DECISION PROCEDURES FOR ELEMENTARY FRAGMENTS OF SEMËNOV ARITHMETIC

Hera Brown

Submitted for the MCompSciPhil Computer Science and Philosophy



Trinity Term 2026

Oxford, United Kingdom

9000 words

Abstract

We prove elementary upper bounds on the decision problems for two novel fragments of Semënov arithmetic. While Semënov arithmetic, the first-order theory of the integers with addition and exponentiation, is a non-elementary theory—any decision procedure for Semënov arithmetic has time complexity unbounded by any tower of exponentials—it has elementary fragments. Previous work has shown that interesting elementary fragments of Semënov arithmetic exist, and provides algorithms for these fragments. Here we extend these ideas to more general fragments of Semënov arithmetic.

Our main contributions are elementary algorithms for the *simple Semënov* and the *fixed-stratification fragments* of Semënov arithmetic, and complexity analyses for these algorithms. We also concisely give solutions of certain Semënov-definable exponential equations, and show that there are non-elementary unary-predicate reducts of Semënov arithmetic extending Presburger arithmetic.

Acknowledgements

It takes a village to write a thesis, and here I'd like to thank some of that village's residents.

- My supervisor, Michael Benedikt, and my informal supervisor, Alessio Mansutti, who both introduced me to the problems covered in this thesis, guided me throughout the process, and introduced me to so much interesting theoretical computer science,
- The academic admin team, for always being on hand to answer my questions (even when asked quite close to the deadline!),
- Christoph Haase, for all the support and teaching over the years, making theoretical computer science so fun, and helping me become the logician I am today,
- Leo and Yoshi, for always being there to talk about our problems (academic or otherwise), giving me the LaTeX code this thesis grew out of, and our many lunches and dinners at Fusion House and Donburi Inn,
- Nich, for talking about our projects over coffee, getting me to put what I was doing into words that other people can actually understand, setting my intuitions straight about what it means to be non-elementary, and reviewing the first draft of my thesis,
- Yusuf, for being a very helpful rubber duck for getting through technical snags, and for helping me out of a number-theoretic tangle in Proposition 4,
- Josephine, for the coffees and pointing out everywhere I was being infelicitous, and
- Ash and Barney, for reminding me that there is (far) more to computer science than technical detail and abstract nonsense.

Finally I would like to thank my Mum, where without her support for the last twenty-one years I wouldn't be where I am today. Thank you for raising me to be a computer scientist.

Contents

1 Introduction **1**
1.1 Contributions 2

2 Background **3**
2.1 A brief history of arithmetic theories 3
2.1.1 Presburger arithmetic 3
2.1.2 Semënov arithmetic 4
2.2 Technical preliminaries 6

3 Fixed-stratum Semënov **9**
3.1 A definition of the stratified fragment 9
3.1.1 Well-behaved fragments of stratified Semënov arithmetic 11
3.2 The destratification algorithm 11
3.2.1 Correctness of the destratification algorithm 17
3.2.2 Complexity of the destratification algorithm 19

4 Simple Semënov and the iterated powers fragment **23**
4.1 A definition of the simple Semënov fragment 23
4.1.1 Interesting sub-fragments of simple Semënov 24
4.2 The complexity of Simple Semënov 26
4.3 A note on the general stratified fragment 30

5 Other results **31**
5.1 Presburger-definable exponential equations 31
5.2 Non-elementary unary reducts of Semënov arithmetic 33

6 Conclusion **35**
6.1 Future work and limitations 35

Chapter 1

Introduction

Formal mathematics is nature's way of letting you know how sloppy your mathematics is.
– Leslie Lamport

In this project we consider the decision problems of arithmetic theories. That is, we are interested in deciding whether a sentence of a given first-order formalism over the integers is true or not. Arithmetic theories are useful throughout computer science: theoretically, they're interesting logical and model-theoretic objects in their own right, and practically, they are useful in both verifying software systems and automated theorem proving.

Here we consider the complexity of arithmetic theories, as their complexity gives us insight into the expressive power of that theory: the higher the complexity, the higher the expressive power. Many arithmetic theories, such as Peano arithmetic, are powerful enough to formalise most of mathematics. Unfortunately these theories are undecidable [Gö31, Chu36]. But even decidable arithmetic theories often have such high complexities that implementing decision procedures is impractical. When studying these theories, we focus on their properties as mathematical objects rather than their practical uses.

Semënov arithmetic—the first-order theory of addition and powers-of-two over the integers—is decidable with such high complexity. With it we express statements such as:

- that the difference of two powers is a power *iff* they're adjacent,
$$\forall x \forall y \ (\exists z \ (2^x - 2^y = 2^z) \leftrightarrow x = y + 1)$$
- that the function $2^{x+y} + x$ is injective,
$$\forall x \forall y \forall z \forall w \ ((2^{x+y} + x = 2^{z+w} + z) \rightarrow (x = z \wedge y = w))$$
- and that the variable x is a power of four.
$$\exists y \ (x = 2^y \wedge \exists z \ (z + z = y))$$

Semënov arithmetic has *non-elementary* complexity: this means that any algorithm for Semënov arithmetic has runtime growing faster than all functions in the set $\left\{2^n, 2^{2^n}, 2^{2^{2^n}}, \dots\right\}$. *Presburger arithmetic*, on the other hand—the first-order theory of addition over the integers—has elementary complexity: still high enough to preclude large-scale implementation, but less complex than Semënov arithmetic. That leads to the question: how much do we have to restrict Semënov arithmetic to get an elementary theory?

Why study arithmetic theories? Firstly, there is a model-theoretic tradition of studying arithmetic theories. Understanding their complexity tells us about how complicated their models are and how powerful their language is, and these are of interest to both model theorists and those studying models of computation.

Secondly, despite decision procedures for full arithmetic theories being too high for practical implementation, the methods used in proving this complexity often work just as well on restricted fragments where complexity is much lower. For instance, [CMS24] applies quantifier-elimination techniques, as we do in this thesis, to show that *integer-linear-exponential programming* is NP-complete. Developing these techniques therefore has implications for further work in algorithms and complexity.

1.1 Contributions

In this project we look for elementary fragments of Semënov arithmetic extending Presburger arithmetic. Benedikt, Chistikov, and Mansutti [BCM23] showed that such fragments exist, and here we give more of these fragments. Our contributions are as follows:

- we introduce the *stratified Semënov arithmetic fragment*, a restriction of Semënov arithmetic that bans each variable from appearing in too many different places (3.1),
- we give an elementary algorithm for *fixed-stratum Semënov arithmetic*, a parameterised restriction of stratified Semënov arithmetic (3.2),
- we give an elementary algorithm for the *simple Semënov fragment*, an extension of the *predicate fragment* presented in [BCM23] (4.2),
- we give solutions to Semënov-definable equations in Presburger arithmetic (5.1),
- and we show that there are unary-predicate reducts of Semënov arithmetic with non-elementary complexity (5.2).

Chapter 2

Background

...there are many beautiful areas of theoretical computer science that I could not cover for lack of time and space. – Dexter Kozen

Here we cover both the history of related work (2.1) and technical preliminaries (2.2).

