Arithmetic Expression Construction

Leo Alcock
Harvard University, Cambridge, MA, USA
leoalcock@college.harvard.edu

Sualeh Asif
MIT, Cambridge, MA, USA
sualeh@mit.edu

Jeffrey Bosboom
CSAIL, MIT, Cambridge, MA, USA
jbosboom@csail.mit.edu

Josh Brunner
CSAIL, MIT, Cambridge, MA, USA
brunnerj@mit.edu

Charlotte Chen
MIT, Cambridge, MA, USA
charlotte_z_chen@yahoo.com

Erik D. Demaine
CSAIL, MIT, Cambridge, MA, USA
edemaine@mit.edu

Rogers Epstein
CSAIL, MIT, Cambridge, MA, USA
rogersep@mit.edu

Adam Hesterberg
Harvard University, Cambridge, MA, USA
achesterberg@gmail.com

Lior Hirschfeld
MIT, Cambridge, MA, USA
liorh@mit.edu

William Hu
MIT, Cambridge, MA, USA
whu2704@mit.edu

Jayson Lynch
MIT, CSAIL, Cambridge, MA, USA
jaysonl@mit.edu

Sarah Scheffler
Boston University, Boston, MA, USA
sscheff@bu.edu

Lillian Zhang
MIT, Cambridge, MA, USA
lillianz@mit.edu

Abstract

When can n given numbers be combined using arithmetic operators from a given subset of $\{+, -, \times, \div\}$ to obtain a given target number? We study three variations of this problem of *Arithmetic Expression Construction*: when the expression (1) is unconstrained; (2) has a specified pattern of parentheses and operators (and only the numbers need to be assigned to blanks); or (3) must match a specified ordering of the numbers (but the operators and parenthesization are free). For each of these variants, and many of the subsets of $\{+, -, \times, \div\}$, we prove the problem NP-complete, sometimes in the weak sense and sometimes in the strong sense. Most of these proofs make use of a *rational function framework* which proves equivalence of these problems for values in rational functions with values in positive integers.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Hardness, algebraic complexity, expression trees

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.12

Related Version A full version of the paper is available at `https://arxiv.org/abs/2011.11767`.

Acknowledgements This work was initiated during open problem solving in the MIT class on Algorithmic Lower Bounds: Fun with Hardness Proofs (6.892) taught by Erik Demaine in Spring 2019. We thank the other participants of that class – in particular, Josh Gruenstein, Mirai Ikebuchi, and Vilhelm Andersen Woltz – for related discussions and providing an inspiring atmosphere.

1 Introduction

Algebraic complexity theory [2, 14] is broadly interested in the smallest or fastest arithmetic circuit to compute a desired (multivariate) polynomial. An *arithmetic circuit* is a directed acyclic graph where each source node represents an input and every other node is an arithmetic operation, typically among $\{+, -, \times, \div\}$, applied to the values of its incoming edges, and one sink vertex represents the output. One of the earliest papers on this topic is Scholz's 1937 study of minimal addition chains [12], which is equivalent to finding the smallest circuit with operation + that outputs a target value t. Scholz was motivated by efficient algorithms for computing $x^n \bmod N$. Minimal addition chains have been well-studied since; in particular, the problem is NP-complete [5].

Algebraic computation models serve as a more restrictive model of computation, making it easier to prove lower bounds. In cryptography, a common model is to limit computations to a group or ring [10]. For example, Shoup [13] proves an exponential lower bound for discrete logarithm in the generic group model, and Aggarwal and Maurer [1] prove that RSA is equivalent to factoring in the generic ring model. Minimal addition chains is the same problem as minimal group exponentiation in generic groups, and thus the problem has received a lot of attention in algorithm design [7].

In our paper, we study a new, seemingly simpler type of problem, where the goal is to design an *expression* instead of a *circuit*, i.e., a *tree* instead of a *directed acyclic graph*. Specifically, the main Arithmetic Expression Construction (AEC) problem is as follows:

▶ **Problem 1** $((\mathbb{L}, \mathsf{ops})$-AEC-Std / Standard$)$**.**
Instance: A multiset of values $A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{L}$ and a target value $t \in \mathbb{L}$.
Question: Does there exist a parenthesized expression using any of the operations in ops that contains each element of A exactly once and evaluates to t?

The problem $(\mathbb{N}, \{+, -, \times, \div\})$-AEC-Std naturally generalizes two games played by humans. The 24 Game [15] is a card game dating back to the 1960s, where players race to construct an arithmetic expression using four cards with values 1–9 (a standard deck without face cards) that evaluates to 24. In the tabletop role-playing game Pathfinder, the Sacred Geometry feat requires constructing an arithmetic expression using dice rolls that evaluate to one of a specified set of prime constants.

In this paper, we prove that this problem is NP-hard when the input values are in $\mathbb{N}$ or the algebraic extension $\mathbb{N}[x_1, \ldots, x_k]$.[1]

1.1 Problem Variants and Results

Expressions can be represented as trees with all operands at leaf nodes and operators at internal nodes using Dijkstra's shunting yard algorithm [4]. Similarly, an expression tree can be converted into a parenthesized expression by concatenating the operands and operators as they are encountered with an inorder traversal, adding an opening parenthesis when descending the tree and a closing parenthesis when ascending.

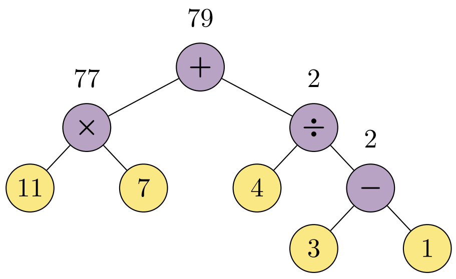

Figure 1 An example expression tree for $7 \times 11 + (4 \div (3 - 1)) = 79$. The numbers above the internal nodes indicate their values.

[1] To clarify the notation: all values are in the field extension $\mathbb{Q}(x_1, \ldots, x_k)$, but the *input* values are restricted to $\mathbb{N}[x_1, \ldots, x_k]$, i.e., have nonnegative integer coefficients.

We also consider following two variants of AEC which impose additional constraints (specified by some data we denote by D) on the expression trees:

▶ **Problem 2** (($\mathbb{L}$, ops)-AEC-EL / Enforced Leaves)**.**
Instance: A target value $t \in \mathbb{L}$ and a multiset of values $A = \{a_1, \dots, a_n\} \subseteq \mathbb{L}$ with the leaf order encoded by $D : A \to [n]$.
Question: Can an expression tree be formed such that each internal node has an operation from ops, and the leaves of the tree are the list A in order D, where the tree evaluates to t?

