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.

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

Keywords and phrases Hardness, algebraic complexity, expression trees

Related Version A full version of the paper is available on arXiv.

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 \( x^n \) mod \( N \). 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** ($(\mathbb{L}, \text{ops})$-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 $\text{ops}$ 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 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](image.png) An example expression tree for $7 \times 11 + (4 \div (3 - 1)) = 79$. The numbers above the internal nodes indicate their values.

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

---

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.
Problem 2 (\((L, \text{ops})\)-AEC-EL / Enforced Leaves).

**Instance:** A target value \(t \in L\) and a multiset of values \(A = \{a_1, \ldots, a_n\} \subseteq L\) 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 \(\text{ops}\), and the leaves of the tree are the list \(A\) in order \(D\), where the tree evaluates to \(t\)?

Problem 3 (\((L, \text{ops})\)-AEC-EO / Enforced Operations).

**Instance:** A multiset of values \(A = \{a_1, a_2, \ldots, a_n\} \subseteq L\), a target \(t \in L\), and an expression tree \(D\) with internal nodes each containing an operation from \(\text{ops}\) 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 \(A = \{a_1, a_2, \ldots, a_n\}\) 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 \(L = \mathbb{N}\) 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>
<thead>
<tr>
<th>Operations</th>
<th>Standard</th>
<th>Enforced Operations</th>
<th>Enforced Leaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>{+}</td>
<td>(\in P)</td>
<td>(\in P)</td>
<td>(\in P)</td>
</tr>
<tr>
<td>{-}</td>
<td>weakly NP-complete</td>
<td>weakly NP-complete</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{\times}</td>
<td>(\in P)</td>
<td>(\in P)</td>
<td>(\in P)</td>
</tr>
<tr>
<td>{\div}</td>
<td>strongly NP-complete</td>
<td>strongly NP-complete</td>
<td>strongly NP-complete</td>
</tr>
<tr>
<td>{+, -}</td>
<td>weakly NP-complete</td>
<td>weakly NP-complete</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{+, \times}</td>
<td>weakly NP-complete (§3)</td>
<td>weakly NP-complete (^a) (§5)</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{+, \div}</td>
<td>weakly NP-complete</td>
<td>strongly NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td>{-, \times}</td>
<td>weakly NP-complete</td>
<td>strongly NP-complete</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{-, \div}</td>
<td>weakly NP-complete</td>
<td>strongly NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td>{\times, \div}</td>
<td>strongly NP-complete</td>
<td>strongly NP-complete</td>
<td>strongly NP-complete</td>
</tr>
<tr>
<td>{+, -}</td>
<td>weakly NP-complete (§3)</td>
<td>strongly NP-complete</td>
<td>weakly NP-complete (§4)</td>
</tr>
<tr>
<td>{+, \times}</td>
<td>weakly NP-complete (§3)</td>
<td>strongly NP-complete</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{+, \div}</td>
<td>weakly NP-complete</td>
<td>strongly NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td>{-, \times}</td>
<td>weakly NP-complete (§3)</td>
<td>strongly NP-complete</td>
<td>weakly NP-complete</td>
</tr>
<tr>
<td>{-, \div}</td>
<td>weakly NP-complete</td>
<td>strongly NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td>{\times, \div}</td>
<td>weakly NP-complete (§3)</td>
<td>strongly NP-complete</td>
<td>Open</td>
</tr>
</tbody>
</table>

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.

\(^a\) 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 \( \text{var} \in \{\text{STD}, \text{EO}, \text{EL}\} \), we obtain a simple reduction
\[
(\mathbb{N}[x_1, \ldots, x_k], \text{ops})\text{-AEC-var} \rightarrow (\mathbb{N}[x_1, \ldots, x_{k-1}], \text{ops})\text{-AEC-var}
\]
Because this reduction preserves algebraic properties, it yields interesting positive results in addition to hardness results. For example, this result demonstrates that \((\mathbb{N}[x_1, \ldots, x_k], \{+,-\})\text{-AEC-Std}\) has a pseudopolynomial-time algorithm via a chain of reductions to \((\mathbb{N}, \{+,-\})\text{-AEC-Std}\) which is equivalent to the classic \textsc{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(·)” 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, \( Q(x_1, \ldots, x_k) \), 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.