2.1 A brief history of arithmetic theories

An arithmetic theory is a set of first-order sentences, closed under logical entailment, containing all true statements about a given model. For instance, *Presburger arithmetic* is the set of true sentences holding over the model $\mathbb{Z}$, with the usual interpretations of the symbols $<$ and $+$. The decision problem for an arithmetic theory T asks: given a first-order formula φ in the signature of T, is φ in the set T?

2.1.1 Presburger arithmetic

The first arithmetic theory to be shown decidable was Presburger arithmetic in 1929 [Pre29] (translated in [Sta84]). Presburger gave a *quantifier elimination procedure* for Presburger arithmetic. Given an input formula φ, a quantifier elimination procedure returns an equivalent quantifier-free formula φ'. Assuming that the quantifier-free formulae of the theory are easy to decide—in Presburger arithmetic they are—and assuming that transforming φ into φ' is itself decidable, this proves an arithmetic theory is decidable.

Presburger's algorithm works as follows. We assume that all quantifiers in the input formula are at the start of the formula, and negation only appears in front of atomic formulae. With this, we repeatedly eliminate the innermost existential quantifier of the formula. For each innermost quantifier, the quantifier-free part of the formula is converted into disjunctive normal form (DNF), the existential quantifier is pushed through to each

disjunct, and the conjuncts of each disjunct are processed on a case-by-case basis. Once a quantifier-free formula is reached each atomic formula is evaluated and the formula decided.

However, Presburger's algorithm isn't elementary. Each eliminated quantifier requires putting the quantifier-free part of φ in DNF, and each conversion to DNF introduces an exponential blowup in the size of the formula. Since the number of quantifiers in a formula is unbounded, so is the height of the exponential in the size blowup of the formula.

Despite this, Presburger arithmetic is an elementary theory. In the 1970s, work began on the complexity of arithmetic theories, particular Presburger arithmetic. Cooper's algorithm for Presburger arithmetic [Coo72], introduced in 1972, is elementary and triply-exponential [Opp78], by avoiding the repeated conversion to DNF in Presburger's original algorithm.

Work on the complexity of Presburger arithmetic continued: Oppen [Opp73] gave a $2^{2^{2^{cn \log n}}}$ upper bound on deterministic time and space complexity, Ferrante & Rackoff [FR75] gave a deterministic doubly-exponential space upper bound, Reddy & Loveland [RL78] gave a singly-exponential space and doubly-exponential time upper bounds for bounded-quantifier-depth Presburger arithmetic, Berman [Ber80] showed $\mathrm{STA}(*, 2^{2^{cn}}, n)$-completeness (where the first coordinate means no limit on space, the second means a $2^{2^{cn}}$ bound on time, and the third means an n bound on alternations), and Fürer [Fü82] gave doubly-exponential time lower bounds for bounded quantifier-depth Presburger arithmetic.

For lower bounds, both Fischer & Rabin and Heintz [FR74, Hei76] gave deterministic doubly-exponential lower bounds on the time complexity of Presburger arithmetic. Weispfenning [Wei90] gave the finest bound on runtime where, for an input formula φ with a quantifier blocks each of length at most b, a quantifier elimination procedure is given with time and space complexity bounded by $2^{c \cdot |\varphi|^{(4b)^a}}$. We use Weispfenning's algorithm later (Algorithm 2) essentially as it is presented in [BCM23].

2.1.2 Semënov arithmetic

Around the same time as the work on the complexity of Presburger arithmetic, Semënov worked on the complexity of two fragments central to our thesis, defined as follows.

Definition 1. *Semënov arithmetic* is the extension of Presburger arithmetic by a powers-of-two function $x \mapsto 2^x$, defined in [Sem84]. The *predicate fragment* is the extension of

Presburger arithmetic by a predicate $P(x)$ for the powers of two, defined in [Sem80].

Note that in [Sem80] more general extensions of Presburger arithmetic predicates, like the set of Fibonacci numbers or the image of the factorial function, are considered. In [Sem84], similarly, other non-exponential functions are considered. But here we only concern ourselves with the powers-of-two predicate and function.

Example 1. In Semënov arithmetic, we can define the set of powers of four:

$$\varphi(x) \stackrel{\text{def}}{=} \exists y\ (x = 2^y \wedge \exists z\ (z + z = y)).$$

In the predicate fragment, we can define pairs $\langle x, y\rangle$ where x and y sum to a power of two:

$$\psi(x, y) \stackrel{\text{def}}{=} \exists z\ x + y = z \wedge P(z).$$

Semënov proved both Semënov arithmetic and the predicate fragment decidable, using quantifier elimination. His method in both cases has non-elementary complexity.

Unfortunately there is no hope of an elementary algorithm for Semënov arithmetic, because the *pairing function* $f(x, y) = 2^{x+y} + x$—an injection from $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$—is definable. Ferrante and Rackoff [FR79] show that all first-order theories of a pairing function are non-elementary, by an encoding of Turing machines with non-elementary runtimes.

On the other hand, we do have an elementary algorithm for the predicate fragment [BCM23]. The predicate fragment is strictly weaker than general Semënov arithmetic, because even if we know x is a power of two, we do not know *which* power of 2; this lost information gets us a weaker theory. Because of this we have an elementary algorithm.

The algorithm in [BCM23, Theorem 2] for the predicate fragment works in two stages:

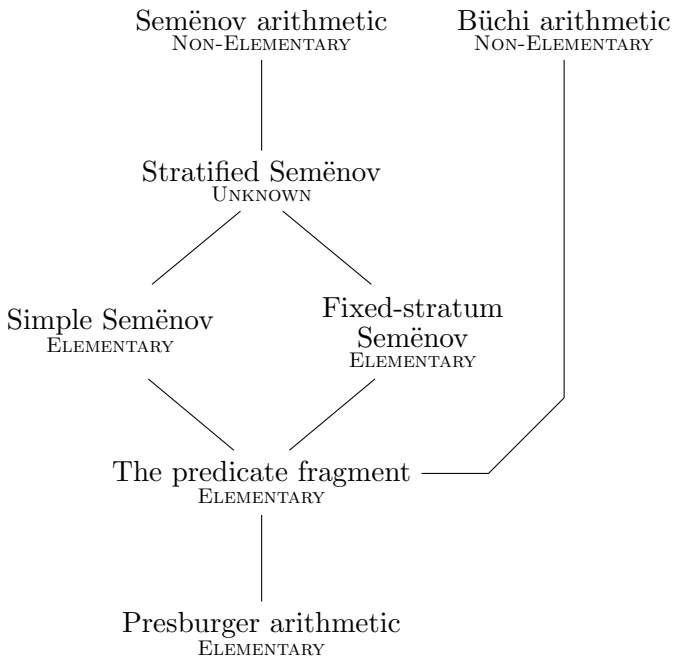
- in the first stage, the input formula is put into a normal form where all variables appear only as comparisons between two exponentiated variables, and
- in the second stage, the normal-form formula is reduced to a Presburger sentence, then dealt with using Weispfenning's Presburger quantifier-elimination algorithm.

In this project we extend techniques in [BCM23] to extensions of the predicate fragment; these extensions are the *simple* fragment and the *fixed-stratification* fragment. We provide the first elementary decision procedures for these fragments.

Büchi arithmetic. A natural extension of Presburger arithmetic is *Büchi arithmetic* (essentially from [Bü60] and re-formulated in [Bru85]). Büchi arithmetic extends Presburger arithmetic with a binary predicate $V_2(x, y)$ that holds *iff* y is the greatest power of 2 dividing x (this definition is given in [Bè02]).