▶ **Problem 3** (($\mathbb{L}$, ops)-AEC-EO / Enforced Operations)**.**
Instance: A multiset of values $A = \{a_1, a_2, \dots, a_n\} \subseteq \mathbb{L}$, a target $t \in \mathbb{L}$, and an expression tree D with internal nodes each containing an operation from ops and empty leaf nodes.
Question: Can the expression tree be completed by assigning each value in A to exactly one leaf node such that the tree evaluates to t?

The first variant fixes the ordering of leaf nodes of the tree, and asks whether an expression can be formed which reaches the target. The second variant constrains the shape of the tree and internal node operations, and asks whether an ordering of the leaves can be found which evaluates to the target. We represent all instances of these variants by triples (A, t, D) where $A = \{a_1, a_2, \dots a_n\}$ is a multiset of values, t is the target value, and D is additional data for the instance: a leaf ordering for EL, and an expression tree for EO.

In this paper, we prove hardness results in all of these variants by reduction from PARTITION and related problems listed in Appendix A, and develop polynomial or pseudopolynomial algorithms where appropriate. Table 1 summarizes our results. In particular, we prove NP-hardness with $\mathbb{L} = \mathbb{N}$ for the Standard and EO variants for all subsets of operations $\{+, -, \times, \div\}$. Note that all of these problems are in NP: simply evaluate the expression given as a certificate.

Table 1 Our results for Arithmetic Expression Construction. Bold font indicates NP-completeness results that are tight; for weakly NP-complete results, this means that we have a corresponding pseudopolynomial-time algorithm. The proof is given in the section in parentheses, or if no number is given, in the full paper.

Operations	Standard	Enforced Operations	Enforced Leaves
$\{+\}$	$\in$ P	$\in$ P	$\in$ P
$\{-\}$	**weakly NP-complete**	**weakly NP-complete**	**weakly NP-complete**
$\{\times\}$	$\in$ P	$\in$ P	$\in$ P
$\{\div\}$	**strongly NP-complete**	**strongly NP-complete**	**strongly NP-complete**
$\{+, -\}$	**weakly NP-complete**	**weakly NP-complete**	**weakly NP-complete**
$\{+, \times\}$	weakly NP-complete (§3)	weakly NP-complete[a] (§5)	**weakly NP-complete**
$\{+, \div\}$	weakly NP-complete	**strongly NP-complete**	Open
$\{-, \times\}$	weakly NP-complete	**strongly NP-complete**	weakly NP-complete
$\{-, \div\}$	weakly NP-complete	**strongly NP-complete**	Open
$\{\times, \div\}$	**strongly NP-complete**	**strongly NP-complete**	**strongly NP-complete**
$\{+, -, \times\}$	weakly NP-complete (§3)	**strongly NP-complete**	weakly NP-complete (§4)
$\{+, -, \div\}$	weakly NP-complete	**strongly NP-complete**	Open
$\{+, \times, \div\}$	weakly NP-complete (§3)	**strongly NP-complete**	weakly NP-complete
$\{-, \times, \div\}$	weakly NP-complete	**strongly NP-complete**	Open
$\{+, -, \times, \div\}$	weakly NP-complete (§3)	**strongly NP-complete**	Open

[a] Strong in all variables except the target t

Our first step is to show that, for any k and k', there is a polynomial-time reduction from the k-variable variant to the k'-variable variant. Such a reduction is trivial for $k \leq k'$ by leaving the instance unchanged. For the converse, we present the *Rational Function Framework* in Section 2, which provides a polynomial-time construction of a positive integer B on an instance I (i.e., set of values $\{a_i\}, t \in \mathbb{N}[x_1, \ldots, x_k]$) such that replacing $x_k = B$ yields a solvable instance if and only if I is solvable. That is, for all variants $\mathsf{var} \in \{\textsc{Std}, \text{EO}, \text{EL}\}$, we obtain a simple reduction

$$(\mathbb{N}[x_1, \ldots, x_k], \mathsf{ops})\text{-AEC-}\mathsf{var} \rightarrow (\mathbb{N}[x_1, \ldots, x_{k-1}], \mathsf{ops})\text{-AEC-}\mathsf{var}$$

Because this reduction preserves algebraic properties, it yields interesting positive results in addition to hardness results. For example, this result demonstrates that $(\mathbb{N}[x_1, \ldots, x_k], \{+, -\})$-AEC-Std has a pseudopolynomial-time algorithm via a chain of reductions to $(\mathbb{N}, \{+, -\})$-AEC-Std which is equivalent to the classic Partition problem.

1.2 Notation

Beyond the $I = (A, t, D)$ instance notation introduced above, we often use the variable E to denote an expression; the Standard variant is to decide whether $\exists E : E(A) = t$. We also use "ev($\cdot$)" to denote the value of an expression at a node of an expression tree (i.e., the evaluation of the subtree whose root is that node).

1.3 Outline of Paper

In Section 2, we describe the Rational Function Framework which demonstrates equivalence between AEC variants over different numbers of free variables. In Section 3, we present the structure theorem which will be used to prove hardness of the nontrivial cases of Standard and we present a proof of the full case with it. In Section 4, and Section 5, we sketch two selected interesting reductions for Enforced Leaves and Enforced Operations respectively. The rest of our hardness proofs along with pseudopolynomial algorithms for some weakly NP-hard problems can be found in the full paper. Appendix A lists the problems we reduce from for our hardness proofs.

2 Rational Function Framework

In this section, we present the rational function framework. This framework proves the polynomial-time equivalence of all Arithmetic Expression Construction variants with values as ratios of polynomials with integer coefficients, that is, $\mathbb{Q}(x_1, \ldots, x_k)$, for differing k. This equivalence also allows us to restrict to $\mathbb{N}[x_1, \ldots, x_k]$ and critically will make proving hardness for variants over $\mathbb{N}$ easier by allowing us to reduce to $\mathbb{N}[x_1, \ldots, x_k]$ versions.

