \text{and} \quad C = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix} := AB,$$

Strassen's multiplication scheme proceeds as follows:

*First section.*

$M_1 = (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2})$
$M_2 = (a_{2,1} + a_{2,2})(b_{1,1})$
$M_3 = (a_{1,1})(b_{1,2} - b_{2,2})$
$M_4 = (a_{2,2})(b_{2,1} - b_{1,1})$
$M_5 = (a_{1,1} + a_{1,2})(b_{2,2})$
$M_6 = (a_{2,1} - a_{1,1})(b_{1,1} + b_{1,2})$
$M_7 = (a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2})$

*Second section.*

$c_{1,1} = M_1 + M_4 - M_5 + M_7$
$c_{1,2} = M_3 + M_5$
$c_{2,1} = M_2 + M_4$
$c_{2,2} = M_1 - M_2 + M_3 + M_6.$

Observe that the number of multiplications is exactly the number of $M$'s. Also observe that while it is not obvious how to construct such a scheme from scratch, checking that a given scheme is correct is an easy and straightforward calculation. For example, $c_{2,1} = M_2 + M_4 = (a_{2,1} + a_{2,2})(b_{1,1}) + (a_{2,2})(b_{2,1} - b_{1,1}) = a_{2,1}b_{1,1} + a_{2,2}b_{2,1}$.

In order to search for a multiplication scheme for a prescribed shape of matrices (e.g., $3 \times 3$) and a prescribed number of multiplications (e.g., 23), we can make an ansatz for the coefficients

of the various linear combinations,

$$\begin{aligned}
M_1 &= (\alpha^{(1)}_{1,1}a_{1,1} + \alpha^{(1)}_{1,2}a_{1,2} + \cdots)(\beta^{(1)}_{1,1}b_{1,1} + \beta^{(1)}_{1,2}b_{1,2} + \cdots)\\
M_2 &= (\alpha^{(2)}_{1,1}a_{1,1} + \alpha^{(2)}_{1,2}a_{1,2} + \cdots)(\beta^{(2)}_{1,1}b_{1,1} + \beta^{(2)}_{1,2}b_{1,2} + \cdots)\\
&\vdots\\
M_{23} &= (\alpha^{(23)}_{1,1}a_{1,1} + \alpha^{(23)}_{1,2}a_{1,2} + \cdots)(\beta^{(23)}_{1,1}b_{1,1} + \beta^{(23)}_{1,2}b_{1,2} + \cdots)\\
c_{1,1} &= \gamma^{(1)}_{1,1}M_1 + \gamma^{(2)}_{1,1}M_2 + \cdots + \gamma^{(23)}_{1,1}M_{23}\\
c_{1,2} &= \gamma^{(1)}_{1,2}M_1 + \gamma^{(2)}_{1,2}M_2 + \cdots + \gamma^{(23)}_{1,2}M_{23}\\
&\vdots\\
c_{3,3} &= \gamma^{(1)}_{3,3}M_1 + \gamma^{(2)}_{3,3}M_2 + \cdots + \gamma^{(23)}_{3,3}M_{23}
\end{aligned}$$

and then compare coefficients such as to enforce $c_{i,j} = \sum_k a_{i,k}b_{k,j}$. Doing so leads to a system of polynomial equations for the undetermined coefficients $\alpha^{(\iota)}_{i_1,i_2}, \beta^{(\iota)}_{j_1,j_2}, \gamma^{(\iota)}_{k_1,k_2}$. The equations in this system are known as the Brent equations (Brent, 1970). For 3×3-matrices and 23 multiplications, the equations turn out to be

$$\sum_{\iota=1}^{23} \alpha^{(\iota)}_{i_1,i_2}\beta^{(\iota)}_{j_1,j_2}\gamma^{(\iota)}_{k_1,k_2} = \delta_{i_2,j_1}\delta_{i_1,k_1}\delta_{j_2,k_2}$$

for $i_1, i_2, j_1, j_2, k_1, k_2 \in \{1, 2, 3\}$, i.e., there are 621 variables and 729 cubic equations. The $\delta_{u,v}$ on the right refer to the Kronecker-delta, i.e., $\delta_{u,v} = 1$ if $u = v$ and $\delta_{u,v} = 0$ otherwise.

The equations become a bit more symmetric if we connect the matrices $A, B, C$ through $C^\top = AB$ rather than $C = AB$. In the version with the transposition, which we shall use from now on, and which is also more common in the literature, the right hand side has to be replaced with $\delta_{i_2,j_1}\delta_{j_2,k_1}\delta_{k_2,i_1}$.

In any case, the problem boils down to finding a solution of the Brent equations. In principle, this system could be solved using Gröbner bases (Buchberger, 1965; Cox et al., 1992; Buchberger and Kauers, 2010), but doing so would require an absurd amount of computation time. Some of the solutions reported in the literature have been found using numerical solvers (Smirnov, 2013; Oh et al., 2013), and Laderman (1976) claims that his solution was found by solving the Brent equations by hand. He writes that he would explain in a later paper how exactly he did this, but apparently this later paper has never been written. Only recently, Sedoglavic (2017a) has given a convincing explanation of how Laderman's scheme can be derived from Strassen's scheme for $2 \times 2$ matrices. Courtois et al. (2011) found their solution using a SAT solver. We also start our search using SAT solvers.

## 3. SAT Encoding and Streamlining

In order to encode the problem as a SAT problem, we view the Brent equations as equations for the finite field $\mathbb{Z}_2$, interpret its elements as truth values, its addition as exclusive or ($\oplus$), and its multiplication as conjunction ($\wedge$). These propositional formulas cannot be directly be processed by most state-of-the-art SAT solvers, because they require the formulas in conjunctive normal form (CNF). A formula is in CNF if it is a conjunction of clauses, where a clause is

a disjunction ($\vee$) of literals and a literal is a Boolean variable $x$ or the negation of a Boolean variable ($\bar{x}$). For avoiding an exponential blow-up when transforming an arbitrary structured formula to CNF, auxiliary variables are introduced that abbreviate certain subformulas. For every $i_1, i_2, j_1, j_2 \in \{1, 2, 3\}$ and every $\iota = 1, \ldots, 23$, we introduce a fresh variable $s^{(\iota)}_{i_1,i_2,j_1,j_2}$ and impose the condition

$$s^{(\iota)}_{i_1,i_2,j_1,j_2} \leftrightarrow (\alpha^{(\iota)}_{i_1,i_2} \wedge \beta^{(\iota)}_{j_1,j_2}),$$

whose translation to CNF requires three clauses. Similarly, for every $i_1, i_2, j_1, j_2, k_1, k_2 \in \{1, 2, 3\}$ and every $\iota = 1, \ldots, 23$, we introduce a fresh variable $t^{(\iota)}_{i_1,i_2,j_1,j_2,k_1,k_2}$ and impose the condition

$$t^{(\iota)}_{i_1,i_2,j_1,j_2,k_1,k_2} \leftrightarrow (s^{(\iota)}_{i_1,i_2,j_1,j_2} \wedge \gamma^{(\iota)}_{k_1,k_2}),$$

whose translation to CNF costs again three clauses.

For each fixed choice $i_1, i_2, j_1, j_2, k_1, k_2 \in \{1, 2, 3\}$, there is a Brent equation which says that the number of $\iota$'s for which $t^{(\iota)}_{i_1,i_2,j_1,j_2,k_1,k_2}$ is set to true should be even (if $\delta_{i_2,j_1}\delta_{i_1,k_1}\delta_{j_2,k_2} = 0$) or that it should be odd (if $\delta_{i_2,j_1}\delta_{i_1,k_1}\delta_{j_2,k_2} = 1$). It therefore remains to encode the condition that an even number (or an odd number) of a given set of $p$ variables should be true, i.e., we need to construct a formula $\mathrm{even}(x_1, \ldots, x_p)$ which is true if and only if an even number among the variables $x_1, \ldots, x_p$ is true. Such a formula can again be constructed using auxiliary variables. Note that $\mathrm{even}(x_1, \ldots, x_p)$ is true if and only if $\mathrm{even}(x_1, \ldots, x_i, y) \wedge \mathrm{even}(x_{i+1}, \ldots, x_p, y)$ is true, because this is the case if and only if both $\{x_1, \ldots, x_i\}$ and $\{x_{i+1}, \ldots, x_p\}$ contain an even number of variables set to true (and then $y$ is set to false) or both sets contain an odd number of variables set to true (and then $y$ is set to true). Applying this principle recursively for $p = 23$ (the number of summands in each Brent equation), the problem can be broken down to chunks of size four:

$$\begin{aligned}
&\mathrm{even}(x_1, x_2, x_3, y_1) \wedge \mathrm{even}(x_4, x_5, x_6, y_2) \wedge \mathrm{even}(x_7, x_8, x_9, y_3) \wedge \mathrm{even}(x_{10}, x_{11}, x_{12}, y_4)\\
\wedge\ &\mathrm{even}(x_{13}, x_{14}, x_{15}, y_5) \wedge \mathrm{even}(x_{16}, x_{17}, x_{18}, y_6) \wedge \mathrm{even}(x_{19}, x_{20}, x_{21}, y_7) \wedge \mathrm{even}(x_{22}, x_{23}, y_1, y_8)\\
\wedge\ &\mathrm{even}(y_2, y_3, y_4, y_9) \wedge \mathrm{even}(y_5, y_6, y_7, y_{10}) \wedge \mathrm{even}(y_8, y_9, y_{10}, y_{11}).
\end{aligned}$$

The small chunks can be encoded directly by observing that $\mathrm{even}(a, b, c, d)$ is equivalent to

$$\begin{aligned}
&(a \vee b \vee c \vee \bar{d}) \wedge (a \vee b \vee \bar{c} \vee d) \wedge (a \vee \bar{b} \vee c \vee d) \wedge (\bar{a} \vee b \vee c \vee d) \wedge\\
&(a \vee \bar{b} \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee \bar{b} \vee c \vee \bar{d}) \wedge (\bar{a} \vee b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee \bar{b} \vee \bar{c} \vee d).
\end{aligned}$$

For the cases where an odd number of the variables $x_1, \ldots, x_{23}$ must be true, we can apply the encoding described above to $\mathrm{even}(\bar{x}_1, x_2, x_3, \ldots, x_{23})$.