\textbf{Theorem 1.} For all \( \text{ops} \subseteq \{+,-,\times,\div\} \), for all variants \( \text{var} \), for all integers \( k > 0 \), there exists an efficient algorithm \( A \) mapping instances \( I \) to positive integers \( A(I) \) such that a polynomial-time reduction
\[
(Q(x_1, \ldots, x_k), \text{ops})\text{-AEC-var} \rightarrow (Q(x_1, \ldots, x_{k-1}), \text{ops})\text{-AEC-var}
\]
is given by substituting \( x_k = B \) in an instance \( I \) for any \( B \in \mathbb{N} \) satisfying \( B \geq A(I) \).

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 \( I = (A,t = f_1/y_1, D) \) be a \((Q(x_1, \ldots, x_k), \text{ops})\text{-AEC-var} \) instance. Loosely, we consider \( B \) to be \((I, \text{ops}, \text{var})\)-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 \( \text{ops} \) and \( \text{var} \) 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:

\[
\begin{align*}
(f, g) + (a, b) &= (fb + ga, gb) \\
(f, g) - (a, b) &= (fb - ga, gb) \\
(f, g) \times (a, b) &= (fa, gb) \\
(f, g) \div (a, b) &= (fb, ga) \\
(f, g) \sim (a, b) &\iff fb = ga
\end{align*}
\]

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 \((f_i, g_i) \leftrightarrow f_i/g_i\) 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 its nonpaired variant is solvable. We now rigorously define sufficiency
in Definition 2 and characterize its use in Lemma 3.

\textbf{Definition 2.} Let \(B\) be a positive integer, and \(I = (A, t = f_t/g_t, D)\) be an instance of
\((\mathbb{Q}(x_1, \ldots, x_k), \text{ops})\)-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 \(f/g\) and \(f_t\) are all less than \(B/2\). Then \(B\) is
\((I, \text{ops}, \text{var})\)-sufficient.

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

\[
E(\{a_i(x_1, \ldots, x_k)\}_{a_i \in A}) = t(x_1, \ldots, x_k)
\]
\[
\Leftrightarrow E(\{a_i(x_1, \ldots, x_{k-1}, B)\}_{a_i \in A}) = t(x_1, \ldots, x_{k-1}, B).
\]

\(^2\) Note that this representation is not unique!
41:6 Arithmetic Expression Construction

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 $A$.

Let $m(f) := \left( \frac{\deg(f) + k}{\deg(f)} \right)$ where $m(f)$ is the maximum number of terms a $k$-variable polynomial $f$ of degree $\deg(f)$ can have. Let $\text{maxcoeff}(f)$ denote the max of all of the norms of coefficients of $f$. That is, $\text{maxcoeff}(f) = \max_{c} \{|c| : c \text{ coefficient of } f\}$.

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

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

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

is sufficient for $I$, where $n = |A|$, $q := \max_{f_i/g_i \in A}\{\text{maxcoeff}(f_i), \text{maxcoeff}(g_i)\}$ 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 $(\mathbb{Q}(x_i), \text{ops})$-AEC-var and construct $(I, \text{ops}, \text{var})$-sufficient $B$ such that the composition $P \rightarrow (\mathbb{Q}(x_1, \ldots, x_k), \text{ops})$-AEC-var $\rightarrow (\mathbb{N}, \text{ops})$-AEC-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 $(c^2 x^2)$ is formable by repeated squaring. That is, we can form superpolynomial coefficients 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 $A \subseteq L$, target $t$, $0 < k < n$, can you give a formula of size at most $k$ with values in $A$ which reaches a target $t \in L$?

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 $(L,S)$-$AEC$ is solvable, then for any operation set $S' \supset S$, the same instance will be solvable in $(L,S')$-$AEC$. This allows us to bundle reductions to several $AEC$ cases simultaneously by giving a reduction $R$ from some partition problem $P$ to $(L,S)$-$AEC$ and proving that if any constructed instance is solvable in $(L,S')$-$AEC$, the partition instance is also solvable. That is, we have the following implications:

$$P\text{-instance } x \text{ Solvable} \implies R(x) \text{ is } S\text{-Solvable}$$

$$R(x) \text{ is } S'\text{-Solvable}$$

\[ \text{Theorem 5.} \] Standard $\{+,\times\} \subseteq S \subseteq \{+,-,\times,\div\}$ is weakly NP-hard by reduction from \texttt{SquareProductPartition-n/2}.