▶ **Theorem 1.** *For all* $\mathsf{ops} \subseteq \{+, -, \times, \div\}$*, for all variants* var*, for all integers* $k > 0$*, there exists an efficient algorithm* $\mathcal{A}$ *mapping instances* I *to positive integers* $\mathcal{A}(I)$ *such that a polynomial-time reduction*

$$(\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})\text{-}AEC\text{-}\mathsf{var} \rightarrow (\mathbb{Q}(x_1, \ldots, x_{k-1}), \mathsf{ops})\text{-}AEC\text{-}\mathsf{var}$$

is given by substituting $x_k = B$ *in an instance* I *for any* $B \in \mathbb{N}$ *satisfying* $B \geq \mathcal{A}(I)$*.*

To formalize the idea of a "big enough" B, we introduce the concept of *sufficiency* of integers for instances of AEC. Let B be a positive integer and let $I = (A, t = f_t/g_t, D)$ be a $(\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})$-AEC-$\mathsf{var}$ instance. Loosely, we consider B to be $(I, \mathsf{ops}, \mathsf{var})$-*sufficient* if substituting $x_k = B$ in instance I creates a valid reduction on I, as in Theorem 1.

We will shorten the terminology and call this *I-sufficient* or *sufficient for I* when ops and var are clear from context. Theorem 1 says there is an efficient algorithm that produces sufficient integers. Note that this definition is not yet rigorous. To remedy this we introduce the paired model of computation on rational functions.

In the paired model of computation, objects are given by pairs (f, g) of integer-coefficient polynomials $f, g \in \mathbb{Z}[x_1, \ldots, x_k]$. Intuitively, the paired model simulates rational functions by $(f, g) \leftrightarrow f/g$. We define operations $(+, -, \times, \div)$ and equivalence relation $(\sim)$ on pairs (a, b) and (f, g) as follows:

$$\begin{aligned}
&(f, g) + (a, b) = (fb + ga, gb)\\
&(f, g) - (a, b) = (fb - ga, gb)\\
&(f, g) \times (a, b) = (fa, gb)\\
&(f, g) \div (a, b) = (fb, ga)\\
&(f, g) \sim (a, b) \Leftrightarrow fb = ga
\end{aligned}$$

As mentioned, the intuition is that f is the numerator and g is the denominator of a ratio of polynomials with integer coefficients. The utility of the model is that it keeps track of rational functions as *specific* quotients of integer coefficient polynomials. This will remove the ambiguity of representation of elements in $\mathbb{Q}(x_1, \ldots, x_n)$. Such a model allows us to make arguments about which polynomials can occur in the numerator and denominator of a rational function, such as by defining the range of these polynomials.

We can define Arithmetic Expression Construction in the paired model for all variants by changing target and values into pairs and using all the operations as defined above. An instance in the paired model is solvable if there exists an expression E in values from A and satisfying conditions imposed by D such that given $(f, g) = E(A)$, we have $(f, g) \sim t = (f_t, g_t)$. For example, in enforced leaves, the entries of leaves of E must be in the order specified by D, and in enforced order, the expression E is already specified and one must reorder A. The only difference is that we now compute in the paired model rather than with rational functions.

Similarly, note that one can convert instances in the paired model to the nonpaired model via mapping entries $(f_i, g_i) \mapsto f_i/g_i$ and for a nonpaired model, one can always write $r \in \mathbb{Q}(x_1, \ldots, x_k)$ as f_i/g_i where f_i, g_i have integer coefficients.[2] A paired instance of AEC is solvable if and only if it's nonpaired variant is solvable. We now rigorously define sufficiency in Definition 2 and characterize its use in Lemma 3.

▶ **Definition 2.** *Let B be a positive integer, and $I = (A, t = f_t/g_t, D)$ be an instance of $(\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})$-AEC-var. Represent I in the paired model. Suppose that, for every evaluation $(f, g) = E(A)$ of a valid AEC expression E (as restricted by D) in the paired model, the norms of the coefficients of fg_t and f_tg are all less than $B/2$. Then B is $(I, \mathsf{ops}, \mathsf{var})$-sufficient.*

▶ **Lemma 3.** *Given an instance $I = (A, t = f_t/g_t, D)$ of $(\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})$-AEC-var and $B \in \mathbb{N}$ which is I-sufficient as defined above. Let $E(\cdot)$ be some expression from a valid ops expression tree according to D. Then, for every evaluation of E over the polynomials in A, one has:*

$$\begin{aligned}
&E\left(\{(a_i(x_1, \ldots, x_k)\}_{a_i \in A}\right) = t(x_1, \ldots, x_k)\\
&\qquad\qquad\qquad\qquad \Leftrightarrow E\left(\{a_i(x_1, \ldots, x_{k-1}, B)\}_{a_i \in A}\right) = t(x_1, \ldots, x_{k-1}, B).
\end{aligned}$$

[2] Note that this representation is not unique!

The proof of this lemma can be found in the full paper. Essentially, this lemma shows that constructing I-sufficient integers efficiently is sufficient to prove our main theorem. The rest of this section is dedicated to the polynomial-time construction of I sufficient integers B by an algorithm $\mathcal{A}$.

Let

$$m(f) := \binom{\deg(f) + k}{\deg(f)}$$

where $m(f)$ is the maximum number of terms a k-variable polynomial f of degree $\deg(f)$ can have. Let maxcoeff(f) denote the max of all of the *norms* of coefficients of f. That is,

$$\operatorname{maxcoeff}(f) = \max_c\{|c| : c \text{ coefficient of } f\}.$$

Now we are ready to present an integer sufficient for an instance.

▶ **Lemma 4.** *Let $I = (A, t, D)$ be an instance of $(\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})$-AEC-$\mathsf{var}$. Then*

$$B = 2m(t)\operatorname{maxcoeff}(t)(2Mq)^n$$

is sufficient for I, where $n = |A|$, $q := \max_{f_i/g_i \in A}(\operatorname{maxcoeff}(f_i), \operatorname{maxcoeff}(g_i))$ is the largest coefficient appearing in a paired polynomial within A, and $M = \sum_{a_i \in A} m(a_i)$.

▶ Remark. The algorithm presented in the proof (found in the full paper) gives a large B that will give blowup sizes which are unnecessary for most AEC instances. One key use of sufficiency is to facilitate proofs with lower blowup. Often times we will have the following situation: We will give a reduction from a partition-type problem P to $(\mathbb{Q}(x_i), \mathsf{ops})$-AEC-$\mathsf{var}$ and construct $(I, \mathsf{ops}, \mathsf{var})$-sufficient B such that the composition

$$P \to (\mathbb{Q}(x_1, \ldots, x_k), \mathsf{ops})\text{-AEC-}\mathsf{var} \to (\mathbb{N}, \mathsf{ops})\text{-AEC-}\mathsf{var}$$

is a valid reduction.