The SAT problems obtained in this way are very hard. In order to make the problems more tractable, we added further constraints in order to simplify the search performed by the solver. This approach is known as streamlining (Gomes and Sellmann, 2004). The following restrictions turned out to be successful:

- Instead of a faithful encoding of the sums in the Brent equations using the even predicate as described above, we also used a more restrictive sufficient condition which instead of requiring an even number of arguments to be true enforces that zero or two arguments should be true. This predicate zero-or-two can be broken into at-most-two and not-exactly-

one, which can be efficiently encoded as

$$\begin{aligned}
\text{not-exactly-one}(x_1, \ldots, x_p) &= \bigwedge_{i=1}^{p} \Big( x_i \to \bigvee_{j \neq i} x_j \Big) \\
\text{at-most-two}(x_1, \ldots, x_p) &= (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \\
&\quad \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \\
&\quad \wedge (\bar{x}_1 \vee y) \wedge (\bar{x}_2 \vee y) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee z) \\
&\quad \wedge (\bar{x}_3 \vee z) \wedge (\bar{x}_4 \vee z) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee y) \\
&\quad \wedge \text{at-most-two}(y, z, x_5, \ldots, x_p),
\end{aligned}$$

where $y$ and $z$ are fresh variables. The first two lines of at-most-two assert that at most two variables of $x_1, x_2, x_3, x_4$ are true. If two or more of those variables are true then the new variables $y$ and $z$ have to be both true, if one variable is true, then either $y$ or $z$ has to be true, and if all four variables are false, then also $y$ and $z$ can be both false. Encoding this information in $y$ and $z$ allows to recursively apply at-most-two with two arguments less. A straightforward direct encoding as in the first two lines is used when $p \leq 4$.

- We selected a certain portion, say 50%, of the variables $\alpha^{(\iota)}_{i_1,i_2}$, $\beta^{(\iota)}_{j_1,j_2}$, $\gamma^{(\iota)}_{k_1,k_2}$ and instantiate them with the values they have in one of the known solutions. The SAT solver then has to solve for the remaining variables. It turns out that in many cases, it does not just rediscover the known solution but finds a truly different one that only happens to have an overlap with the original solution.

- Another approach was to randomly set half of the terms $\alpha^{(\iota)}_{i_1,i_2}\beta^{(\iota)}_{j_1,j_2}\gamma^{(\iota)}_{k_1,k_2}$ with $i_2 \neq j_1$ and $j_2 \neq k_1$ and $k_2 \neq i_1$ to zero. This strategy was motivated by the observation that in most of the known solutions, almost all these terms are zero.

- A third approach concerns the terms $\alpha^{(\iota)}_{i_1,i_2}\beta^{(\iota)}_{j_1,j_2}\gamma^{(\iota)}_{k_1,k_2}$ with $i_2 = j_1$ and $j_2 = k_1$ and $k_2 = i_1$. Again motivated by the inspection of known solutions, we specified that for each $\iota$ either one or two such terms should be one. More precisely, we randomly chose a distribution of the 27 terms with $i_2 = j_1$ and $j_2 = k_1$ and $k_2 = i_1$ to the 23 summands of the scheme, with the condition that 19 summands should contain one term each and the remaining four summands should contain two terms each.

Each of the latter three approaches was used in combination with both the 'even' and the 'zero-or-two' encoding of the Brent equations. The resulting instances were presented to the SAT solver yalsat by Biere (2018). When it didn't find a solution for an instance within a few minutes, the instance was discarded and a new instance with another random choice was tried. A detailed analysis of the effect of our optimizations on the performance of the solver is provided in a separate paper (Heule et al., 2019a).

## 4. Recognizing Equivalences

From any given solution of the Brent equations we can generate many equivalent solutions. For example, exchanging $\alpha$ with $\beta$ and flipping all indices maps a solution to another solution. This operation corresponds to the fact that $(AB)^\top = B^\top A^\top$. It is also clear from the equations that replacing $\alpha$ by $\beta$, $\beta$ by $\gamma$, and $\gamma$ by $\alpha$ maps a solution to another solution, although this operation

is less obvious in terms of matrix multiplication. Finally, for any fixed invertible matrix $U$, we can exploit the fact $AB = AUU^{-1}B$ to map solutions to other solutions.

The operations just described form a group of symmetries of matrix multiplication which was introduced by de Groote (1978), who used them for showing that Strassen's scheme for $2\times 2$ matrices is essentially unique: it is unique modulo the action of this symmetry group. To describe the group more formally, it is convenient to express matrix multiplication schemes as tensors,

$$\sum_{\iota=1}^{23} \begin{pmatrix} \alpha_{1,1}^{(\iota)} & \alpha_{1,2}^{(\iota)} & \alpha_{1,3}^{(\iota)} \\ \alpha_{2,1}^{(\iota)} & \alpha_{2,2}^{(\iota)} & \alpha_{2,3}^{(\iota)} \\ \alpha_{3,1}^{(\iota)} & \alpha_{3,2}^{(\iota)} & \alpha_{3,3}^{(\iota)} \end{pmatrix} \otimes \begin{pmatrix} \beta_{1,1}^{(\iota)} & \beta_{1,2}^{(\iota)} & \beta_{1,3}^{(\iota)} \\ \beta_{2,1}^{(\iota)} & \beta_{2,2}^{(\iota)} & \beta_{2,3}^{(\iota)} \\ \beta_{3,1}^{(\iota)} & \beta_{3,2}^{(\iota)} & \beta_{3,3}^{(\iota)} \end{pmatrix} \otimes \begin{pmatrix} \gamma_{1,1}^{(\iota)} & \gamma_{1,2}^{(\iota)} & \gamma_{1,3}^{(\iota)} \\ \gamma_{2,1}^{(\iota)} & \gamma_{2,2}^{(\iota)} & \gamma_{2,3}^{(\iota)} \\ \gamma_{3,1}^{(\iota)} & \gamma_{3,2}^{(\iota)} & \gamma_{3,3}^{(\iota)} \end{pmatrix}.$$

A scheme is correct if and only if it is equal, as element of $(K^{3\times 3})^{\otimes 3}$, to $\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} E_{i,k} \otimes E_{k,j} \otimes E_{j,i}$, where $E_{u,v} \in K^{3\times 3}$ refers to the matrix which has a 1 at position $(u, v)$ and zeros everywhere else.

A permutation $\pi \in S_3$ acts on a tensor $A \otimes B \otimes C$ by permuting the three factors, and transposing each of them if $\operatorname{sgn}(\pi) = -1$. For example, $(1\ 2) \cdot (A \otimes B \otimes C) = B^\top \otimes A^\top \otimes C^\top$ and $(1\ 2\ 3) \cdot (A \otimes B \otimes C) = B \otimes C \otimes A$. A triple $(U, V, W) \in \mathrm{GL}(K, 3)^3$ of invertible matrices acts via

$$(U, V, W) \cdot (A \otimes B \otimes C) = UAV^{-1} \otimes VBW^{-1} \otimes WCU^{-1}.$$

A tuple $(U, V, W, \pi) \in \mathrm{GL}(K, 3)^3 \times S_3$ acts on a tensor $A\otimes B\otimes C$ by first letting the permutation act as described above, and then applying the matrices as described above. The set $G = \mathrm{GL}(K, 3)^3 \times S_3$ is turned into a group by defining the multiplication in such a way that the operation described above becomes a group action. The action of the group $G$ defined on tensors $A\otimes B\otimes C$ is extended to the whole space $(K^{3\times 3})^{\otimes 3}$ by linearity. In other words, elements of $G$ act on sums of tensors by acting independently on all summands.