In Büchi arithmetic the predicate fragment can be defined as $P(x) \stackrel{\text{def}}{=} V_2(x, x)$, and like Semënov arithmetic it is a non-elementary theory (see e.g. [HS26]). Interestingly, Büchi arithmetic is incomparable with Semënov arithmetic in expressive power, and the union of the two theories is undecidable [CP86].

The following picture lays out the complexity landscape. Theories higher up contain the theories below. *Simple Semënov*, *fixed-stratum Semënov*, and the *stratified fragment*, as well as their complexities, are novel fragments introduced in this thesis.



2.2 Technical preliminaries

Notation and language. We work in the signature $(\mathbb{Z}; 0, 1, <, +, (q \mid x)_{q \in \mathbb{N}^+}, x \mapsto 2^x)$. Constant multiplication is not in our signature, but we write $a \cdot x$ as shorthand for $x + x + \cdots + x$ added to itself a times. Likewise, constant division is not in our signature, but we write $(\frac{t_1}{a}) < t_2$ as shorthand for $t_1 < a \cdot t_2$. When writing exponentiation explicitly, we omit the absolute value brackets for the sake of readability; so we write 2^x for $2^{|x|}$.

Divisibility constraints $(q \mid x)$ are in our signature, but note that these are definable in Presburger arithmetic by the following translation:

$$(q|x) \iff \exists y\ q \cdot y = x$$

so using these do not affect the expressive power of our theories.

We exclusively denote division using stacked notation (like $\frac{a}{b}$), and exclusively denote term substitution using slashes (so we write $(x + y = z)[a/x]$ for $a + y = z$).

We use $\top$ as shorthand for $0 = 0$ and $\bot$ as shorthand for $0 \neq 0$.

Formula normalisation. We assume the following normalisations of terms and formulae. We do these normalisations implicitly, and all are syntactic transformations done in linear time. The following transformations do not factor into our complexity analyses.

- We put formulae in prenex normal form; that is the following form:

  $$Q_1 x_1 Q_2 x_2 \dots Q_n x_n\ \varphi$$

  where each Q_i is either an $\forall$ or an $\exists$, and where φ is quantifier-free.
- We put all inequalities in the form $t_1 - t_2 < 0$, sometimes writing this as $t_1 < t_2$. Recall that we can express subtraction in Presburger arithmetic with only linearly many existential quantifiers, as below:

  $$t_1 - t_2 < 0 \iff \exists x\ (t_2 + x = 0 \wedge t_1 + x < 0).$$

- We freely assume commutativity of $+$ and rearrange terms accordingly.
- We evaluate all terms $a + b$ and 2^a where a and b are constant terms.
- We put all formulae in *flat* form: that is, the only exponentiated terms that appear are 2^x for variables x. We do this with the following substitution:

  $$\varphi \mapsto \exists z\ z = 2^x \wedge \varphi[2^z/2^{2^x}] \text{ for each sub-term } 2^{2^x} \text{appearing in } \varphi.$$

  We sometimes write non-flat formulae in this thesis for the sake of intuition.
- We assume that all terms are of the following form:

  $$\sum_{i \in I} a_i \cdot 2^{x_i} + \sum_{j \in J} a_j \cdot x_j + c$$

  where the finite sets of indices I and J may overlap. Because of our assumption of

flatness above, this assumption does not restrict our expressive power.

We say that a term t is *homogeneous* if it has no constant term; that is, a term t is homogeneous *iff* it i of the form $\sum_{i \in I} a_i \cdot 2^{x_i} + \sum_{j \in J} a_j \cdot x_j$. We say a variable appears *linearly* in a term if it doesn't appear exponentiated anywhere in that term.

We also assume that $0 \in \mathbb{N}$, and that $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$.

Parameters and parameter tables. Our complexity analysis involves keeping track of multiple measures of growth in a formula, and how they interact. We call these *parameters*, and define the most relevant parameters below.

- **#hom:** The number of homogeneous terms appearing in the formula,
- **heft:** The maximum number of variables appearing in any single term of the formula,
- **max. coefficient:** The maximum absolute value over all coefficients in the formula,
- **max. constant:** The maximum absolute value over all constants in the formula,
- **# variables:** The number of distinct variables in the formula,
- **mod:** The least common multiple of all q appearing in divisibility constraints $(q \mid t)$ in the formula. When there are no divisibility constraints, we take **mod** to be 1.

In our complexity analyses we extensively use *parameter tables*; these are useful in keeping track of how a formula changes with the application of a function, and we use them to give upper bounds. An example is given below, for a toy function f:

	parameter A	parameter B	parameter C
φ	a	b	c
$f(\varphi)$	a^2	$3ab$	1

The parameter table tells us that, if our input φ has values a, b, and c for parameters A, B, and C, then the output $f(\varphi)$ upper-bounds A, B, and C by a^2, $3ab$, and 1.

Chapter 3

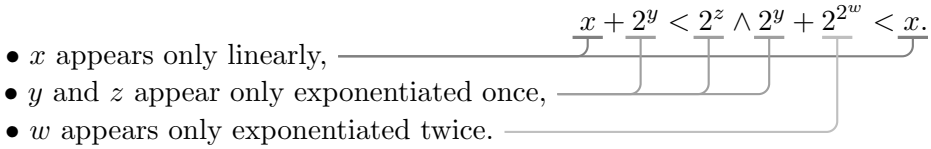
Fixed-stratum Semënov

It's elementary, my dear Watson.
– Sherlock Holmes

In this chapter, we introduce the *stratified* fragment of Semënov arithmetic (3.1), introduce the *destratification algorithm* (3.2), and analyse the algorithm's complexity (3.2.2).

3.1 A definition of the stratified fragment

Let's begin with some intuition for the stratified fragment. In some Semënov formulae, there are variables that only ever appear linearly, some that only ever appear exponentiated once, and so on. Consider the following example:



Above, each variable appears only in one layer, lower down or higher up. Like layers in a rock, we call these *strata*. If all variables each appear only in one stratum, we say that formula is *stratified*. On the other hand, if a variable appears in multiple strata in the same atomic formula, we say the formula isn't stratified. Consider the following example:



The variable x appears in too many strata at once, so $2^{x+y} + x = z$ isn't stratified.

These intuitions about stratified formulae corresponds to a precise formal definition.

Definition 2 (Stratified formulae)**.** A *stratification function* $s : \mathit{terms}(\varphi) \to \mathbb{N}$ for a formula φ is a map from terms in φ to naturals where the following five conditions hold:

- if $s(x) = k$ then $s(2^x) = k - 1$,
- if $s(t) = k$ and c is a constant then $s(t + c) = k$,
- if $s(t_1 + t_2) = k$ then $s(t_1) = s(t_2) = k$,
- for subformulae $t_1 = t_2$ or $t_1 < t_2$ we have $s(t_1) = s(t_2)$, and
- for subformulae $t_1 = 2^{t_2}$ or $t_1 < 2^{t_2}$ we have $s(t_1) = s(t_2) - 1$.

We say a formula is *stratified* just when it has a stratification function. We refer to a set of variables all with the same value under s as a *stratum*.