2.1 Possible Generalizations to the Rational Framework

In this section, we informally explore the possibility of extending the rational framework to the problems more general than expression construction, such as circuits. The generalization to circuits naturally becomes an arithmetic version of the Minimum Circuit Size Problem.

The original Minimum Circuit Size Problem (MCSP) [9] asks if given a truth table and an integer k, can you construct a boolean circuit of size at most k that computes the truth table; this problem has many connections throughout complexity theory. A new variant, "Arithmetic MCSP" would ask if given n values in $\{a_1, \ldots, a_n\} \subseteq \mathbb{L}$, within $0 < k < n$ operations from $\{+, -, \times, \div\}$ can you construct a target $t \in \mathbb{L}$?[3] For $\mathbb{L} = \mathbb{Q}(x_1, \ldots, x_k)$, this problem asks whether a given rational function is constructible by an arithmetic circuit of size at most k starting from a set of rational functions. It would be very useful if the rational framework could be adapted for Arithmetic MCSP; this would demonstrate an equivalence between the problem of circuit construction of rational functions and of reaching a rational number given input rational numbers.

[3] Note that since we can reuse values here, picking k to be less than n is the same as picking k to be bounded by a fixed polynomial $p(n)$ by a padding argument. That is, you can reduce from this problem where you specify $k < p(n)$ to $k < n$ by padding any given instance A with $\approx p(n)$ copies of a_1. This is similar to the proof that linear space simulation is PSPACE complete.

Unfortunately, the reduction methods provided above do not work naively for circuits: Given a polynomial-sized “sufficient” B as presented, and a polynomial of the form cx, the term $(c^{2^k}x^{2^k})$ is formable by repeated squaring. That is, we can form superpolynomial coefficents that will be bigger than B. This removes the concept of “sufficiency” which is a key requirement for the rational framework as it is.

On the bright side, the rational framework should work for Arithmetic Minimum Formula Constructions. Arithmetic formulae are expression trees with internal nodes operations $\{+,-,\times,\div\}$ except that one may use the input values in A a flexible number of times. This is analogous to Boolean formulae; indeed, Minimum Boolean Formula problems [3, 8] have also received significant attention. We can define Arithmetic Minimum Formula Construction as follows: Given multiset $A \subseteq \mathbb{L}$, target t, $0 < k < n$, can you give a formula of size at most k with values in A which reaches a target $t \in \mathbb{L}$?

The intuitive reason that the rational framework should hold in this case is because formulae still have a tree structure and the number of leaves is polynomial. Thus, the same proofs in the rational framework will carry over. However, we expect the complexity and hardness proofs for this family of problems should be very different than those in this paper. All the reductions in this paper are from Partition-type problems, which allow for at most a single use of each input number. Hardness of this family of problems and generalizations of the rational function framework are interesting areas for further study.

3 Arithmetic Expression Construction Standard Results

In this section, we provide NP-hardness proofs for operations $\{+,\times\} \subseteq S \subseteq \{+,-,\times,\div\}$ of the Standard variant of Arithmetic Expression Construction. In the full paper, we give similar reductions that cover all other subsets of operations.

All of these results use the rational function framework described in Section 2.

First, we outline some proof techniques that are used in this section to both combine proofs of results from differing sets of operations as well as simplify them. The first comes from the observation that if an instance of $(\mathbb{L}, S)$-AEC-STD is solvable, then for any operation set $S' \supset S$, the same instance will be solvable in $(\mathbb{L}, S')$-AEC-STD. This allows us to bundle reductions to several AEC-STD cases simultaneously by giving a reduction (R) from some partition problem P to $(\mathbb{L}, S)$-AEC-STD and proving that if any constructed instance is solvable in $(\mathbb{L}, S')$-AEC-STD, the partition instance is also solvable. That is, we have the following implications:

$$\begin{tikzcd}
P\text{-instance } x \text{ Solvable} \arrow[r, Rightarrow] & R(x) \text{ is } S\text{-Solvable} \arrow[d, Rightarrow] \\
& R(x) \text{ is } S'\text{-Solvable} \arrow[ul, Rightarrow]
\end{tikzcd}$$

▶ **Theorem 5.** *Standard $\{+,\times\} \subseteq S \subseteq \{+,-,\times,\div\}$ is weakly NP-hard by reduction from SQUAREPRODUCTPARTITION-$n/2$.*

We spend the remainder of this section proving this theorem.

We will reduce from SQUAREPRODUCTPARTITION-$n/2$ (defined in Appendix A) to $(\mathbb{Z}[x,y,z], S)$-AEC-STD. On an instance $\{a_1, \ldots, a_n\}$ with all $a_i \geq 2$,[4] of SQUAREPRODUCT-PARTITION-$n/2$ construct the following:

[4] We can assume this property with loss of generality by replacing all a_i with $2a_i$.

Let

$$B_y = y - x^{n/2}\sqrt{\prod_i a_i};\ B_z = z - x^{n/2}\sqrt{\prod_i a_i}.$$

We then construct the instance of Arithmetic Expression Construction with input set $A = \{B_y, B_z\} \cup \{a_i x\}_i$ and target $t = yz$. Here the square root of the product of all a_i is the value we want each partition to achieve, the polynomial $x^{n/2}$ will help us argue that we must multiply all of our a_i values, and B_y, B_z are gadgets which will force a partitioned tree structure as given by Theorem 8. Methods from Section 2 allow us to construct a reduction by replacing x, y, and z with *sufficient* integers B_1, B_2 and B_3.

It is clear that if the SquareProductPartition-$n/2$ is solvable then this AEC instance is solvable with operations $\{+, \times\} \subseteq S$. On the partition with equalized products, partition the $a_i x$ terms into corresponding sets and take their products to get two polynomials of value $x^{n/2}\sqrt{\prod_i a_i}$. Then form $(B_y + x^{n/2}\sqrt{\prod_i a_i})(B_z + x^{n/2}\sqrt{\prod_i a_i}) = yz$.

Next, we prove the converse via contradiction by proving the following theorem that will be useful for the other AEC-Std cases. This theorem shows that any expression tree which evaluates to target $t \approx yz$ on an instance of similar structure to the constructed instance above must have a very particular partitioned structure described in Theorem 8. This will be the key to showing the soundness of our reduction. We use $\mathrm{ev}(T)$ to refer to the evaluation of the subtree rooted at node T.