Two matrix multiplication schemes are called equivalent if they belong to the same orbit under the action of $G$. Whenever a new matrix multiplication scheme is discovered, the question is whether it is equivalent to a known scheme, for if it is, it should not be considered as new. A common test for checking that two schemes are not equivalent proceeds by computing certain invariants of the group action. For example, since permutation and multiplication by invertible matrices do not change the rank of a matrix, we can count how many matrices of rank 1, 2, and 3 appear in the scheme. If the counts differ for two schemes, then these schemes cannot be equivalent. For example, Courtois et al. (2011) and Oh et al. (2013) proved in this way that their schemes were indeed new. If $u_i$ is the number of matrices of rank $i$ ($i = 1, 2, 3$) appearing in a certain scheme, then any two equivalent schemes will have the same tuple $(u_1, u_2, u_3)$, so this tuple can serve as an invariant. It is only a weak invariant though. A stronger one, which also takes into account some information about where the matrices of the various ranks are located in a scheme $\sum_{\iota=1}^{23} A_\iota \otimes B_\iota \otimes C_\iota$, can be expressed as the polynomial

$$f(x, y, z) := \sum_{\iota=1}^{23} \sum_{\pi\in S_3} \pi\big(x^{\operatorname{rank}(A_\iota)} y^{\operatorname{rank}(B_\iota)} z^{\operatorname{rank}(C_\iota)}\big),$$

where the permutations $\pi$ are meant to permute the variables $x, y, z$. Because of $u_1x+u_2x^2+u_3x^3 = \frac{1}{6}(f(x, 1, 1)+f(1, x, 1)+f(1, 1, x))$, the invariant $f(x, y, z)$ is a refinement of the invariant obtained by simply counting how many matrices of which rank there are. But also $f(x, y, z)$ does not capture all the information contained in the ranks. For example, the polynomial

$$g(w) = w^{\sum_{\iota=1}^{23} \operatorname{rank}(A_\iota)} + w^{\sum_{\iota=1}^{23} \operatorname{rank}(B_\iota)} + w^{\sum_{\iota=1}^{23} \operatorname{rank}(C_\iota)}$$

is an invariant which can distinguish some orbits that $f(x, y, z)$ cannot distinguish. And even when both $f(x, y, z)$ and $g(w)$ agree for two schemes, they may still be inequivalent. In fact, two schemes $\sum_{\iota=1}^{23} A_\iota \otimes B_\iota \otimes C_\iota$, $\sum_{\iota=1}^{23} A'_\iota \otimes B'_\iota \otimes C'_\iota$ may be inequivalent even if $\operatorname{rank}(A_\iota) = \operatorname{rank}(A'_\iota)$, $\operatorname{rank}(B_\iota) = \operatorname{rank}(B'_\iota)$, $\operatorname{rank}(C_\iota) = \operatorname{rank}(C'_\iota)$ for all $\iota$, so a pure consideration of ranks, while useful as an efficient and powerful preprocessing step, is in general not sufficient for deciding equivalence. In fact, many solutions found by the SAT solver were inequivalent despite sharing the same rank pattern.

Fortunately, it is not too hard to decide the equivalence of two given schemes by constructing, whenever possible, a group element that maps one to the other. A straightforward way to do so is as follows. Suppose we are given two multiplication schemes $S, S'$ and we want to decide whether there exists a tuple $(U, V, W, \pi) \in \mathrm{GL}(K, 3)^3 \times S_3$ such that $(U, V, W, \pi) \cdot S = S'$. As far as the permutation is concerned, there are only six candidates, so we can simply try each of them. Writing $S = \sum_{\iota=1}^{23} (A_\iota \otimes B_\iota \otimes C_\iota)$ and $S' = \sum_{\iota=1}^{23} (A'_\iota \otimes B'_\iota \otimes C'_\iota)$, it remains to find $U, V, W$ that map all the summands of $S$ to the summands of $S'$, albeit possibly in a different order. We search for a suitable order by the following recursive algorithm, which is initially called with $Q$ being the full space $K^{3\times3} \times K^{3\times3} \times K^{3\times3}$.

**Algorithm 1.** (Equivalence)
*Input:* two multiplication schemes $S = \sum_{\iota=1}^{23} (A_\iota \otimes B_\iota \otimes C_\iota)$, $S' = \sum_{\iota=1}^{23} (A'_\iota \otimes B'_\iota \otimes C'_\iota)$ as above, a basis of a subspace $Q$ of $K^{3\times3} \times K^{3\times3} \times K^{3\times3}$
*Output:* A triple $(U, V, W) \in \mathrm{GL}(K, 3)^3 \cap Q$ with $(U, V, W) \cdot S = S'$, or $\perp$ if no such triple exists.

1 if $S$ and $S'$ are empty, then:
2 return any element $(U, V, W)$ of $Q$ with $\det(U) \det(V) \det(W) \neq 0$, or $\perp$ if no such element exists.
3 for all summands $A'_\iota \otimes B'_\iota \otimes C'_\iota$ of $S'$, do:
4 if $\operatorname{rank}(A_1) = \operatorname{rank}(A'_\iota)$ and $\operatorname{rank}(B_1) = \operatorname{rank}(B'_\iota)$ and $\operatorname{rank}(C_1) = \operatorname{rank}(C'_\iota)$, then:
5 compute a basis of the space $P$ of all $(U, V, W)$ such that $UA_1 = A'_\iota V$, $VB_1 = B'_\iota W$, $WC_1 = C'_\iota U$ by making an ansatz, comparing coefficients, and solving a homogeneous linear system.
6 compute a basis of $R = P \cap Q$.
7 if $R$ contains at least one triple $(U, V, W)$ with $\det(U) \det(V) \det(W) \neq 0$, then:
8 call the algorithm recursively with the first summand of $S$ and the $\iota$th summand of $S'$ removed, and with $R$ in place of $Q$.
9 if the recursive call yields a triple $(U, V, W)$, return it.
10 return $\perp$.

The algorithm terminates because each recursive call is applied to a sum with strictly fewer summands. The correctness of the algorithm is clear because it essentially performs an exhaustive search through all options. In order to perform the check in step 7, we can consider a generic linear combination of the basis elements of $R$, with variables as coefficients. Then $\det(U) \det(V) \det(W)$ is a polynomial in these variables, and the question is whether this polynomial vanishes identically on $K$. For the case $K = \mathbb{Z}_2$, we can answer this by an exhaustive search.

The recursive structure of the algorithm with up to 23 recursive calls at every level may seem prohibitively expensive. However, the two filters in lines 4 and 7 turn out to cut down the number of recursive calls considerably. A straightforward implementation in Mathematica needs

no more than about one second of computation time to decide whether or not two given schemes are equivalent. Of course, we first compare the invariants, which is almost for free and suffices to settle many cases.

As pointed out by one of the referees, a similar but more sophisticated algorithm for checking the equivalence of any two given schemes was recently proposed by Berger et al. (2019).

For each scheme found by the SAT solver we have checked whether it is equivalent (for $K = \mathbb{Z}_2$) to one of the schemes found earlier, or to one of the four known schemes found by Laderman, Smirnov, Oh et al., and Courtois et al., respectively. From the roughly 300,000 solutions found by the SAT solver that were distinct modulo the order of the summands, we isolated about 17,000 schemes that were distinct modulo equivalence over $K = \mathbb{Z}_2$. These are the schemes announced in the introduction. In the appendix, we list the number of schemes we found separated by invariant.

## 5. Simplifying Solutions

We can use the symmetries introduced in the previous section not only to recognize that a seemingly new scheme is not really new. We can also use them for simplifying schemes. A scheme can for example be regarded as simpler than another scheme if the number of terms $\alpha^{(t)}_{i_1,i_2}\beta^{(t)}_{j_1,j_2}\gamma^{(t)}_{k_1,k_2}$ in it which evaluate to 1 is smaller. We call this number the *weight* of a scheme. For example, a summand

$$\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\0&1&1\\0&1&1\end{pmatrix}$$

would contribute $1 \cdot 3 \cdot 5 = 15$ to the weight of a scheme. Note that the weight of a scheme defined in this way is not directly related to the computational cost associated with it (e.g., to the number of additions needed). Instead, the definition motivated by the observation that schemes with low weight seem to be more fertile when used as seed in the search for further schemes.