For instance, the formula $x < 2^y \wedge 2^y < z$ is stratified, with stratification function $s(x) = 0,\ s(y) = 1$, and $s(z) = 0$. On the other hand, the formula $x < 2^x$ isn't stratified; we would need $s(x) = s(x) - 1$ for any candidate stratification function.

Remark 1. Given two stratification functions s_1 and s_2 for a formula φ, say that $s_1 \leq s_2$ *iff* $s_1(x) \leq s_2(x)$ for all variables x in φ. For purely technical reasons, throughout this thesis we assume that all stratification functions are $\leq$-minimal.

Remark 2. Recall from Section 2.2 that we flatten all our formulae with this substitution:

$$\varphi \mapsto \exists z\ z = 2^x \wedge \varphi[2^z/2^{2^x}] \text{ for each sub-term } 2^{2^x} \text{appearing in } \varphi.$$

This preserves stratification functions; that is, if s is a stratification function for φ, then s', defined as $s' \cup \{s(z) = s(x) - 1\}$, is a stratification function for $\exists z\ z = 2^x \wedge \varphi[2^z/2^{2^x}]$.

Definition 3. The *stratified fragment* is the subset of Semënov formulae that are stratified.

Definition 4. The *k-strata fragment* of Semënov arithmetic consists of all stratified formulae that have stratification functions with $\leq k$ strata.

For instance, the formula $\exists x \exists y \exists z\ (x = 2^y \wedge y = 2^z)$ is in the 3-strata fragment, but not the 2-strata fragment, as any stratification function for it must have $s(x) = 0$, $s(y) = 1$, and $s(z) = 2$.

Finally we define *fixed-stratum Semënov*, a family of decision problems, as follows.

Definition 5. The *fixed-stratum Semënov* decision problem contains all k-strata decision problems for each $k \in \mathbb{N}$.

When we talk about the *fixed-stratum Semënov* decision problem as though it were just one decision problem, we mean the following: given a promise that our input formula φ is stratified with at most k strata, for some $k \in \mathbb{N}$, is φ true?

3.1.1 Well-behaved fragments of stratified Semënov arithmetic

Here we go through fragments of stratified Semënov arithmetic with pleasant properties. When in these fragments, we either run specialised algorithms, or run our usual algorithms much faster. The goal of our algorithms is to get into these fragments and then stay in them. The relevant fragments here are as follows:

- **Strat-Sem** (generalising Sem in [BCM23]): All stratum-0 and stratum-1 variables appear either always linearly or always in a power term, and every *bound* variable appears only in comparisons $a \cdot 2^x < c$, with c a constant, or $a \cdot 2^x < b \cdot 2^y$.
- **Strat-PowCmp** (generalising PowCmp in [BCM23]): All stratum-1 variables appear only in comparisons $a \cdot 2^x < c$, with c a constant, or $a \cdot 2^x < b \cdot 2^y$.
- The **thin** fragment: In the thin fragment, at most two variables appear in each atomic formula, and all coefficients of variables are ± 1.

We sometimes abuse notation and say that a variable x *is in Strat-Sem* or *is in Strat-PowCmp*; what we mean by this is that every atomic formula that x appears in is in Strat-Sem or Strat-Powcmp (with bound variables remaining bound).

3.2 The destratification algorithm

We are now ready to give an algorithm for the stratified fragment; this is the destratification algorithm below. Note that this algorithm works on any stratified formula, but we restrict our complexity analyses to the fixed, and later simple, fragments in this thesis.

We also use the functions *PresQE* (Algorithm 2), *Linearise* (Algorithm 3), and *SemCover* (Algorithm 4), all taken with minor modifications from [BCM23].

The Destratification Algorithm. This (Algorithm 1) is the main algorithm for the stratified fragment; intuitively, on each iteration, it runs as follows:

- it eliminates all stratum-0 variables from the input (by calling PresQE on line 7),
- it puts the stratum-1 variables into a normal form (by calling SemCover on line 12),
- then it lowers the stratum of every variable (by calling Linearise on line 16).

Doing this repeatedly gets us a formula with 0 strata, so a formula $\top$ or $\bot$ as required.

Algorithm 1 The Destratification Algorithm.

Input: Restricted stratified sentence φ.
Output: Equivalent sentence $\top$ or $\bot$.

while φ is not quantifier free **do**
  for each $\Pi \leftarrow$ shortest quantifier prefix of φ such that $\varphi = \Pi \cdot \exists \boldsymbol{x} \cdot \psi$, with all stratum-1 variables ψ in Strat-Sem, **do**
    $\Pi' \leftarrow$ the empty string of quantifiers ▷ Π' is a global variable
    $S \leftarrow \{\psi\}$
    for each stratum-0 variable x_0 in $\boldsymbol{x}$ **do**
      for each formula ξ in S **do**
        $S \leftarrow S \cup \text{PresQE}(x_0, \xi) \setminus \{\xi\}$
      end for
    end for
    $\psi \leftarrow \bigvee S$ ▷ Now there are no bound stratum-0 variables in ψ
    for each stratum-1 variable x_1 in $\boldsymbol{x}$ **do**
      $\psi \leftarrow \text{SemCover}(x_1, \psi)$
    end for ▷ Now all bound stratum-1 variables are in Strat-PowCmp
    $\psi \leftarrow \Pi' \cdot \psi$
  end for
  $\varphi \leftarrow \text{Linearise}(\psi)$
end while

Algorithm 2 PresQE.

Input: Variable x and Semënov formula $\varphi(x, \boldsymbol{y})$ where x appears only linearly in φ.
Output: Set of formulae S where for each $\varphi' \in S$ the variable x does not appear, and where $\bigvee S$ is equivalent to φ.

$T \leftarrow \{(a, -t(\boldsymbol{y})) \mid a > 0, a \cdot x + t \in \mathit{lin}(\varphi)\} \cup \{(-a, t(\boldsymbol{y})) \mid a < 0, a \cdot x + t \in \mathit{lin}(\varphi)\} \cup \{(1, 0)\}$
$\Gamma \leftarrow \left\{\varphi[\frac{(t+k)}{a}/x] \wedge (a \mid t + k) \mid (a, t) \in T, k \in [-r, r] \text{ where } r \stackrel{\text{def}}{=} a \cdot \text{mod}(\varphi)\right\}$
return Γ

PresQE. The PresQE algorithm (Algorithm 2) is taken from [BCM23], itself an implementation of Weispfenning's procedure [Wei90]. Any solution to $\varphi(x, \boldsymbol{y})$, where x appears only linearly, can be given in terms of just $\boldsymbol{y}$; we rewrite the equation using divisibility constraints to do this. That gets us an equisatisfiable formula where x does not appear.

Linearise. The Linearise algorithm (Algorithm 3) essentially takes logarithms of formulae in Strat-PowCmp. Importantly, this reduces the stratum of each variable.

SemCover. The SemCover algorithm (Algorithm 4) does a large case analysis to put exponential equations in Strat-Sem; each case takes terms of the form 2^x and considers its size relative to the rest of the term. Careful analysis of each case puts the variable into Strat-Sem, which can then be linearised as desired.

Algorithm 3 Linearise.

Input: Stratified formula φ.

Output: Equivalent formula φ' where, if x appears only in Strat-PowCmp in φ, then x appears only linearly in φ'.