Before stating Theorem 8, we first introduce the concept of $\mathbb{Q}(x)$-equivalence and give a couple of characterizations of it:

▶ **Definition 6.** *Given a field K with a subfield F, for $L_1, L_2 \in K - F$, we say L_1 and L_2 are F-equivalent (written $L_1 \sim_F L_2$) if by a sequence of operations between L_1 and elements of F we can form L_2.*

The following lemma gives an alternate characterization of $\sim_F$:

▶ **Lemma 7.** *$\sim_F$ is an equivalence relation and $L_2 \sim_F L_1$ if and only if for some $c_i, d_i \in F$ with $c_1 d_2 - c_2 d_1 \neq 0$,*

$$L_2 = \frac{c_1 L_1 + d_1}{c_2 L_1 + d_2}$$

We will refer to $\mathbb{Q}(x)$ equivalence with respect to $\mathbb{Q}(x)$ as a subfield of $\mathbb{Q}(x, y, z)$.

We now state our structure theorem:

▶ **Theorem 8.** *For any $S \subseteq \{+, -, \times, \div\}$, let I be a solvable $(\mathbb{Q}(x, y, z), S)$-AEC-Std instance with entries of the form $\{B_y, B_z\} \cup \{r_i(x)\}_i$ where $B_y \sim_{\mathbb{Q}(x)} y, B_z \sim_{\mathbb{Q}(x)} z$, and $r_i \in \mathbb{Q}[x]$ and target t with $t \sim_{\mathbb{Q}(x)} yz$. Then any solution expression tree for I has the form depicted in Figure 2: The operation at the least common ancestor of leaves B_y and B_z, denoted N, is $\times$ or $\div$, and $\mathrm{ev}(N) = (lyz)^{\pm 1}, l \in \mathbb{Q}(x)$. For T_y, T_z the children of N containing B_y, B_z respectively, $\mathrm{ev}(T_y) = (ay)^{\pm 1}, \mathrm{ev}(T_z) = (a'z)^{\pm 1}$, where $a, a' \in \mathbb{Q}(x)$.*

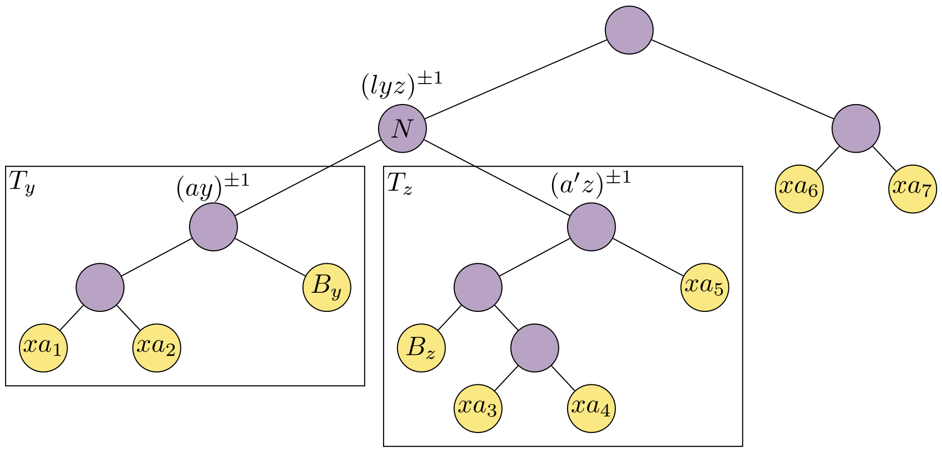

Figure 2 Example expression tree for standard $\{+, -, \times, \div\}$.

Proof. In our expression tree T, N is the least common ancestor between B_y and B_z. One has that

$$\text{ev}(N) = \frac{eyz + f}{gyz + h}, eh - gf \neq 0, e, f, g, h \in \mathbb{Q}(x)$$

since $\text{ev}(N)$ is combined with a sequence of operations with elements in $\mathbb{Q}(x)$ to form t. That is, it is $\mathbb{Q}(x)$ equivalent to yz.

Let T_y be the child of N containing B_y as a leaf and T_z the child of N containing B_z. A priori we know

$$\text{ev}(T_y) = \frac{ay + b}{cy + d}, \text{ev}(T_z) = \frac{a'z + b'}{c'z + d'}, \text{ev}(N) = \frac{eyz + f}{gyz + h},$$
$$ad - bc \neq 0, a'd' - b'c' \neq 0, eh - fg \neq 0, a, b, c, d, a', b', c', d', e, f, g, h \in \mathbb{Q}(x)$$

by similar $\mathbb{Q}(x)$-equivalence arguments. The rest of the proof is casework done via trying out different operations at N. We will see that if the operation is $\times, \div$ then the evaluations must be of the form described in the statement of the theorem and that if the operation is $\pm$ then we reach a contradiction.

First we check the case that the operation at N is $\times$. For this argument we'll reduce to a set of equations in $\mathbb{Q}(x)[y, z]$ and make some divisibility arguments using the fact that this is a unique factorization domain.

$$\frac{ay + b}{cy + d} \cdot \frac{a'z + b'}{c'z + d'} = \frac{eyz + f}{gyz + h}$$
$$\Rightarrow (ay + b)(a'z + b')(gyz + h) = (cy + d)(c'z + d')(eyz + f)$$

If both $e, f \neq 0$, then $eyz + f$ is irreducible and since $eyz + f | (ay + b)(a'z + b')(gyz + h)$ we find that $eyz + f | gyz + h$ and $\frac{eyz+f}{gyz+h} = l \in \mathbb{Q}(x)$. However, this would contradict $\text{ev}(N) \sim_{\mathbb{Q}(x)} yz$. We conclude that exactly one of e, f is nonzero. A similar argument with $gyz + h$ allows us to conclude that at most one of g, h is nonzero. We cannot have $g = 0$ and $e = 0$, or we would have $\text{ev}(N) \in \mathbb{Q}(x)$. This reduces us to the case that $\text{ev}(N) = (lyz)^{\pm 1}$. We now have one of the two cases:

$$(ay + b)(a'z + b') = lyz(cy + d)(c'z + d') \tag{1}$$
$$lyz(ay + b)(a'z + b') = (cy + d)(c'z + d') \tag{2}$$

For the first case to hold one must have $c = c' = 0$ for the degrees in y and z to match up. Given $c = c' = 0$, one must also have $b = b' = 0$ so that the right hand side of the equation is divisible by yz. A similar argument for the second case yields $a = a' = d = d' = 0$. For multiplication, this case is covered. If the operation is division, one gets the relation:

$$\frac{ay+b}{cy+d} \div \frac{a'y+b'}{c'y+d'} = \frac{ay+b}{cy+d} \cdot \frac{c'y+d'}{a'y+b'} = \frac{eyz+f}{gyz+h}$$

and the same argument follows through.