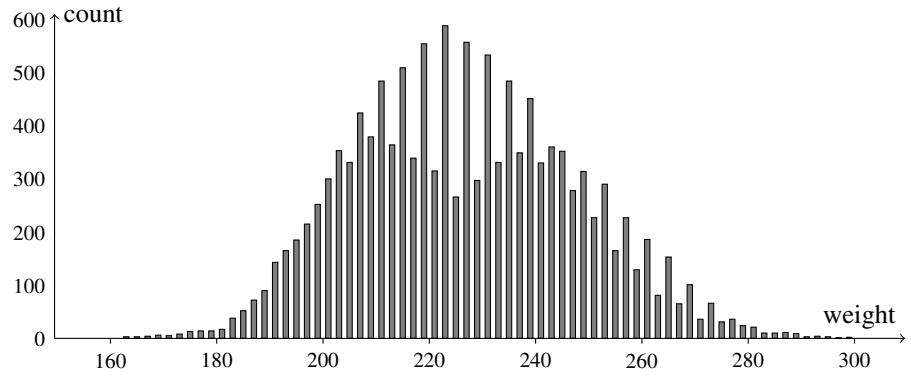
Ideally, we would therefore like to replace every scheme $S$ by an equivalent scheme with smallest possible weight. In principle, we could find such a minimal equivalent element by applying all elements of $G$ to $S$ and taking the smallest result. Unfortunately, even for $K = \mathbb{Z}_2$, the group $G$ has $168^3 \cdot 6 = 28\,449\,792$ elements, so trying them all might be feasible if we had to do it for a few schemes, but not for thousands of them. If we do not insist on the smallest possible weight, we can take a pragmatic approach and just spend for every scheme $S$ a prescribed amount of computation time (say half an hour) applying random elements of $G$ to $S$:

**Algorithm 2.** (Weight reduction)
*Input:* a multiplication scheme $S$
*Output:* an equivalent multiplication scheme whose weight is less than or equal to the weight of $S$.

```
1  while the time limit is not exhausted, do
2     pick a group element g at random
3     if weight(g(S)) < weight(S), then set S = g(S)
4  return S
```

With this algorithm, we were able to replace about 20% of the new schemes found by the SAT solver by equivalent schemes with smaller weight. It is not too surprising that no improvement was found for the majority of cases, because the way we specified the problem to the SAT solver already induces a bias towards solutions with a small weight.

The figure below shows the distribution of our 17,000 schemes according to weight, after simplification. It is clear that the weight is always odd, hence the small gaps between the bars. It is less clear why we seem to have an overlay of three normal distributions, but we believe that this is rather an artifact of the way we generated the solutions than a structural feature of the entire solution set.




## 6. Generalizing the Coefficient Ring

At this point, we have a considerable number of new matrix multiplication schemes for the coefficient field $K = \mathbb{Z}_2$. The next step is to lift them to schemes that work in any coefficient ring. The SAT solver presents us with a solution for $\mathbb{Z}_2$ in which all coefficients are 0 or 1, and in order to lift such a solution, we make the hypothesis that this solution originated from a solution for an arbitrary coefficient ring in which all coefficients are $+1$, $-1$, or 0. The distinction between $+1$ and $-1$ gets lost in $\mathbb{Z}_2$, and the task consists in recovering it. There is a priori no reason why such a lifting should exist, and indeed, we have seen a small number of instances where it fails. One such example is given in the appendix. Interestingly however, these examples seem to be very rare. In almost all cases, a lifting turned out to exist. Note also that any two schemes that are inequivalent over $\mathbb{Z}_2$ will lift to schemes that are necessarily inequivalent over $\mathbb{Z}$, albeit not necessarily over $\mathbb{Q}$.

In order to explain the lifting process, we return to the Brent equations discussed in Section 2. We set variables corresponding to coefficients that are zero in the SAT solution to zero, which simplifies the system considerably. According to the axioms of tensor products, we have $(\lambda A) \otimes B \otimes C = A \otimes (\lambda B) \otimes C = A \otimes B \otimes (\lambda C)$ for any $A, B, C$ and every constant $\lambda$. We may therefore select in every summand $A \otimes B \otimes C$ one variable appearing in $A$ and one variable appearing in $B$ and set them to $+1$. This reduces the number of variables further. However, the resulting system is still to hard to be solved directly.

Before calling a general purpose Gröbner basis engine, we apply some simplifications to the system, which take into account that we are only interested in solutions whose coordinates are $-1$ or $+1$. In particular, we can replace any exponent $k$ appearing in any of the polynomials by $k$ mod 2, we can cancel factors that clearly do not vanish on the points of interest, and we

can replace polynomials of the from $xy \pm 1$ by $x \pm y$. These simplifications may bring up some linear polynomials. By triangularizing the linear system corresponding to these polynomials, we can eliminate some of the variables. We can then simplify again, and possibly obtain new linear equations. The process is repeated until no further linear equations appear. We then add for each variable $x$ the polynomial $x^2 - 1$ and compute a Gröbner basis with respect to a degree order. If this leads to new linear polynomials, we return to iterating triangularization, elimination, and simplification until no further linear equations show up, and then compute again a degree Gröbner basis. The whole process is repeated until we obtain a Gröbner basis that does not contain any new linear equations. If there are more than 15 variables left, we next compute a minimal associated prime ideal of an elimination ideal involving only five variables, and check whether adding it to the original system and computing a Gröbner basis leads to new linear equations. If it does, we start over with the whole procedure. Otherwise, we compute the minimal associated prime ideal of the whole system and return the solution corresponding to one of the prime factors. The process is summarized in the following listing.

**Algorithm 3.** (System Solving for Lifting)
*Input:* A finite subset $B$ of $\mathbb{Q}[x_1, \ldots, x_n]$
*Output:* A common root $\xi \in \{-1, 1\}^n$ of all the elements of $B$, or $\perp$ if no such common root exists.

Replace every exponent $k$ appearing in an element of $B$ by $k$ mod 2
For every $p \in B$ and every $i$ with $x_i \mid p$, replace $p$ by $p/x_i$
Replace every element of the form $xy - 1$ or $-xy - 1$ by $x - y$ or $x + y$, respectively.
if $B$ now contains linear polynomials, then:
    Use them to eliminate some variables, say $y_1, \ldots, y_k$
    Call the procedure recursively on the resulting set of polynomials
    if there is a solution, extend it to the eliminated variables $y_1, \ldots, y_k$ and return the result
    if there is no solution, return $\perp$.
Compute a Gröbner basis $G$ of $B \cup \{x_i^2 - 1 : i = 1, \ldots, n\}$ with respect to a degree order
if $G = \{1\}$, return $\perp$
if $G$ contains linear polynomials, then call this procedure recursively and return the result
if $n > 15$, then:
    Compute a basis $P$ of one of the minimal associated prime ideals of $\langle G \rangle \cap \mathbb{Q}[x_1, \ldots, x_5]$.
    Compute a Gröbner basis $G'$ of $G \cup P$ with respect to a degree order
    if $G'$ contains linear polynomials, then call this procedure recursively and return the result
Compute a basis $P$ of one of the minimal associated prime ideals of $\langle G \rangle \subseteq \mathbb{Q}[x_1, \ldots, x_n]$.
Return the common solution $\xi$ of $P$.

An implementation of this procedure in Mathematica is available on the website of this article (Heule et al., 2019b). In this implementation, we use Singular (Greuel and Pfister, 2002) for doing the Gröbner basis calculations and for the computation of minimal associated prime ideals. Despite the large number of variables, Singular handles the required computations with impressive speed. The task of signing all the solutions was distributed on several Linux servers with up to 128 cores and up to 386GB of memory and took only about 20 seconds sequential computing time per solution on the average. Only a small number of cases, which happen to have a few more variables than the others, need much longer, up to a few hours.

## 7. Introducing Parameters

The idea of instantiating some of the variables based on a known scheme and then solving for the remaining variables approach not only applies to SAT solving. It also has an algebraic counterpart. Solving the Brent equations with algebraic methods is infeasible because the equations are nonlinear, but observe that we only have to solve a linear system if we start from a known scheme and only replace all $\gamma^{(\iota)}_{k_1,k_2}$ by fresh variables. Solving linear systems is of course much easier than solving nonlinear ones.

More generally, we can select for each $\iota \in \{1, \ldots, 23\}$ separately whether we want to replace all $\alpha^{(\iota)}_{i_1,i_2}$'s or all $\beta^{(\iota)}_{j_1,j_2}$'s or all $\gamma^{(\iota)}_{k_1,k_2}$'s by fresh variables, and we still just get a linear system for these variables. Once we make a selection, solving the resulting linear system yields an affine vector space. One might expect this affine space will typically consist of a single point only, but this is usually not the case.

A solution space with positive dimension can be translated into a multiplication scheme involving one or more free parameters. Starting from the resulting parameterized scheme, we can play the same game with another selection of variables, which may allow us to introduce further parameters. If we repeat the procedure several times with random selections of which variables are known, we obtain huge schemes involving 40 or more parameters. These parameters are however algebraically dependent, or at least it is too costly check whether they are dependent or not. We got better results by proceeding more systematically, as summarized int in the following listing.