We will spend the remainder of this section proving this theorem. We will reduce from \texttt{SquareProductPartition-n/2} (defined in Appendix A) to $(\mathbb{Z}[x,y,z],S)$-$AEC$. On an instance $\{a_1,\ldots,a_n\}$ with all $a_i \geq 2$,\footnote{We can assume this property with loss of generality by replacing all $a_i$ with $2a_i$.} of \texttt{SquareProductPartition-n/2} construct the following:
Let
\[ B_y = y - x^{n/2} \sqrt[2]{\prod_{i} a_i}; \quad B_z = z - x^{n/2} \sqrt[2]{\prod_{i} a_i}. \]

We then construct the instance of Arithmetic Expression Construction with input set
\[ A = \{B_y, B_z\} \cup \{a_i\}, \] 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 \( \{+,-,\times,\div\} \subseteq S \). On the partition with equalized products, partition the \( a_i \) terms into corresponding sets and take their products to get two polynomials of value \( x^{n/2} \sqrt[2]{\prod_{i} a_i} \). Then form \((B_y + x^{n/2} \sqrt[2]{\prod_{i} a_i})(B_z + x^{n/2} \sqrt[2]{\prod_{i} a_i}) = yz\).

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 \( ev(T) \) to refer to the evaluation of the subtree rooted at node \( T \).

Before stating Theorem 8, we first introduce the concept of \( Q(x) \)-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 \( c_1d_2 - c_2d_1 \neq 0 \),
\[ L_2 = \frac{c_1L_1 + d_1}{c_2L_1 + d_2}. \]

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

We now state our structure theorem:

**Theorem 8.** For any \( S \subseteq \{+,-,\times,\div\} \), let \( I \) be a solvable \((Q(x,y,z),S)\)-AEC-STD instance with entries of the form \( \{B_y, B_z\} \cup \{r_i(x)\} \), where \( B_y \sim_Q(x) y, B_z \sim_Q(z) z, \) and \( r_i \in Q[x] \) and target \( t \) with \( t \sim_Q(x) yz \). 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 \( ev(N) = (iyz)^\pm 1, i \in Q(x) \). For \( T_y, T_z \) the children of \( N \) containing \( B_y, B_z \) respectively, \( ev(T_y) = (ay)^\pm 1, ev(T_z) = (a'z)^\pm 1 \), where \( a, a' \in Q(x) \).
We conclude that exactly one of
would have
is a unique factorization domain.

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
\[
\text{ev}(N) = \frac{eyz + f}{gyz + h}, \quad eh - gf \neq 0, e, f, g, h \in \mathbb{Q}(x)
\]
since \(\text{ev}(N)\) is combined with a sequence of operations with elements in \(\mathbb{Q}(x)\) to form \(t\). That is, \(\mathbb{Q}(x)\) is 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
\[
\text{ev}(T_y) = \frac{ay + b}{cy + d}, \quad \text{ev}(T_z) = \frac{a'z + b'}{c'z + d'}, \quad \text{ev}(N) = \frac{eyz + f}{gyz + h},
\]
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 \(\text{ev}(N) \overset{\sim}{\not\sim} \mathbb{Q}(x)\) \(yz\).

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:
\[
(ay + b)(a'z + b') = lyz(cy + d)(c'z + d') \quad (1)
\]
\[
lyz(ay + b)(a'z + b') = (cy + d)(c'z + d') \quad (2)
\]
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 +:

\[
\frac{ay + b}{cy + d} + \frac{a'y + b'}{c'y + d'} = \frac{eyz + f}{gyz + h}
\]

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

Suppose \( g = 0 \). Then we must have \( e \neq 0 \) to maintain \( ev(N) \sim Q(x) yz \). 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 \((ac' + a'h)c = 0 \) 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 \( ev(N) \sim Q(x) yz \). The LHS of the equation is divisible by \( yz \). Thus \( yz | (cy + d)(cz + d')(eyz + f) \) 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: \( 0 \neq c'f = g(bd' + b'd) = 0 \). 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 \( S \in \{ \text{leaves}(T_y) \cap \{ a_i x \}, \text{leaves}(T_z) \cap \{ a_i x \} \} \), either

1. \( S \) contains \( < n/2 \) leaves \( a_i x \).
2. \( S \) contains \( n/2 \) leaves \( a_i x \) with product \( ax^\alpha \) with \( \alpha < \sqrt{\prod a_i} \).

WLOG let this set be \( \text{leaves}(T_y) \cap \{ a_i x \} \). 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.