Next we show that the operation at N can not be $+$:

$$\frac{ay+b}{cy+d} + \frac{a'z+b'}{c'z+d'} = \frac{eyz+f}{gyz+h}$$
$$((ac'+a'c)yz + (ad'+b'c)y + (bc'+a'd)z + (bd'+b'd))(gyz+h)$$
$$= (cy+d)(c'z+d')(eyz+f) \quad (3)$$

Starting with a similar divisibility argument, if $g, h \neq 0$, we find that $gyz + h$ is irreducible and that $gyz + h | eyz + f$, $\frac{eyz+h}{gyz+f} \in \mathbb{Q}(x)$. Thus either $g = 0$ or $h = 0$.

Suppose $g = 0$. Then we must have $e \neq 0$ to maintain $\mathrm{ev}(N) \sim_{\mathbb{Q}(x)} yz$. With nonzero e, one must have that $c = c' = 0$ so that the RHS of equation (9) has degree no bigger than the left hand side. The coefficient of yz on the LHS of the equation is $(ac' + a'c)h = 0$ and the coefficient of yz on the RHS is edd' which must be nonzero and thus we get a contradiction.

Suppose $h = 0$. We must have $g, f \neq 0$ to maintain $\mathrm{ev}(N) \sim_{\mathbb{Q}(x)} yz$. The LHS of the equation is divisible by yz. Thus $yz|(cy + d)(c'z + d')(eyz + f)$ and this can only occur if $d = d' = 0$ and $c, c' \neq 0$. Expanding the equations now and looking at the coefficient of yz in the LHS and RHS we find: $0 \neq cc'f = g(bd' + b'd) = 0$. This concludes the proof of our helper theorem. ◀

Now we will return to our proof of the soundness of the reduction to AEC-Std. Suppose that the constructed instance I is solvable and the product partition instance is not solvable. Then for some $S \in \{\mathrm{leaves}(T_y) \cap \{a_i x\}, \mathrm{leaves}(T_z) \cap \{a_i x\}\}$, either

1. S contains $< n/2$ leaves $a_i x$.
2. S contains $n/2$ leaves $a_i x$ with product $\alpha x^{n/2}$ with $\alpha < \sqrt{\prod_i a_i}$.

WLOG let this set be $\mathrm{leaves}(T_y) \cap \{a_i x\}$. In the next two claims, we prove that in neither of these two cases can a subtree evaluate to an expression of the form $(ay)^{\pm 1}$ as Theorem 8 requires.

▷ Claim 9. If T_y contains $< n/2$ leaves $\{a_i x\}$ and $y' = y - x^{n/2}\sqrt{\prod a_i}$, then $\mathrm{ev}(T_y)$ is not of the form $(ay)^{\pm 1}$ for any $a \in \mathbb{Q}[x]$.

Proof. The value of any subtree can be written in the form $\frac{p(x,y')}{q(x,y')}$ for polynomials p and q. Let $\deg_x(\frac{p(x,y')}{q(x,y')}) = \max(\deg_x(p(x,y')), \deg_x(q(x,y')))$. This degree is subadditive for the four arithmetic operations $(+, -, \times, \div)$. Also, if $\deg_x(p \pm q) \leq 0$, $\deg_x(p * q) \leq 0$, or $\deg_x(p/q) \leq 0$, then $\deg_x(p) = \deg_x(q)$.

By induction, the degree in x (resp. to y') at a node A is at most the number of leaves of A's subtree of the form $a_i x$. This is true for the leaves ($\deg_x(a_i x) = 1$), and subadditivity proves it for the inductive step.

Hence $\mathrm{ev}(T_y)$ has degree at most 1 in y' and less than $n/2$ in x. If $\mathrm{ev}(T_y) = (ay)^{\pm 1} = (a(y' + x^{n/2})^{\pm 1})$ for nonzero $a \in \mathbb{Q}(x)$, then it has degree at least $n/2$ in x, a contradiction. ◁

▷ Claim 10. If T_y contains $n/2$ leaves $a_i x$ with $\prod_i a_i = \alpha < \sqrt{\prod_i a_i}$ and $y' = y - x^{n/2}\sqrt{\prod a_i}$, then $\mathrm{ev}(T_y)$ is not of the form $(ay)^{\pm 1}$ for any $a \in \mathbb{Q}(x)$.

Proof. First, we rewrite our target $\mathrm{ev}(T_y)$ in terms of y', yielding $\mathrm{ev}(T_y) = (a(y' + x^{n/2}\sqrt{\prod a_i}))^{\pm 1}$. We will first show that regardless of the value of a, the maximum coefficient of the rational function $\mathrm{ev}(T_y)$ is at least $\sqrt{\prod a_i}$. Note that since y' is not in $\mathbb{Q}(x)$, $(y' + x^{n/2}\sqrt{\prod a_i})$ is an irreducible polynomial in x, so the denominator of a will never cancel out with anything. Thus, we only consider the numerator of a. Consider the leading coefficient of the numerator of the product. This leading coefficient must be exactly the product of the leading coefficient of the numerator of a and $x^{n/2}\sqrt{\prod a_i}$. Since the leading coefficient of the numerator of a is an integer, it must be at least 1, so the leading coefficient of the numerator of $a(y' + x^{n/2}\sqrt{\prod a_i})$ must be at least $x^{n/2}\sqrt{\prod a_i}$.