**Algorithm 4.** (Introducing Parameters)
*Input:* A matrix multiplication scheme $S = \sum_{\iota=1}^{23}(A_\iota \otimes B_\iota \otimes C_\iota)$. Write $A_\iota = ((\alpha^{(\iota)}_{i,j}))$, $B_\iota = ((\beta^{(\iota)}_{i,j}))$, $C_\iota = ((\gamma^{(\iota)}_{i,j}))$.
*Output:* A family of matrix multiplication schemes with parameters $x_1, x_2, \ldots$

1 for $\iota = 1, \ldots, 23$, do:
2  for every choice $u, v \in \{\alpha, \beta, \gamma\}$ with $u \neq v$, do:
3  replace all entries $u^{(\iota)}_{i,j}$ for $i, j = 1, \ldots, 3$ in $S$ by fresh variables
4  replace all entries $v^{(m)}_{i,j}$ for $i, j = 1, \ldots, 3$ and $m \neq \iota$ in $S$ by fresh variables
5  equate the resulting scheme $S$ to $\sum_{i,j,k} E_{i,j} \otimes E_{j,k} \otimes E_{k,i}$ and compare coefficients
6  solve the resulting inhomogeneous linear system for the fresh variables introduced in steps 3 and 4
7  substitute the generic solution, using new parameters $x_i, x_{i+1}, \ldots$, into $S$
8 return $S$

With this algorithm and some slightly modified variants (e.g., letting the outer loop run backwards or transposing the inner and the outer loop), we were able to obtain schemes with altogether up to 17 parameters. Although all new parameters introduced in a certain iteration can only appear linearly in the scheme, old parameters that were considered as belonging to the ground ring during the linear solving can later appear rationally. However, by manually applying suitable changes of variables, we managed to remove all denominators from all the families we inspected. We do not even need integer denominators. Using Gröbner bases, we can further check whether the parameters are independent, and for several families with 17 parameters they turn out to be. In the language of algebraic geometry, this means that the solution set of the Brent equations has at least dimension 17 as an algebraic variety.

One of our families is shown in the appendix, and some further ones are provided electronically on our website. These families should be contrasted with the family found by Johnson and McLoughlin (1986). In particular, while they lament that their family contains fractional coefficients such as $\frac{1}{2}$ and $\frac{1}{3}$ and therefore does not apply in every coefficient ring, our families only involve integer coefficients and therefore have no such restriction. Moreover, their family has only three parameters, and with the method described above, only 6 additional parameters can be introduced into it. The number of parameters we managed to introduce into the known solutions by Laderman, Courtois et al., Oh et al., and Smirnov are 0, 6, 10, and 14, respectively.

## 8. Concluding Remarks

Although we have found many new multiplication schemes with 23 multiplications, we did not encounter a single scheme with 22 multiplications. We have checked all schemes whether some of their summands can be merged together using tensor product arithmetic. For doing so, it would suffice if a certain scheme contains some summands which share the same $A$'s, say, and where the corresponding $B$'s, say, of these rows are linearly dependent. We could then express one of these $B$'s in terms of the others and eliminate the summand in which it appears. For example, if $B_3 = \beta_1 B_1 + \beta_2 B_2$, then we have $A \otimes B_1 \otimes C_1 + A \otimes B_2 \otimes C_2 + A \otimes B_3 \otimes C_3 = A \otimes B_1 \otimes (C_1 + \beta_1 C_3) + A \otimes B_2 \otimes (C_2 + \beta_2 C_3)$. Since none of our schemes admits a simplification of this kind, it remains open whether a scheme with 22 multiplications exists.

Another open question is: how many further schemes with 23 multiplications and coefficients in $\{-1, 0, 1\}$ are there? We have no evidence that we have found them all. In fact, we rather believe that there are many further ones, possibly including schemes that are very different from ours. There may also be parametrized families with more than 17 parameters, and it would be interesting to know the maximal possible number of parameters, i.e., the actual dimension of the solution set of the Brent equations.

## Acknowledgments

We acknowledge the Texas Advanced Computing Center at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. We also thank the referees for their valuable remarks and suggestions.

## References

Berger, G.O., Absil, P.A., Lathauwer, L.D., Jungers, R.M., Barel, M.V., 2019. Equivalent Polyadic Decompositions of Matrix Multiplication Tensors. Technical Report 1902.03950. arxiv.

Biere, A., 2018. CaDiCaL, Lingeling, Plingeling, Treengeling and YalSAT Entering the SAT Competition 2018, in: Proc. of SAT Competition 2018 – Solver and Benchmark Descriptions, University of Helsinki. pp. 13–14.

Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (Eds.), 2009. Handbook of Satisfiability. volume 185 of *Frontiers in Artificial Intelligence and Applications*, IOS Press.

Bläser, M., 2003. On the complexity of the multiplication of matrices of small formats. Journal of Complexity 19, 43–60.

Bläser, M., 2013. Fast Matrix Multiplication. Number 5 in Graduate Surveys, Theory of Computing Library. URL: `http://www.theoryofcomputing.org/library.html`, doi:`10.4086/toc.gs.2013.005`.

Brent, R.P., 1970. Algorithms for matrix multiplication. Technical Report. Department of Computer Science, Stanford.

Buchberger, B., 1965. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal. Ph.D. thesis. Universität Innsbruck.

Buchberger, B., Kauers, M., 2010. Gröbner basis. Scholarpedia 5, 7763. `http://www.scholarpedia.org/article/Groebner_basis`.

Bürgisser, P., Clausen, M., Shokrollahi, M.A., 2013. Algebraic complexity theory. volume 315. Springer Science & Business Media.

Courtois, N., Bard, G.V., Hulme, D., 2011. A new general-purpose method to multiply $3 \times 3$ matrices using only 23 multiplications. CoRR abs/1108.2830. URL: http://arxiv.org/abs/1108.2830, arXiv:1108.2830.

Cox, D., Little, J., OShea, D., 1992. Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics, Springer.

Drevet, C., Islam, M.N., Schost, É., 2011. Optimization techniques for small matrix multiplication. Theor. Comput. Sci. 412, 2219–2236.

Dumas, J.G., Giorgi, P., Pernet, C., 2008. Dense linear algebra over word-size prime fields: the fflas and ffpack packages. ACM Trans. on Mathematical Software (TOMS) 35, 1–42. doi:10.1145/1391989.1391992.

Gomes, C., Sellmann, M., 2004. Streamlined constraint reasoning, in: Principles and Practice of Constraint Programming (CP 2004), Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 274–289.

Greuel, G.M., Pfister, G., 2002. A Singular Introduction to Commutative Algebra. Springer.

de Groote, H.F., 1978. On varieties of optimal algorithms for the computation of bilinear mappings i. the isotropy group of a bilinear mapping. Theoretical Computer Science 7, 1–24.

Heule, M.J.H., 2018. Schur number five, in: McIlraith, S.A., Weinberger, K.Q. (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), AAAI Press. pp. 6598–6606. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16952.

Heule, M.J.H., Kauers, M., Seidl, M., 2019a. Local Search for Fast Matrix Multiplication, in: Proceedings of SAT'19, pp. 155–163. Also ArXiv 1903.11391.

Heule, M.J.H., Kauers, M., Seidl, M., 2019b. Matrix multiplication repository. http://www.algebra.uni-linz.ac.at/research/matrix-multiplication/.

Heule, M.J.H., Kullmann, O., Marek, V.W., 2016. Solving and verifying the boolean Pythagorean triples problem via cube-and-conquer, in: Creignou, N., Berre, D.L. (Eds.), Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016), Springer. pp. 228–245. URL: https://doi.org/10.1007/978-3-319-40970-2_15, doi:10.1007/978-3-319-40970-2_15.

Huang, J., Smith, T.M., Henry, G.M., van de Geijn, R.A., 2016. Strassen's algorithm reloaded, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press, Piscataway, NJ, USA. pp. 59:1–59:12. URL: http://dl.acm.org/citation.cfm?id=3014904.3014983.

Johnson, R.W., McLoughlin, A.M., 1986. Noncommutative bilinear algorithms for $3 \times 3$ matrix multiplication. SIAM J. Comput. 15, 595–603. URL: https://doi.org/10.1137/0215043, doi:10.1137/0215043.

Kerber, A., 1991. Algebraic Combinatorics via Finite Group Actions. BI-Wissenschaftsverlag.

Laderman, J.D., 1976. A noncommutative algorithm for multiplying $3 \times 3$ matrices using 23 multiplications. Bulletin of the American Mathematical Society 82, 126–128.

Landsberg, J.M., 2017. Geometry and complexity theory. volume 169. Cambridge University Press.

Le Gall, F., 2014. Powers of tensors and fast matrix multiplication, in: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM. pp. 296–303. URL: http://doi.acm.org/10.1145/2608628.2608664, doi:10.1145/2608628.2608664.

Makarov, O., 1987. A non-commutative algorithm for multiplying $5 \times 5$ matrices using one hundred multiplications. USSR Computational Mathematics and Mathematical Physics 27, 205 – 207. URL: http://www.sciencedirect.com/science/article/pii/0041555387901455, doi:https://doi.org/10.1016/0041-5553(87)90145-5.

Oh, J., Kim, J., Moon, B.R., 2013. On the inequivalence of bilinear algorithms for 3×3 matrix multiplication. Information Processing Letters 113, 640–645.

Pan, V.Y., 2018. Fast feasible and unfeasible matrix multiplication. CoRR abs/1804.04102. URL: http://arxiv.org/abs/1804.04102, arXiv:1804.04102.

Sedoglavic, A., 2017a. Laderman matrix multiplication algorithm can be constructed using strassen algorithm and related tensor's isotropies. CoRR abs/1703.08298. URL: http://arxiv.org/abs/1703.08298, arXiv:1703.08298.