1: **for** each $x \in \mathit{variables}(\varphi)$ where x is in Strat-PowCmp **do**

2: apply the following substitutions to each subformula of φ:

3: $a \cdot 2^x < b \mapsto \begin{cases} x < \left\lceil \log(\frac{b}{a}) \right\rceil & \text{if } a > 0 \text{ and } b > 0 \\ x > \left\lceil \log(\frac{b}{a}) \right\rceil & \text{if } a < 0 \text{ and } b < 0 \end{cases}$

4: $a \cdot 2^x < b \cdot 2^y \mapsto \begin{cases} x < y + \left\lceil \log(\frac{b}{a}) \right\rceil & \text{if } a > 0 \text{ and } b > 0 \\ x > y + \left\lceil \log(\frac{b}{a}) \right\rceil & \text{if } a < 0 \text{ and } b < 0 \end{cases}$

5: $q \mid 2^x - r \mapsto \begin{cases} (q' \mid x - r') \wedge (x \geq r') & \text{if } r' = \min\{s \geq 0 \mid (q \mid 2^s - r)\}, \\ & \quad q' = \min\left\{t > 0 \mid (q \mid r \cdot (2^t - 1))\right\} \\ x = r' & \text{if } r' = \min\{s \geq 0 \mid (q \mid 2^s - r)\}, \\ & \quad \left\{t > 0 \mid (q \mid r \cdot (2^t - 1))\right\} = \emptyset \\ \bot & \text{otherwise.} \end{cases}$

6: **end for**

7: **return** φ

We now define terminology to make talking about the destratification algorithm easier.

Definition 6. An *inner iteration* of the destratification algorithm is one run through the **for**-loop on lines 2–15 (and we call this loop the *inner loop*). An *outer iteration* of the destratification algorithm is one run through the **while**-loop on lines 1–17 (and we call this loop the *outer loop*).

So each inner iteration processes one block of quantifiers, and each outer iteration processes one whole stratum. There are multiple inner iterations in each outer iteration, and there are as many outer iterations as there are strata in the input formula.

First we analyse the complexity of the PresQE and SemCover algorithms above. Note these analyses are essentially the same as in [BCM23], but we include them to guarantee that the minor edits we made do not change the complexity of each algorithm.

Lemma 1. *On input (x, φ), where x appears only linearly in φ, PresQE returns a set $S = \{\psi_i\}$ whose formulae satisfy the parameter table below:*

	#hom	*heft*	*max. coef.*	*max. constant*	*#vars.*
φ	h	k	a	c	v
ψ_i	h	$\min(2k, v)$	$2a^2$	$a^2m + 2ac$	$v - 1$
$\bigvee_{\psi_i \in \Gamma} \psi_i$	$h^2 + h$	$\min(2k, v)$	$2a^2$	$a^2m + 2ac$	$v - 1$

where m is the largest modulus appearing in any divisibility constraint in φ.

Proof. We go over the blowup in each parameter.

Algorithm 4 SemCover.

Input: Variable x and a stratified formula φ, in Strat-Sem, where φ contains 2^x.
Output: Equivalent formula φ' where 2^x appears only in comparisons from Strat-PowCmp.

1: $I \leftarrow$ set of inequalities in φ outside Strat-PowCmp where x appears as a power
2: $H \leftarrow \{(\eta, \sigma) \mid \eta(x) + \sigma(\boldsymbol{z}) + c < 0$ in I, with η and σ homogeneous$\}$
3: $\Gamma \leftarrow \{\varphi\}$
4: **for** $(\eta, \sigma) \in H$ **do**
5: $\quad A \leftarrow$ subset of I with these η and σ
6: $\quad 2^g \leftarrow 2^7 \cdot (\lambda(\|\eta\|_1 + \max\{|c| \mid (\eta + \sigma + c < 0) \in A\}))^2$
7: $\quad a \leftarrow$ coefficient at 2^x in η
8: $\quad \beta \leftarrow 2^x > 2^g$
9: $\quad \Gamma \leftarrow \{2^x = 2^j \wedge \gamma[\alpha[2^j/2^x]/\alpha \mid \alpha \in A]$,
10: $\qquad \beta \wedge \lambda(a) \cdot 2^x < \lambda(\sigma) \wedge \sigma < 0 \wedge \gamma[\top/\alpha \mid \alpha \in A]$,
11: $\qquad \beta \wedge \lambda(a) \cdot 2^x < \lambda(\sigma) \wedge \sigma \geq 0 \wedge \gamma[\bot/\alpha \mid \alpha \in A]$,
12: $\qquad \beta \wedge \lambda(a) \cdot 2^x = \lambda(\sigma) \wedge \gamma[\alpha[\frac{\lambda(\sigma)}{\lambda(a)}/2^x]/\alpha \mid \alpha \in A]$,
13: $\qquad \beta \wedge \lambda(a) \cdot 2^x = 2 \cdot \lambda(\sigma) \wedge \gamma[\alpha[\frac{2 \cdot \lambda(\sigma)}{\lambda(a)}/2^x]/\alpha \mid \alpha \in A]$,
14: $\qquad \beta \wedge \lambda(a) \cdot 2^x > 2 \cdot \lambda(\sigma) \wedge a < 0 \wedge \gamma[\top/\alpha \mid \alpha \in A]$,
15: $\qquad \beta \wedge \lambda(a) \cdot 2^x > 2 \cdot \lambda(\sigma) \wedge a > 0 \wedge \gamma[\bot/\alpha \mid \alpha \in A]$
16: $\qquad \mid \gamma \in \Gamma, 0 \leq j \leq g\}$
17: **end for**
18: $\Sigma \leftarrow \{\sigma \mid \lambda(\sigma)$ is a subterm of some $\gamma \in \Gamma\}$
19: **for** $\sigma \in \Sigma$ **do**
20: $\quad$ **if** $\forall w_\sigma$ is not in Π' **then**
21: $\qquad w_\sigma \leftarrow$ fresh variable; add $\forall w_\sigma$ to Π' $\quad \triangleright$ Π' is a global variable
22: $\quad$ **end if**
23: **end for**
24: $\Theta \leftarrow \{(\sigma \neq 0 \wedge \neg(2^{w_\sigma} \leq |\sigma| < 2 \cdot 2^{w_\sigma}) \mid \sigma \in \Sigma\}$
25: **for** each $\Sigma' \subseteq \Sigma$ and each $\gamma \in \Gamma$ **do**
26: $\quad$ add to Θ the following formula:
27: $\quad (\bigwedge_{\sigma \in \Sigma'} 2^{w_\sigma} \leq |\sigma| < 2 \cdot 2^{w_\sigma}) \wedge (\bigwedge_{\sigma \in \Sigma \setminus \Sigma'} \sigma = 0) \wedge \gamma[2^{w_\sigma}/\lambda(\sigma) \mid \sigma \in \Sigma'][0/\lambda(\sigma) \mid \sigma \in \Sigma \setminus \Sigma']$
28: **end for**
29: **return** $\bigvee \exists x\ \theta \mid \theta \in \Theta$

#hom: In line 2 of PresQE we substitute each term with only one other term, for each formula ψ_i, so the number of homogeneous terms in ψ_i does not change from the input. The set Γ has at most $h + 1$ elements (one for each homogeneous term in φ plus one for the pair $(1, 0)$ in T), so there are $(h + 1) \cdot h = h^2 + h$ homogeneous terms in $\bigvee \Gamma$ in total.

heft: For each homogeneous term t of heft k in φ we can make at most one substitution into t. This introduces at most k new variables, so the heft at most doubles (and the minimum term follows from the bound on heft by number of variables). The disjunction of a set of formulae leaves the heft unchanged, so the same bound holds for $\bigvee_{\psi_i \in \Gamma} \psi_i$.