From our reduction we have that all the a_i are at least 2, and the largest possible integer that can be generated from the a_i and arithmetic operations is their product α. Every coefficient of $\mathrm{ev}(T_y)$ is some combinations of arithmetic operations of the a_i since it is comprised of the $a_i x$ and y' and arithmetic operations. Thus, it is not possible for $\mathrm{ev}(T_y)$ to ever have a coefficient of at least $x^{n/2}\sqrt{\prod a_i}$. Thus, from the above argument it cannot be of the form $(a(y' + x^{n/2}\sqrt{\prod a_i}))^{\pm 1}$. ◁

Note that the proof of this claim yields a reduction from SquareProductPartition-$n/2$ to $(\mathbb{Z}[x,y,z], S)$-AEC-Std for all $\{+,\times\} \subseteq S \subseteq \{+,-,\times,\div\}$. Using our rational function framework, we get a reduction from $(\mathbb{Z}[x,y,z], S)$-AEC-Std to $(\mathbb{Z}, S)$-AEC-Std by replacements[5] based on instance I with

$$x = B_1 = \mathcal{A}(I), y = B_2 = \mathcal{A}(I(B_1)), z = \mathcal{A}(I(B_1, B_2)).$$

However, since the reduction is of the form

$$\{y - \alpha x^{n/2}, z - \alpha x^{n/2}\} \cup \{a_i x\},$$

if we replace B_2 with $B_2' = \max(B_2, 1 + \alpha B_1^{n/2})$, and B_3 with $B_3' = \max(\mathcal{A}(I(B_1, B_2')), 1 + \alpha(\mathcal{A}((B_1, B_2')))^{n/2})$ this will yield still sufficient B_2, B_3 such that the composition of these maps is a reduction from ProductPartition-$n/2$ to $(\mathbb{N}, S)$-AEC-Std.

4 Arithmetic Expression Construction Enforced Leaves $\{+,-,\times\}$

Recall that an instance of the Enforced Leaves (EL) AEC variant has a fixed ordering of leaves (operands), and the goal is to arrange the internal nodes of the expression tree such that the target t is the result of the tree's evaluation. In this section we present a proof sketch for the weak NP-hardness of $(\mathbb{N}[x,y], \{+,-,\times\})$-AEC-EL. Using the technique described in Section 2, this also proves NP-hardness of $(\mathbb{N}, \{+,-,\times\})$-AEC-EL. In the full paper, we provide the full proof and present additional hardness proofs for operation sets $\{-,\times\}, \{+,\times\}, \{+,\times,\div\}$.

Our proof is a reduction from SetProductPartitionBound-K. This strongly NP-hard problem asks if given a set (without repetition) of positive integers $A = \{a_1, a_2, \ldots, a_n\}$ where all $a_i > K$ and all prime factors of all a_i are also greater than K, we can partition A into two subsets with equal products. The problem is also defined formally in Appendix A.

[5] Note that this denotes replacing with B_i which are I-sufficient but since this is done via three reductions the instance I changes. Therefore, when replacing with B_2, you need B_2 to be $I(B_1)$ sufficient (i.e., the instance I with $x = B_1$ replaced). Similar requirements hold for B_3.

▷ Claim 11. $(\mathbb{N}[x,y], \{+,-,\times\})$-AEC-EL is weakly NP-hard.

Proof sketch. This statement is proved via reduction from SetProductPartitionBound-3. Let the instance be $A = \{a_1, \ldots, a_n\}$, where all prime factors of all $a_i \in A$ (and all a_i themselves) are greater than 3. We find n unique primes p_i with some additional properties specified in the appendix. Let $L = 2^n \prod_{i\in[n]} a_i$. Then for each a_i we can construct terms b_i and c_i such that $b_i + c_i = (La_i)p_i y$ and $b_i - c_i = (L/a_i)p_i y$. We set our target polynomial as $t(x,y) = L^n x^{n-1} y^n \prod_{i\in[n]} p_i$, and we enforce the following order of leaves:

$$b_1 \quad c_1 \quad x \quad b_2 \quad c_2 \quad x \quad \cdots \quad x \quad b_n \quad c_n$$

If an instance of this product partition variant is solvable, then the constructed instance evaluates to $t(x,y) = L^n x^{n-1} y^n \prod_{i\in[n]} p_i$ when we have $(b_i + c_i)$ for a_i in one partition and $(b_i - c_i)$ for a_i in the other, and the $\times$ operator at every other node. The partition corresponds to whether the a_i was written as a difference or a sum.

We must also show that any expression achieving the target *must* take the form above. We restrict the set of possible forms by (1) inducting to show that each subtree of a solution must have degree in x equal to its number of leaves of value x, (2) counting primes factors of the highest degree term to show that subtrees with no x values must be of form $\{\pm b_i, \pm c_i, \pm b_i \pm c_i\}$, (3) a divisibility argument to show that sums of elements of form $\{\pm b_i, \pm c_i, \pm b_i \pm c_i\}$ as appearing in any evaluation of a subtree is nonzero, and (4) an argument on the degree of y for terms with degree 0 in x to show that these sums can never be cancelled. ◁

In the full paper, we expand on details and rigorously prove that the final evaluation must be of the described form.

5 Arithmetic Expression Construction Enforced Operations $\{+, \times\}$

This section concerns the Enforced Operations (EO) variant of AEC. Here, we give a short proof for the NP-hardness of $(\mathbb{N}, \{+,\times\})$-AEC-EO; see the full paper for straightforward proofs for all the other operation sets. Note that for Enforced Operations, if we prove hardness for enforced operations with set S, we have also proved it for all $S' \supset S$, since in Enforced Operations, the expression tree can be restricted to using operations in reductions.

▷ Claim 12. $(\mathbb{N}, \{+,\times\})$-AEC-EO is weakly NP-hard.

Proof. This proof proceeds by reduction from 3-Partition-3, which is 3-Partition with the extra restriction that all the subsets have size 3. Given an instance of 3-Partition-3, $A = \{a_1, a_2, \cdots, a_n\}$, construct instance I_A of $(\mathbb{N}, \{+,\times\})$-AEC-EO with the same set of values A, target $t = \left(\frac{S}{n/3}\right)^{n/3}$, where $S = \sum_i a_i$, and expression-tree:

$$(\square + \square + \square) \times (\square + \square + \square) \times \cdots \times (\square + \square + \square),$$

where there are $n/3$ pairs of parentheses and 3 positive integers between each pair of parentheses.

Given a solution of the 3-Partition-3 instance, one can use the same partition to fill in the 3-sums and solve our $(\mathbb{N}, \{+,\times\})$-AEC-EO instance. If the constructed instance is solvable, we claim that each expression $(\square + \square + \square)$ must have equal value. Denote the value of the ith $(\square + \square + \square)$ by s_i. Since $\sum_i s_i = S$, the arithmetic mean-geometric mean inequality yields $\prod_{i=1}^{n/3} s_i \leq \left(\frac{S}{n/3}\right)^{n/3}$, with equality occurring if and only if $s_i = \frac{S}{n/3}$ for all i. This completes the proof. ◁

References

1 Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, pages 36–53, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 2009.