Sedoglavic, A., 2017b. A non-commutative algorithm for multiplying $5 \times 5$ matrices using 99 multiplications. CoRR abs/1707.06860. URL: http://arxiv.org/abs/1707.06860, arXiv:1707.06860.

Sedoglavic, A., 2017c. A non-commutative algorithm for multiplying $7 \times 7$ matrices using 250 multiplications. CoRR abs/1712.07935. URL: http://arxiv.org/abs/1712.07935, arXiv:1712.07935.

Sedoglavic, A., 2019. Yet another catalogue of fast matrix multiplication algorithms. https://fmm.univ-lille.fr/. Accessed: 2019-03-17.

Smirnov, A.V., 2013. The bilinear complexity and practical algorithms for matrix multiplication. Computational Mathematics and Mathematical Physics 53, 1781–1795.

Smirnov, A.V., 2017. Several bilinear algorithms for matrix multiplication. Technical Report. Technical report.

Strassen, V., 1969. Gaussian elimination is not optimal. Numerische Mathematik 13, 354–356.

Vassilevska Williams, V., 2012. Multiplying matrices faster than Coppersmith-Winograd, in: Proceedings of the 44th

Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA. pp. 887–898. URL: http://doi.acm.org/10.1145/2213977.2214056, doi:10.1145/2213977.2214056.
Winograd, S., 1971. On multiplication of $2 \times 2$ matrices. Linear algebra and its applications 4, 381–388.

## Appendix

**Table 1.** Number of non-equivalent schemes by invariant $f(x, y, z)$ as defined in Section 4. This table is based on the schemes for $\mathbb{Z}_2$ and include the few schemes that could not be lifted to $\mathbb{Z}$. The fact that the rank of a matrix appearing in a scheme can only be 1, 2, or 3 restricts the number of different polynomials that can arise as value of the invariant $f(x, y, z)$ to 28048800. We have encountered 106 of them. The following table shows those for which we found more than 10 orbits, together with the respective counts. The remaining ones can be found on our website (Heule et al., 2019b).

| | |
|---|---|
| 12 | $x^3y^2z + x^2y^3z + x^3yz^2 + 24x^2y^2z^2 + xy^3z^2 + x^2yz^3 + xy^2z^3 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 12x^2yz + 12xy^2z + 12xyz^2 + 60xyz$ |
| 14 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 22x^2yz + 22xy^2z + 22xyz^2 + 36xyz$ |
| 14 | $24x^2y^2z^2 + 4x^3yz + 2x^2y^2z + 4xy^3z + 2x^2yz^2 + 2xy^2z^2 + 4xyz^3 + 8x^2yz + 8xy^2z + 8xyz^2 + 72xyz$ |
| 15 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 22x^2yz + 22xy^2z + 22xyz^2 + 42xyz$ |
| 15 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 28x^2yz + 28xy^2z + 28xyz^2 + 24xyz$ |
| 15 | $24x^2y^2z^2 + 4x^3yz + 2x^2y^2z + 4xy^3z + 2x^2yz^2 + 2xy^2z^2 + 4xyz^3 + 14x^2yz + 14xy^2z + 14xyz^2 + 54xyz$ |
| 16 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 8x^2yz + 8xy^2z + 8xyz^2 + 84xyz$ |
| 16 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 4x^2yz + 4xy^2z + 4xyz^2 + 78xyz$ |
| 19 | $24x^2y^2z^2 + 4x^3yz + 2x^2y^2z + 4xy^3z + 2x^2yz^2 + 2xy^2z^2 + 4xyz^3 + 10x^2yz + 10xy^2z + 10xyz^2 + 66xyz$ |
| 21 | $24x^2y^2z^2 + 4x^2y^2z + 4x^2yz^2 + 4xy^2z^2 + 14x^2yz + 14xy^2z + 14xyz^2 + 60xyz$ |
| 21 | $24x^2y^2z^2 + 4x^3yz + 4xy^3z + 4xyz^3 + 10x^2yz + 10xy^2z + 10xyz^2 + 72xyz$ |
| 27 | $24x^2y^2z^2 + 4x^3yz + 4xy^3z + 4xyz^3 + 16x^2yz + 16xy^2z + 16xyz^2 + 54xyz$ |
| 31 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 16x^2yz + 16xy^2z + 16xyz^2 + 42xyz$ |
| 36 | $24x^2y^2z^2 + 4x^3yz + 4xy^3z + 4xyz^3 + 12x^2yz + 12xy^2z + 12xyz^2 + 66xyz$ |
| 40 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 6x^2yz + 6xy^2z + 6xyz^2 + 84xyz$ |
| 42 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 20x^2yz + 20xy^2z + 20xyz^2 + 42xyz$ |
| 45 | $24x^2y^2z^2 + 4x^2y^2z + 4x^2yz^2 + 4xy^2z^2 + 10x^2yz + 10xy^2z + 10xyz^2 + 72xyz$ |
| 53 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 6x^2yz + 6xy^2z + 6xyz^2 + 72xyz$ |
| 54 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 24x^2yz + 24xy^2z + 24xyz^2 + 36xyz$ |
| 59 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 10x^2yz + 10xy^2z + 10xyz^2 + 78xyz$ |
| 63 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 18x^2yz + 18xy^2z + 18xyz^2 + 54xyz$ |
| 68 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 14x^2yz + 14xy^2z + 14xyz^2 + 48xyz$ |
| 77 | $24x^2y^2z^2 + 20x^2yz + 20xy^2z + 20xyz^2 + 54xyz$ |
| 79 | $24x^2y^2z^2 + 4x^2y^2z + 4x^2yz^2 + 4xy^2z^2 + 12x^2yz + 12xy^2z + 12xyz^2 + 66xyz$ |
| 81 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 18x^2yz + 18xy^2z + 18xyz^2 + 48xyz$ |
| 83 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 20x^2yz + 20xy^2z + 20xyz^2 + 48xyz$ |
| 91 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 16x^2yz + 16xy^2z + 16xyz^2 + 54xyz$ |
| 101 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 8x^2yz + 8xy^2z + 8xyz^2 + 66xyz$ |
| 102 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 12x^2yz + 12xy^2z + 12xyz^2 + 54xyz$ |
| 105 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 8x^2yz + 8xy^2z + 8xyz^2 + 78xyz$ |
| 106 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 14x^2yz + 14xy^2z + 14xyz^2 + 66xyz$ |
| 114 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 16x^2yz + 16xy^2z + 16xyz^2 + 60xyz$ |
| 144 | $24x^2y^2z^2 + 6x^3yz + 2x^2y^2z + 6xy^3z + 2x^2yz^2 + 2xy^2z^2 + 6xyz^3 + 10x^2yz + 10xy^2z + 10xyz^2 + 60xyz$ |
| 148 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 10x^2yz + 10xy^2z + 10xyz^2 + 72xyz$ |
| 150 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 14x^2yz + 14xy^2z + 14xyz^2 + 60xyz$ |
| 176 | $24x^2y^2z^2 + 2x^3yz + 2xy^3z + 2xyz^3 + 12x^2yz + 12xy^2z + 12xyz^2 + 72xyz$ |
| 186 | $24x^2y^2z^2 + 2x^3yz + 2x^2y^2z + 2xy^3z + 2x^2yz^2 + 2xy^2z^2 + 2xyz^3 + 12x^2yz + 12xy^2z + 12xyz^2 + 66xyz$ |
| 248 | $24x^2y^2z^2 + 6x^2yz + 6xy^2z + 6xyz^2 + 96xyz$ |
| 264 | $24x^2y^2z^2 + 2x^2y^2z + 2x^2yz^2 + 2xy^2z^2 + 16x^2yz + 16xy^2z + 16xyz^2 + 60xyz$ |
| 402 | $24x^2y^2z^2 + 8x^2yz + 8xy^2z + 8xyz^2 + 90xyz$ |
| 431 | $24x^2y^2z^2 + 2x^2y^2z + 2x^2yz^2 + 2xy^2z^2 + 8x^2yz + 8xy^2z + 8xyz^2 + 84xyz$ |
| 745 | $24x^2y^2z^2 + 18x^2yz + 18xy^2z + 18xyz^2 + 60xyz$ |
| 828 | $24x^2y^2z^2 + 2x^2y^2z + 2x^2yz^2 + 2xy^2z^2 + 14x^2yz + 14xy^2z + 14xyz^2 + 66xyz$ |
| 902 | $24x^2y^2z^2 + 2x^2y^2z + 2x^2yz^2 + 2xy^2z^2 + 10x^2yz + 10xy^2z + 10xyz^2 + 78xyz$ |

| | |
|---|---|
| 1027 | $24x^2y^2z^2 + 2x^2y^2z + 2x^2yz^2 + 2xy^2z^2 + 12x^2yz + 12xy^2z + 12xyz^2 + 72xyz$ |
| 1431 | $24x^2y^2z^2 + 10x^2yz + 10xy^2z + 10xyz^2 + 84xyz$ |
| 2374 | $24x^2y^2z^2 + 16x^2yz + 16xy^2z + 16xyz^2 + 66xyz$ |
| 2795 | $24x^2y^2z^2 + 12x^2yz + 12xy^2z + 12xyz^2 + 78xyz$ |
| 3304 | $24x^2y^2z^2 + 14x^2yz + 14xy^2z + 14xyz^2 + 72xyz$ |