Maximum coefficient: Assuming we have atomic formulae $a_1x+t_1 < 0$ and $a_2x+t_2 < 0$ in φ, consider the following substitution:

$$(a_2x + t_2 < 0)\left[\frac{t_1 + k}{a_1}/x\right] = a_2\left(\frac{t_1 + k}{a_1}\right) + t_2 < 0$$
$$= a_2(t_1 + k) + a_1t_2 < 0.$$

Note that x does not appear in $a_2(t_2 + k) + a_1t_2 < 0$, so no further substitutions are done by PresQE. Assuming that a_1 and a_2 are both equal to a, and that t_1 and t_2 are both the variable y each with coefficient a, it follows that a_2t_1 and a_1t_2 each have coefficient a^2 and their sum has coefficient $2a^2$. Taking the disjunction of Γ does not change any coefficients, so the same bound holds for $\bigvee_{\psi_i\in\Gamma}\psi_i$.

Maximum constant: As above, consider the result of a substitution $a_2(t_1+k)+a_1t_2 < 0$. Assuming that a constant c appears in both t_1 and t_2, and that k assumes its maximum value of am, the maximum coefficient of each ψ_i is $a^2m + 2ac$, likewise for $\bigvee_{\psi_i\in\Gamma}\psi_i$.

 # variables: PresQE eliminates the variable x and does not introduce any new variables, so each ψ_i has $v - 1$ variables, as does $\bigvee_{\psi_i\in\Gamma}\psi_i$. □

Now we give the complexity of the SemCover subroutine.

Lemma 2. *On input (x, φ), SemCover returns a formula satisfying the parameter table:*

	#hom	*heft*	*max. const.*	*#vars.*
φ	h	$k \geq 2$	c	v
$\bigvee_{\psi_i\in\Gamma}\psi_i$	$h \cdot 2^6k \log c$	k	$2^{12}k^2c^3$	$v + h$

Proof. As before, we analyse the growth of each parameter in turn.

#hom: We analyse the number of homogeneous terms introduced relative to each homogeneous term t originally in the formula; we consider separately the number of substitutes of t introduced (that is, terms introduced in lines of the form $\gamma[\alpha'/\alpha \mid \alpha \in A]$) and the number of non-substitute homogeneous terms introduced. For substitutes we have:
Line 9: We introduce $g + 1$ terms here, for each value j can take.
Lines 12–15: At most 1 homogeneous term is substituted in on each of these lines.

Line 27: For each $\lambda(\sigma)$-substitute introduced above we create two new homogeneous terms, hence at most double the number of homogeneous terms introduced above.

In total this gets us $2g + 10$ new homogeneous terms. We then bound 2^g as follows:

$$\begin{aligned} 2^g \leq 2^7 \cdot (ck + c)^2 &\leq 2^7(c^2k^2 + 2c^2k + c^2) \\ &\leq 2^8 \cdot c^2 \cdot k^2. \end{aligned}$$

By taking logarithms of both sides we get the following bound on g:

$$\begin{aligned} g \leq \log(2^8 \cdot c^2 \cdot k^2) &\leq 8 + 2\log(c \cdot k) \\ &\leq 10 \cdot \log(k) \cdot \log(c). \end{aligned}$$

Thus we add at most $2g + 10 = 20 \cdot k \cdot \log(c) + 10$ substitute homogeneous terms.

For non-substitutes we add the following homogeneous terms:

Line 8: We add 1 homogeneous term when defining β.

Line 9: We add $g + 1$ homogeneous terms, as before, for each $2^x = 2^j$ atomic formula.

Line 10: We add 2 homogeneous terms here.

Lines 11–15: We add 1 homogeneous term here (not double-counting introduced terms).

Line 24: We add 4 homogeneous terms here.

Together these get us $g + 10 = 10 \cdot k \cdot \log c + 10$ homogeneous terms in total. Summing up all these terms, both substituted and new, gets us $30 \cdot k \cdot \log c + 19$ terms, which is less than $2^6 \cdot k \cdot \log c$ (assuming, as we may, that $k \geq 2$ and $c \geq 1$).

heft: At no point in SemCover is a hefty term introduced, nor a substitution made that increases the heft. We see this by looking at the program line-by-line:

- formulae added in lines 9–15 are all 1-heft or 2-heft,
- substitutions made in lines 9–15 either replace a term with a constant (as in line 9) or an exponential with another (as in line 12), neither of which increase the heft,
- formulae added in line 24 increase the heft of σ by 1, but since σ originally appeared together with at least one other variable this does not change the overall heft,
- formulae added in line 27 likewise do not increase overall heft, and
- substitutions made in line 27 only replace terms with the same number of variables as the substituted term.

So the heft of the input, so long as it is at least 2, does not increase with calls to SemCover.

Maximum constant: First we bound 2^g, recalling the similar bound in the #hom case. The size of $\|\eta\|_1$—the sum of all constants and coefficients of η—is bounded above by $2k \cdot c$, so the bound on 2^g is $2^7 \cdot (2kc + c)^2 \leq 2^{11} c^2 k^2$. Then consider substitutions in line 9, particularly into the term $\alpha = c \cdot 2^x + c$, where the following bound holds:

$$\|\alpha[2^j/2^x]\| \leq c \cdot 2^j + c \leq c \cdot 2^{11} c^2 k^2 + c \leq 2^{12} c^3 k^2. \tag{3.1}$$

No other substitution increases the maximum constant by this much, nor does any other introduced formula, so we have our bound.

#vars: We introduce at most $|\Sigma|$ many definitional quantifiers—that is, quantifiers introduced on line 21—and $|\Sigma|$ is bounded in size by h. So we add at most h new variables to the output formula. □

In the next section we argue that the destratification algorithm terminates, and that it really does quantifier elimination as promised.

3.2.1 Correctness of the destratification algorithm

Proposition 1. *The destratification algorithm terminates correctly.*

Proof. We cover termination and correctness separately.

Termination. The termination of PresQE, SemCover, and Linearise are each proved in [BCM23]. The modifications we have made here are minor, so the termination arguments from [BCM23] apply here as well.

For the termination of the destratification algorithm, take the variant of the inner loop to be the number of different ∃- or ∀-blocks Π. The following lemma guarantees that, after each inner iteration, the quantifier prefix Π gets shorter as more of φ is in Strat-Sem:

Lemma 3. *After each inner iteration, the formula $\exists \boldsymbol{x} \cdot \psi$ (as defined on Line 2) has all its stratum-1 variables in Strat-Sem.*

Proof. The **for**-loop on lines 11–13 guarantees that all stratum-1 variables appearing in ψ have SemCover called on them. The output of SemCover puts all these variables in

Strat-Sem, so there are no stratum-1 variables at the end of an inner iteration that appear in non-Strat-Sem formulae. □

Note that all variables introduced by the definitional quantifiers in Π' are in Strat-Sem, so adding quantifiers in Π' to ψ does not increase the length of the prefix Π. So the subformula ψ of φ in Strat-Sem strictly grows and Π strictly shrinks with each inner iteration. The inner loop terminates when Π is empty, so always terminates.