3 David Buchfuhrer and Christopher Umans. The complexity of boolean formula minimization. In Luca Aceto, Ivan Damgard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, pages 24–35, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

4 E. W. Dijkstra. ALGOL-60 translation. Technical Report MR 34/61, Rekenafdeling, Stichting Mathematisch Centrum, 1961. URL: `https://ir.cwi.nl/pub/9251`.

5 Peter Downey, Benton Leong, and Ravi Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 10(3):638–646, 1981.

6 Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 2002.

7 Daniel M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.

8 Edith Hemaspaandra and Henning Schnoor. Minimization for generalized boolean formulas. arXiv:1104.2312, 2011.

9 Valentine Kabanets and Jin yi Cai. Circuit minimization problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 73–79, Portland, OR, 2000.

10 Ueli Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding*, pages 1–12, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

11 C. T. Ng, M. S. Barketau, T. C. E. Cheng, and Mikhail Y. Kovalyov. "Product Partition" and related problems of scheduling and systems reliability: Computational complexity and approximation. *European Journal of Operational Research*, 207(2):601–604, 2010. `doi:10.1016/j.ejor.2010.05.034`.

12 Arnold Scholz. Aufgaben und Lösungen 253. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 47:41–42, 1937.

13 Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

14 Joachim von zur Gathen. Algebraic complexity theory. In *Annual Review of Computer Science*, volume 3, pages 317–347. Annual Reviews Inc., 1988.

15 Wikipedia. 24 game. `https://en.wikipedia.org/wiki/24_Game`.

A Related Problems

To show the NP-hardness of the variants of Arithmetic Expression Construction, we reduce from the following problems:

▶ **Problem 4** (Partition).
Instance: A multiset of positive integers $A = a_1, a_2, \dots, a_n$.
Question: Can A be partitioned into two subsets with equal sum?
Reference: [6], problem SP12.
Comment: Weakly NP-hard.

▶ **Problem 5** (Partition-$n/2$).
Instance: A multiset of positive integers $A = a_1, a_2, \dots, a_n$.
Question: Can A be partitioned into two subsets with equal size $\frac{n}{2}$ and equal sum?
Reference: [6], problem SP12.
Comment: Weakly NP-hard.

▶ **Problem 6** (ProductPartition).
Instance: A multiset of positive integers $A = a_1, a_2, \ldots, a_n$.
Question: Can A be partitioned into two subsets with equal product?
Reference: [11].
Comment: Strongly NP-hard.

▶ **Problem 7** (ProductPartition-$n/2$).
Instance: A multiset of positive integers $A = a_1, a_2, \ldots, a_n$.
Question: Can A be partitioned into two subsets with equal size $\frac{n}{2}$ and equal product?
Comment: Strongly NP-hard. See Theorem 13.

▶ **Problem 8** (SquareProductPartition).
Instance: A multiset of square numbers $A = a_1, a_2, \ldots, a_n$.
Question: Can A be partitioned into two subsets with equal product?
Comment: Strongly NP-hard. See Theorem 14.

▶ **Problem 9** (SquareProductPartition-$n/2$).
Instance: A multiset of square numbers $A = a_1, a_2, \ldots, a_n$.
Question: Can A be partitioned into two subsets with equal size $\frac{n}{2}$ and equal product?
Comment: Strongly NP-hard. See Theorem 14.

▶ **Problem 10** (SetProductPartitionBound-K).
Instance: A set (without repetition) of positive integers $A = a_1, a_2, \ldots, a_n$ where $a_i > K$ and all prime factors of a_i are also greater than K. K is fixed and the prime factors are not specified in the instance.
Question: Can A be partitioned into two subsets with equal product?
Reference: [11].
Comment: Strongly NP-hard by a modification of the proof for ProductPartition in [11]. The reduction constructs a set of positive integers A where all elements are unique, which we modify by choosing primes factors $> K$ when constructing A.

▶ **Problem 11** (3-Partition-3).
Instance: A multiset of positive integers $A = a_1, a_2, \ldots, a_n$, with n a multiple of 3.
Question: Can A be partitioned into $n/3$ subsets with equal sum, where all subsets have size 3?
Reference: [6], problem SP15.
Comment: Strongly NP-hard, even when all subsets are required to have size 3 (3-Partition3).

▶ **Theorem 13.** *ProductPartition-$n/2$ is strongly NP-complete.*

Proof. We can reduce from ProductPartition to ProductPartition-$n/2$. Given instance of ProductPartition $\{a_1, \cdots, a_n\}_i$ with n elements, where n is even, we construct an corresponding instance of ProductPartition-$n/2$ as $\{a_1, \cdots, a_n\} \cup \{1\} * n$, where $\{1\} * n$ denotes n instances of the integer 1.

Clearly if we have a valid solution to ProductPartition-$n/2$, we have a valid solution to the instance of ProductPartition. Conversely, given a valid solution to ProductPartition, two subsets $S_1, S_2 \subseteq \{a_i\}_i$ with equal product, the difference between the sizes of S_1 and S_2 is at most $n-2$. One can then distribute the 1s as needed to even the out the number of elements of S_1 and S_2. We can then construct two sets: $S_1 \cup \{1\} * |S_2|$, $S_2 \cup \{1\} * |S_1|$ which form a solution to ProductPartition-$n/2$. Strong NP-hardness follows from strong NP-hardness of ProductPartition-$n/2$. ◀

▶ **Theorem 14.** *SquareProductPartition and SquareProductPartition-$n/2$ is strongly NP-complete.*

Proof. One can reduce from ProductPartition to SquareProductPartition by simply taking an instance $I = \{a_i\}_{i\in\alpha}$ and producing the instance $I' = \{a_i^2\}_{i\in\alpha}$. Given a partition of $\alpha = \alpha_1 \sqcup \alpha_2$ such that $\prod_{i\in\alpha_1} a_i = \prod_{i\in\alpha_2} a_i$, the same partition of α will produce a valid partition of I' as the squares will remain equal. The converse also holds by taking noting that $\prod_{i\in\alpha'} a_i = \sqrt{\prod_{i\in\alpha'} a_i^2}$. The same construction above, with the added requirement that $|\alpha_1| = |\alpha_2|$, will reduce from ProductPartition-$n/2$ to SquareProductPartition-$n/2$. Strong NP-hardness of both holds by noting that squaring integers scales their bitsize by a factor of 2. ◀