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

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

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

\[ \triangleright \text{Claim 10.} \quad \text{If } T_y \text{ contains } n/2 \text{ leaves } a_i x \text{ with } \prod a_i = a < \sqrt[n]{\prod a_i} \text{ and } y' = y - x^{n/2} \sqrt[2]{\prod a_i}, \text{ then } ev(T_y) \text{ is not of the form } (ay)^\pm 1 \text{ for any } a \in Q(x). \]
Proof. First, we rewrite our target ev(T_y) in terms of y', yielding 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 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 a. Every coefficient of 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 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\textsuperscript{5} based on instance I with

\[ x = B_1 = A(I), y = B_2 = A(I(B_1)), z = A(I(B_1, B_2)). \]

However, since the reduction is of the form \[ \{ y - ax^{n/2}, z - ax^{n/2} \} \cup \{a, x\}, \]
if we replace B_2 with \( B_2' = \max(B_2, 1 + \alpha B_1^{n/2}) \), and B_3 with \( B_3' = \max(A(I(B_1, B_2'))), 1 + \alpha(A((B_1, B_2')))^{n/2} \) 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.

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

\textsuperscript{5} Note that this denotes replacing with \( B_1 \) 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 \( B_3 \).
Proof sketch. This statement is proved via reduction from \textsc{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 \( b_i + c_i = (L/a_i)p_iy \) and \( b_i - c_i = (L/a_i)p_iy \). We set our target polynomial as \( t(x, y) = L^nx^{n-1}y^n \prod_{i \in [n]} p_i \) and we enforce the following order of leaves:

\[
\begin{align*}
&b_1, c_1, x, b_2, c_2, x, \ldots, x, b_n, c_n
\end{align*}
\]

If an instance of this product partition variant is solvable, then the constructed instance evaluates to \( t(x, y) = L^nx^{n-1}y^n \prod_{i \in [n]} p_i \) 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.

\begin{claim}
\( (\mathbb{N}, \{+, \times\})\)-AEC-EO is weakly NP-hard.
\end{claim}

\textbf{Proof.} This proof proceeds by reduction from \textsc{3-Partition-3}, which is \textsc{3-Partition} with the extra restriction that all the subsets have size 3. Given an instance of \textsc{3-Partition-3}, \( A = \{a_1, a_2, \ldots, a_n\} \), 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 \textsc{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 \( i \)th \( (\square + \square + \square) \) 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. \hfill \(\checkmark\)
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 $A = a_1, a_2, \ldots, a_n$.
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 $A = a_1, a_2, \ldots, a_n$.
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?
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?

▶ 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?

▶ 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?

▶ 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?
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?
Theorem 13. \textsc{ProductPartition}-n/2 is strongly NP-complete.

\textbf{Proof.} We can reduce from \textsc{ProductPartition} to \textsc{ProductPartition}-n/2. Given instance of \textsc{ProductPartition} \(\{a_1, \ldots, a_n\}\), with \(n\) elements, where \(n\) is even, we construct an corresponding instance of \textsc{ProductPartition}-n/2 as \(\{a_1, \ldots, a_n\} \cup \{1\} \ast n\), where \(\{1\} \ast n\) denotes \(n\) instances of the integer 1.

Clearly if we have a valid solution to \textsc{ProductPartition}-n/2, we have a valid solution to the instance of \textsc{ProductPartition}. Conversely, given a valid solution to \textsc{ProductPartition}, two subsets \(S_1, S_2 \subseteq \{a_i\}\), 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 out the number of elements of \(S_1\) and \(S_2\). We can then construct two sets: \(S_1 \cup \{1\} \ast |S_2|\), \(S_2 \cup \{1\} \ast |S_1|\) which form a solution to \textsc{ProductPartition}-n/2. Strong NP-hardness follows from strong NP-hardness of \textsc{ProductPartition}-n/2. \(\blacksquare\)

Theorem 14. \textsc{SquareProductPartition} and \textsc{SquareProductPartition}-n/2 is strongly NP-complete.

\textbf{Proof.} One can reduce from \textsc{ProductPartition} to \textsc{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 \(\alpha = \alpha_1 \cup \alpha_2\) such that \(\prod_{i \in \alpha_1} a_i = \prod_{i \in \alpha_2} a_i\), the same partition of \(\alpha\) will produce a valid partition of \(I'\) as the squares will remain equal. The converse also holds by taking noting that \(\prod_{i \in \alpha_1} a_i = \sqrt{\prod_{i \in \alpha_2} a_i^2}\). The same construction above, with the added requirement that \(|\alpha_1| = |\alpha_2|\), will reduce from \textsc{ProductPartition}-n/2 to \textsc{SquareProductPartition}-n/2. Strong NP-hardness of both holds by noting that squaring integers scales their bitsize by a factor of 2. \(\blacksquare\)

References