The variant of the outer loop is the maximum stratum of the whole formula φ. For this we prove that we can decrease the stratum of each variable after each outer iteration.

Lemma 4. *If s is a stratification function for the formula φ, then the new stratification function s', defined by $s'(x) = s(x) - 1$, is a stratification for the output φ' of one outer iteration of the destratification algorithm.*

Proof. After an outer iteration there are no stratum-0 variables left—they've all been eliminated by PresQE—so we don't have to worry about there being negative strata in s'.

All the old stratum-1 variables have been linearised in φ', and because they previously appeared in PowCmp they now appear only linearly in φ'. Because linear variables are stratum-0 variables, we know that the new stratum for each of these variables is 0.

With a stratum-$k \geq 2$ variable, we do an implicit induction. If x is of stratum k, then it must appear in some set of formulae Φ, with each element of the form $\Sigma a_i y_i < \sum b_j \cdot z_j$, where all y_i are stratum $k-1$, all z_j are stratum k, and where $x = z_j$ for some value of j. Inductively, in the new formula φ', all variables in y_i now have stratum $k-2$. So we can set all variables z_j to be stratum $k-1$, so x has had its stratum decreased by 1. □

Using Lemma 4, we can take the number of strata assigned by a minimal stratification function as our loop variant. That always strictly decreases, and when we have no strata left the algorithm terminates (since the only quantifier-free sentences are $\top$ and $\bot$). So we've guaranteed termination of the destratification algorithm.

Correctness. Quantifier-freeness follows immediately from Lemma 4. Since the outer loop terminates, and only does so when φ is quantifier-free, we know the output of the destratification algorithm is quantifier free.

Equivalence follows from the correctness of subroutines PresQE, SemCover, and Linearise; from [BCM23] we know that the output of each of these (strictly speaking, the disjunction over the formulae in the output) is equivalent to the input formula.

Since all function calls made in the destratification algorithm return equivalent formula, and since the output of the destratification algorithm is just the result of all these function calls, we conclude that the algorithm correctly implements its specification and returns an equivalent quantifier-free formula to the input formula. □

3.2.2 Complexity of the destratification algorithm

Here we prove that the fixed-stratum decision problem is elementary, by another analysis of the destratification algorithm.

Lemma 5. *After i iterations of PresQE on lines 5–9, the following parameter table obtains:*

	#hom	*heft*	*max. coef.*	*#vars.*
φ	h	k	a	v
φ'	h^{i+1}	$\min\{v-i, 2^i \cdot k\}$	$2^{(2^i-1)} \cdot a^{2^i}$	$v-i$

where φ' is the result of the i iterations of PresQE.

Proof. This essentially follows from iterated application of Lemma 1, though there are some technical tricks in bounding #hom. We go into some more detail in what follows:

- **#hom:** A naïve analysis gets us the bound of h^{2^i}, but [BCM23] improves the analysis: after each call to PresQE we get a set of pairs, and instead of calling PresQE on their disjunction we instead call PresQE on each individual pair. By Lemma 1, we get $h+1$ of these pairs, each with h homogeneous terms. Iterating this procedure we then get at most h^2 pairs each with h terms, then h^3, and so on. So the final bound on homogeneous terms is h^{i+1}.
- **heft:** If we have the formula $x < t_1 \wedge x < t_2$ in φ, then eliminating x gets us the subformula $t_1 < t_2$. If t_1 and t_2 each have heft k, then $t_1 < t_2$ has heft $2k$.

  The min in the parameter comes from the bound on heft by the number of variables appearing in φ. We subtract i because we've eliminated i variables by the end.
- **Maximum coefficient:** One iteration of PresQE blows up the maximum coefficient a to $2a^2$, from Lemma 1. We just need a closed solution to the recurrence $a_{n+1} = 2a_n^2$.

We can see that the exponent of c doubles after each iteration, and that the coefficient is squared then doubled: so the closed form is $2^{2^i-1} \cdot a^{2^i}$, as per the parameter table.

- **#variables:** Since PresQE introduces no variables and only eliminates a single variable per call, if we call it i times, then we simply remove i variables. □

Lemma 6. *After j iterations of the* for*-loop containing SemCover on lines 11–13, the following parameter table obtains:*

	#hom	*heft*	*max. coef.*	*#vars*
φ	h	k	a	v
φ'	$h \cdot (2^7 k^3 \log a)^j + v^2 (2^7 k^3 \log a)^{j-1}$	k	$a^{2^{2kj+2}}$	$v + h$

where u is the number of terms in φ' that each SemCovered variable is compared to.

Proof. We iterate the parameter table from Lemma 2.

- **#hom:** We give the bound as a recurrence, so want a closed solution to the following:

$$h_{n+1} = h_n \cdot 2^7 k_n^3 \log c_n + v_n^2.$$

Thankfully $k_{n+1} = k_n$ so we can essentially ignore the parameter k. From Lemma 2 we substitute in c_n, and since c_n is increasing we replace it by its largest element. Then we note that after each iteration we multiply h by $2^7 k^3 \log c$, so this gets us the upper bound for the coefficient $(2^7 k^3 \log c)^j$ in the closed form. The term v^2 ends up with the coefficient $(2^7 k^3 \log c)^{j-1}$ by a similar line of reasoning.

- **heft:** SemCover doesn't change the heft of the formula, so this remains unchanged.
- **Maximum coefficient:** Here we want a closed form to the recurrence $a_{n+1} = 2^{12} k_n^2 a_n^3$. Treating k_n as constant, an upper bound for the closed form is $2^{12 \cdot 4^{j-1}} k^{2 \cdot 4^{j-1}} a^{3^j}$. Slackening this by moving variables into the exponent gets a bound of $a^{2^{2kj+2}}$.
- **#variables:** We add a w_σ-variable in line 21 for each σ-term, and the number of introduced σ-terms is bounded by the homogeneous terms in the input. □

Finally we analyse the destratification algorithm as a whole.

Theorem 1. *Given $k \in \mathbb{N}^+$, the k-strata fragment of Semënov arithmetic is elementary.*

Proof. We begin with a complexity analysis of an outer iteration.

Lemma 7. *After one outer iteration of the destratification algorithm on input φ, where φ has v_0 stratum-0 variables, the following parameter table obtains:*

	#hom	heft	max. coef.	#s0 vars.
φ	h	k	a	v_0
φ'	$h^{v_0+1} \cdot 2^{20k^2} \cdot \log c^{k+1} + v_0^2$	$\min(2^{v_0} \cdot k, v)$	$2^{(2^{v_0}-1)} \cdot a^{2^{v_0}}$	$v_0 + (2a+1)^{\min\{k,v_0\}}$

Proof. We go over each parameter in turn.

#hom. We look at the increase from both PresQE and SemCover:

PresQE: Applying Lemma 5, we blow up the number of homogeneous terms by h^{v_0+1}.