**Table 2.** Number of non-equivalent schemes by invariant $g(w)$ as defined in Section 4. This table is based on the schemes for $\mathbb{Z}_2$ and include the few schemes that could not be lifted to $\mathbb{Z}$. The fact that the rank of a matrix appearing in a scheme can only be 1, 2, or 3 restricts the number of different polynomials that can arise as value of the invariant $g(w)$ to 17343. We have encountered the following 77 of them.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | $2w^{28} + w^{27}$ | 1 | $w^{33} + w^{32} + w^{30}$ | 1 | $w^{34} + 2w^{31}$ | 3 | $w^{34} + 2w^{30}$ |
| 3 | $w^{34} + 2w^{32}$ | 5 | $2w^{33} + w^{32}$ | 5 | $w^{34} + 2w^{29}$ | 6 | $w^{33} + w^{31} + w^{28}$ |
| 6 | $w^{34} + w^{30} + w^{29}$ | 6 | $w^{34} + w^{31} + w^{29}$ | 9 | $w^{34} + w^{32} + w^{30}$ | 14 | $w^{33} + 2w^{30}$ |
| 15 | $2w^{32} + w^{28}$ | 15 | $w^{32} + 2w^{30}$ | 16 | $2w^{33} + w^{31}$ | 16 | $w^{30} + w^{28} + w^{27}$ |
| 16 | $w^{33} + w^{32} + w^{29}$ | 17 | $w^{33} + w^{31} + w^{29}$ | 18 | $2w^{31} + w^{27}$ | 19 | $w^{33} + w^{31} + w^{30}$ |
| 20 | $3w^{30}$ | 25 | $w^{31} + w^{28} + w^{27}$ | 31 | $w^{33} + w^{30} + w^{29}$ | 33 | $2w^{32} + w^{29}$ |
| 35 | $2w^{32} + w^{30}$ | 41 | $w^{33} + w^{32} + w^{31}$ | 42 | $2w^{32} + w^{27}$ | 43 | $w^{33} + w^{30} + w^{28}$ |
| 46 | $w^{33} + 2w^{32}$ | 51 | $3w^{32}$ | 63 | $3w^{28}$ | 63 | $w^{33} + 2w^{31}$ |
| 73 | $w^{32} + w^{31} + w^{27}$ | 96 | $3w^{31}$ | 96 | $w^{31} + 2w^{30}$ | 104 | $w^{33} + w^{29} + w^{27}$ |
| 106 | $2w^{31} + w^{28}$ | 112 | $w^{32} + w^{31} + w^{29}$ | 122 | $w^{32} + w^{31} + w^{28}$ | 123 | $3w^{29}$ |
| 134 | $w^{33} + 2w^{29}$ | 138 | $w^{31} + w^{29} + w^{27}$ | 145 | $2w^{30} + w^{29}$ | 146 | $2w^{32} + w^{31}$ |
| 160 | $2w^{31} + w^{30}$ | 165 | $w^{32} + w^{30} + w^{29}$ | 175 | $w^{32} + w^{31} + w^{30}$ | 185 | $w^{29} + w^{28} + w^{27}$ |
| 190 | $2w^{31} + w^{29}$ | 191 | $w^{31} + 2w^{28}$ | 192 | $w^{29} + 2w^{28}$ | 198 | $2w^{29} + w^{27}$ |
| 201 | $w^{32} + 2w^{31}$ | 214 | $w^{30} + 2w^{28}$ | 216 | $w^{30} + 2w^{29}$ | 220 | $w^{31} + w^{30} + w^{27}$ |
| 240 | $2w^{30} + w^{27}$ | 245 | $w^{30} + w^{29} + w^{27}$ | 266 | $2w^{29} + w^{28}$ | 297 | $2w^{30} + w^{28}$ |
| 312 | $w^{32} + w^{30} + w^{27}$ | 330 | $w^{32} + w^{30} + w^{28}$ | 365 | $w^{33} + w^{29} + w^{28}$ | 401 | $w^{33} + 2w^{28}$ |
| 459 | $w^{32} + 2w^{29}$ | 472 | $w^{30} + w^{29} + w^{28}$ | 496 | $w^{33} + 2w^{27}$ | 540 | $w^{31} + 2w^{29}$ |
| 548 | $w^{31} + w^{30} + w^{29}$ | 575 | $w^{31} + w^{30} + w^{28}$ | 684 | $w^{32} + 2w^{27}$ | 733 | $w^{31} + w^{29} + w^{28}$ |
| 749 | $w^{33} + w^{28} + w^{27}$ | 776 | $w^{32} + w^{29} + w^{27}$ | 1317 | $w^{32} + 2w^{28}$ | 1590 | $w^{32} + w^{28} + w^{27}$ |
| 1596 | $w^{32} + w^{29} + w^{28}$ | | | | | | |

**Table 3.** The polynomial invariants $f(x, y, z)$ and $g(w)$ proposed in the text do not contain all the information provided by the rank pattern of a scheme. The rank pattern can be viewed as a table $R \in \{1, 2, 3\}^{3\times 23}$, the $(i, \iota)$-th entry of which is the rank of $A_\iota, B_\iota, C_\iota$ if $i$ is $1, 2, 3$, respectively. The table $R$ itself is not an invariant, but if we consider two such tables $R, R'$ as equivalent if we can map one to the other by suitably permuting rows and columns, then the equivalence classes w.r.t. this equivalence relation are invariants. Using Polya theory (e.g. Kerber, 1991), it can be calculated that the set $\{1, 2, 3\}^{3\times 23}$ splits into 9724560295640 such equivalence classes. We encountered 303 of them. In the table below, we list those for which we found more than 100 orbits, together with the respective counts. The remaining ones can be found on our website (Heule et al., 2019b).

| | | | | | |
|---|---|---|---|---|---|
| 111 | $[\binom{11111111111111111122222}{11111111111111112222222}]_\sim$<br>$11111111111222221112222$ | 111 | $[\binom{11111111111111111222222}{11111111111111121222222}]_\sim$<br>$11111111111222221112222$ | 112 | $[\binom{11111111111111111222222}{11111111111111122112222}]_\sim$<br>$11111111111222211222222$ |
| 118 | $[\binom{11111111111111111222222}{11111111111111122112222}]_\sim$<br>$11111111111122111122222$ | 119 | $[\binom{11111111111111112222223}{11111111111112231122221}]_\sim$<br>$11111111112231111222221$ | 123 | $[\binom{11111111111111111222222}{11111111111111121222222}]_\sim$<br>$11111111111222211122222$ |
| 125 | $[\binom{11111111111111111122222}{11111111111111111222222}]_\sim$<br>$11111111111122222112222$ | 126 | $[\binom{11111111111111111122222}{11111111111111112212222}]_\sim$<br>$11111111112222211222222$ | 148 | $[\binom{11111111111111111122222}{11111111111111112212222}]_\sim$<br>$11111111112222221112222$ |
| 156 | $[\binom{11111111111111111222222}{11111111111112221122222}]_\sim$<br>$11111111112222211112222$ | 166 | $[\binom{11111111111111111122222}{11111111111111122212222}]_\sim$<br>$11111111112222211112222$ | 171 | $[\binom{11111111111111111222222}{11111111111111122122222}]_\sim$<br>$11111111111222211112222$ |

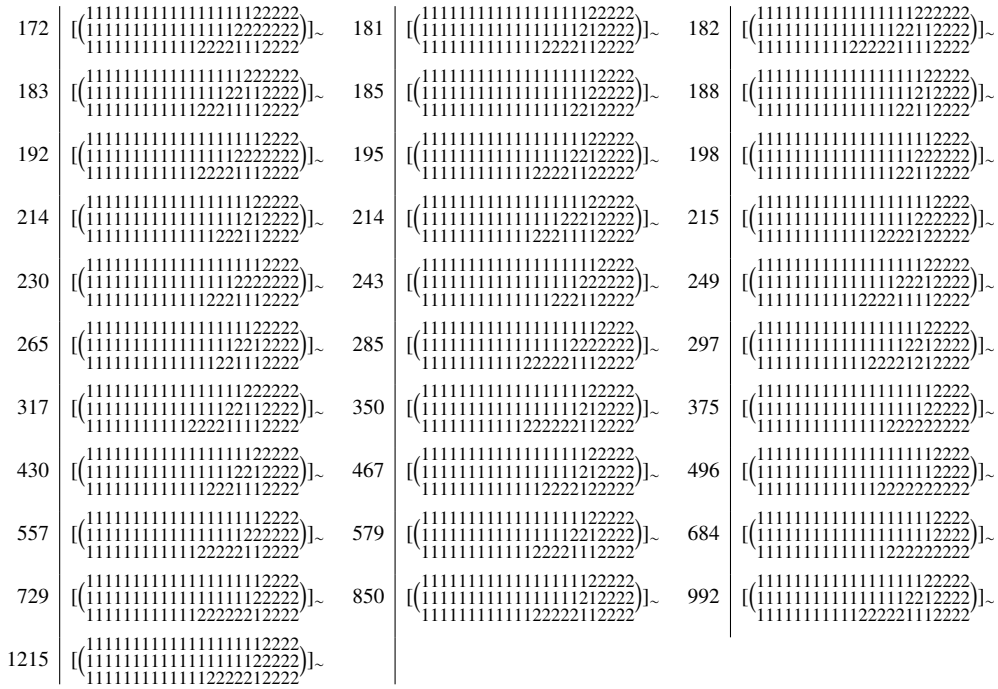


**Table 4.** A multiplication scheme for the coefficient ring $\mathbb{Z}_2$ that cannot be extended to a scheme for $\mathbb{Z}$ by replacing some of the 1's by $-1$'s. It may still be possible to find a scheme with coefficients in $\mathbb{Z}$ which reduces to this scheme modulo 2, but any such scheme must have at least one coefficient with absolute value $\geq 2$.

$$
\begin{aligned}
&\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\0&1&1\\0&1&1\end{pmatrix}+\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\1&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}1&1&1\\0&0&0\\0&0&0\end{pmatrix}\\
+&\begin{pmatrix}0&0&0\\0&0&0\\0&1&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&1\\0&0&1\\0&0&1\end{pmatrix}+\begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}1&1&1\\0&0&0\\0&0&0\end{pmatrix}\\
+&\begin{pmatrix}0&0&0\\1&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}1&0&1\\0&0&0\\0&0&0\end{pmatrix}+\begin{pmatrix}0&0&0\\1&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&1&0\\0&1&0\end{pmatrix}\\
+&\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&0&0\\0&1&1\end{pmatrix}+\begin{pmatrix}0&0&0\\0&1&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&1\\0&1&0\\0&1&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&1&0\\0&1&1\end{pmatrix}\\
+&\begin{pmatrix}0&0&0\\0&1&0\\0&1&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\1&0&1\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}+\begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\1&0&0\\0&0&1\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix}\\
+&\begin{pmatrix}0&0&0\\0&0&1\\0&0&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&0&0\\1&0&1\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}+\begin{pmatrix}0&0&0\\0&1&1\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&0&0\\1&1&0\end{pmatrix}\\
+&\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&0\\0&1&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\1&0&0\\1&0&0\end{pmatrix}+\begin{pmatrix}1&0&0\\0&0&0\\1&0&0\end{pmatrix}\otimes\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}\\
+&\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&1&1\\0&1&1\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}+\begin{pmatrix}1&1&0\\0&0&0\\0&0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\0&1&0\\0&1&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0\\1&0&1\\1&0&0\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&+ \begin{pmatrix}1&1&0\\0&0&0\\1&1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\1&1&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&1\\0&0&0\end{pmatrix} + \begin{pmatrix}1&1&0\\0&0&0\\1&0&1\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\1&0&0\\0&1&0\end{pmatrix} \otimes \begin{pmatrix}1&0&0\\0&0&1\\0&0&0\end{pmatrix}\\
&+ \begin{pmatrix}0&0&1\\0&0&0\\0&0&1\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix} \otimes \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&1&1\\0&1&1\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&1\\0&0&0\\0&1&1\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\\
&+ \begin{pmatrix}1&1&1\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\0&1&0\end{pmatrix} \otimes \begin{pmatrix}1&0&0\\1&1&0\\0&0&0\end{pmatrix} + \begin{pmatrix}1&1&1\\1&0&1\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}\\
&+ \begin{pmatrix}1&1&1\\0&1&1\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&1\\0&0&0\\0&1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&1&0\\1&0&0\end{pmatrix}
\end{aligned}
$$

**Table 5.** A general multiplication scheme with 17 parameters. The parameters are $x_1, \ldots, x_{17}$, and we use the following shortcuts:

$$
\begin{aligned}
x_{i,j} &= x_i x_j + 1 & p_1 &= x_9 x_{6,8} + x_6\\
p_2 &= x_5 x_{1,4} + x_1 & p_3 &= x_{16} x_{3,15} + x_3\\
p_4 &= x_{13} x_{11,12} + x_{11} & p_5 &= x_2 x_{10} x_{1,4} - x_{10} x_{17} x_{1,4} + x_2 x_4\\
p_6 &= x_2 x_5 x_{10} x_{1,4} - x_5 x_{17} x_{10} x_{1,4} + x_2 x_{4,5} + x_1 x_2 x_{10} - x_1 x_{17} x_{10}
\end{aligned}
$$

$$
\begin{aligned}
&\begin{pmatrix}1&-1&-1\\1&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}-1&0&0\\1&0&0\\1&0&0\end{pmatrix}\\
+ &\begin{pmatrix}1&-1&0\\0&0&0\\1&0&0\end{pmatrix} \otimes \begin{pmatrix}1&1&0\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}\\
+ &\begin{pmatrix}x_{4,5}&0&p_2\\x_{4,5}&0&p_2\\p_6&0&p_2x_{17}\end{pmatrix} \otimes \begin{pmatrix}0&-x_{1,4}&x_{1,4}\\0&0&0\\0&x_4&-x_4\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\\
+ &\begin{pmatrix}0&1&1\\0&0&0\\1&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&0\\0&0&0\\-1&0&0\end{pmatrix} \otimes \begin{pmatrix}-1&0&0\\0&0&1\\0&0&1\end{pmatrix}\\
+ &\begin{pmatrix}0&1&1\\0&0&0\\0&0&1\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&1\\0&0&1\end{pmatrix}\\
+ &\begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\1&0&-1\\1&0&-1\end{pmatrix}\\
+ &\begin{pmatrix}0&1&0\\1&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}1&1&0\\1&0&0\\-1&0&0\end{pmatrix} \otimes \begin{pmatrix}1&0&0\\-1&1&0\\0&0&0\end{pmatrix}\\
+ &\begin{pmatrix}0&x_{15,16}&0\\0&x_{15,16}&0\\0&p_3&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&-1&1\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&x_{14}x_{15}&x_{14}x_{15}\\0&-x_{3,15}&x_{15}\\0&0&0\end{pmatrix}\\
+ &\begin{pmatrix}0&1&0\\0&0&-1\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&1\\0&-1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&1&0\\1&0&0\end{pmatrix}\\
+ &\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} \otimes \begin{pmatrix}1&1&0\\1&1&0\\-1&-1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\1&-1&0\\0&0&0\end{pmatrix}\\
+ &\begin{pmatrix}x_4&0&x_{1,4}\\x_4&0&x_{1,4}\\p_5&0&x_{17}x_{1,4}\end{pmatrix} \otimes \begin{pmatrix}0&p_2&-p_2\\0&0&0\\0&-x_{4,5}&x_{4,5}\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\\
+ &\begin{pmatrix}0&0&0\\1&-1&0\\0&1&0\end{pmatrix} \otimes \begin{pmatrix}0&0&0\\-1&0&0\\1&0&0\end{pmatrix} \otimes \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&+ \begin{pmatrix} 0 & 0 & 0 \\ p_1 & 0 & x_{8,9} \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} x_7x_8 & 0 & x_8 \\ x_7x_8 & 0 & x_{6,8} \\ -x_7x_8 & 0 & -x_{6,8} \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ x_7 & -x_7 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ x_{6,8} & 0 & x_8 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} x_7x_{8,9} & 0 & x_{8,9} \\ x_7x_{8,9} & 0 & p_1 \\ -x_7x_{8,9} & 0 & -p_1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & -1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & -1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & -1 & -1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} p_4 & x_{12,13} & -x_{12,13} \\ 0 & 0 & 0 \\ p_4 & x_{10}x_{12,13} & -x_{10}x_{12,13} \end{pmatrix} \otimes \begin{pmatrix} x_{12} & 0 & -x_{12} \\ 0 & 0 & 0 \\ x_2x_{11,12} & 0 & -x_{11,12} \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} x_{11,12} & x_{12} & -x_{12} \\ 0 & 0 & 0 \\ x_{11,12} & x_{10}x_{12} & -x_{10}x_{12} \end{pmatrix} \otimes \begin{pmatrix} x_{12,13} & 0 & -x_{12,13} \\ 0 & 0 & 0 \\ p_4x_2 & 0 & -p_4 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 1 & -x_{14} & x_{14}+1 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & x_{15} & 0 \\ 0 & x_{15} & 0 \\ 0 & x_{3,15} & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & x_{14}x_{15,16} & x_{14}x_{15,16} \\ 0 & -p_3 & x_{15,16} \\ 0 & 0 & 0 \end{pmatrix} \\
&+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -x_{10} & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ x_{17} & 0 & -1 \end{pmatrix}
\end{aligned}
$$