SemCover: From [BCM23, Proposition 38], where repeated calls to SemCover are executed, we get a bound of $h \cdot 2^{20k^2} \cdot \log(c)^{k+1} + v_0^2$, where c is the maximum constant term in φ. Given we had $h^{2^{v_0}}$ homogeneous terms after lines 5–9, we end up with $h^{2^{v_0}} \cdot 2^{20k^2} \cdot \log(c)^{k+1} + v_0^2$ homogeneous terms after lines 11–13.

heft: As in the heft bound for iterated calls to PresQE (Lemma 5), assuming that stratum-≥ 2 terms are repeatedly being substituted for stratum-0 terms, we get at most $2^{v_0} \cdot k$ heft in the output. Heft is bounded by number of variables, hence the minimum.

Maxmimum coefficient: The blowup comes from Lemma 5; subsequent runs of SemCover do not change the blowup of the maximum coefficient as SemCover only changes coefficients of stratum-1 variables that are later linearised.

 # stratum-0 variables. We first have v_1 variables from the stratum above being brought down to stratum 0; this is the first summand. The second summand comes from the w_σ-variables introduced by SemCover. We prove this in the following lemma.

Lemma 8. *All w_σ-variables introduced in lines 20–22 of the SemCover algorithm, when η is a stratum-1 term, are (initially) stratum-1 variables.*

Proof. The w_σ-variables are introduced in two places in SemCover: in fresh atomic formulae $2^{w_\sigma} < \sigma$ and in substitutions of $\lambda(\sigma)$ terms for 2^{w_σ}, on lines 24 and 27 of SemCover.

On line 24, since σ must be a stratum-0 term (originally we had $2^x + \sigma < 0$ in our formula, where x was stratum-1, so σ must be stratum-0), it follows that 2^{w_σ} is a stratum-0 term. So w_σ is a stratum-1 variable. On line 27, all $\lambda(\sigma)$-terms are compared to a multiple of 2^x, where x is stratum-1. So when substituting we get a formula $a \cdot 2^x < 2^{w_\sigma}$, and w_σ must be stratum-1 as well. □

So all new variables introduced by SemCover are stratum-1 variables; after linearising they become stratum-0 variables. A quick analysis tells us there are at most $(2c_0+1)^{\min k, v_0}$ of these new variables, hence the second part of the sum in the parameter table.

 # stratum-1 variables. Since we introduce no new stratum-2 variables before linearising (by Lemma 8), all the new stratum-1 variables are just the old stratum-2 variables from the input formula. So this is our bound and nothing interesting goes on here. □

Finally we prove the fixed-stratum decision problem is elementary. Given an input formula φ with at most k strata, we iterate the parameter table from Lemma 7 k times for our upper bound on each parameter. The only relevant parameter not covered by the table is the total number of (not necessarily stratum-0) variables in the output, but noting that $v = \sum_{i\geq 0} v_i$ the total number of output variables can be bounded as follows:

$$\begin{aligned} v_0 + (2a+1)^{\min\{k,v_0\}} + \sum_{i\geq 1} v_i &= \sum_{i\geq 0} v_i + (2a+1)^{\min\{k,v_0\}} \\ &= v + (2a+1)^{\min\{k,v_0\}}. \end{aligned}$$

So the bound on #vars. is essentially the same as the bound on #s0 vars.

Then note that all parameters, in each iteration, blow up at most doubly-exponentially in any other parameter. So after k iterations of the parameter table each parameter has blown up at most $2k$-exponentially. Since the runtime of the destratification algorithm depends only on each parameter of the output formula it follows that the destratification algorithm terminates in elementary time. By the correctness of the destratification algorithm (Proposition 1) the fixed-stratum decision problem is elementary. □

Chapter 4

Simple Semënov and the iterated powers fragment

...Yet if there is a problem with stratigraphy, it is the converse: its development is increasingly punctuated by paradigm shifts triggered by new theories and technological breakthroughs.
– Smith, Bailey, Burgess, & Fraser

Here we cover the algorithm for the *simple Semënov* (4.1) and *iterated powers* (4.1.1) fragments, and argue the destratification algorithm decides both in elementary time (4.2).

4.1 A definition of the simple Semënov fragment

Recall the definition of the stratified fragment (Definition 3). We now define the *simple* fragment: the fragment where variables only appear in powers and towers of powers.

Definition 7 (The Simple Semënov fragment)**.** A *Simple Semëvov* term is defined by the following two inductions. First we define *tower terms*:

- Each variable x is a tower term.
- If t is a tower term then 2^t is a tower term.

Then we define *simple terms*:

- The constant terms 0 and 1, each variable x, and each tower term t are simple terms.
- If t_1 and t_2 are simple terms then $t_1 + t_2$ and $t_1 - t_2$ are simple terms.

A *simple formula* is a formula where:

(A) all terms are simple, and

(B) where no variable appears in two tower terms of different heights.

The *Simple Semënov fragment* is the set of all simple Semënov formulae.

So tower terms are all of the form $x, 2^y, 2^{2^z}, \ldots$, and simple terms are sums of tower terms. Once a variable appears at a height, it appears at that height in all of the formula.

For instance, a simple Semënov formula is $x + 2^y < 2^{2^z}$. The formula $2^{x+y} < z$ is not a simple Semënov formula, even if we flatten it to $\exists w\ w = x + y \wedge 2^w < z$; the variable w appears in two different tower terms; so $2^{x+y} < z$ is not simple.

We now show that the simple fragment is a sub-fragment of stratified Semënov.

Proposition 2. *Every simple formula is stratified.*

Proof. Take any simple formula φ and let $\boldsymbol{x}$ be the set of all variables appearing in φ. For each $x \in \boldsymbol{x}$ let k be the height at which x appears in a tower term; by condition (B) this is well-defined and each variable has a unique such k. Then define the stratification function s by $s(x) = k$. By definition of s each tower term t in φ has $s(t) = 0$, and each of the seven conditions of Definition 2 are easily verifiable. So φ is stratified. □

4.1.1 Interesting sub-fragments of simple Semënov

Here we look at natural generalisations of the predicate fragment (from [Sem80]), and show that they are in the simple fragment.

Iterated powers. The simple fragment includes the *iterated powers fragment*, as follows:

Definition 8 (Iterated powers)**.** The *iterated powers* fragment of Semënov arithmetic is the extension of Presburger arithmetic with predicates $P_i(x)$, for each $i \geq 1$, each expressing that x is an i^{th} power of 2.

When deciding the iterated powers fragment we expand each predicate P_k as follows:

$$P_k(x) \iff \exists z_1 \exists z_2 \ldots \exists z_k\ (x = 2^{z_1} \wedge z_1 = 2^{z_2} \wedge \cdots \wedge z_{k-1} = 2^{z_k}) \tag{4.1}$$

where each variable z_i is fresh and disjoint from all other variables in the formula. Because all z_i are fresh, we can bring all $\exists z_i$ quantifiers out to a single outermost quantifier block in the final formula; thus this expansion introduces at most ℓp quantifiers in a single block, where ℓ is the highest power seen and p is the number of P_k predicates in the input.

Now we prove that the iterated powers fragment is a sub-fragment of simple Semënov.

Proposition 3. *Every iterated powers formula can be converted to a simple formula.*

Proof. Take any iterated powers formula φ and expand it as in Equation 4.1. Exponentiation in φ only appears in expanded predicates, so all terms in φ are simple and condition (A) of the definition of simple Semënov is satisfied. Each exponentiated variable appears exactly once in Equation 4.1, so condition (B) of the definition of simple Semënov is satisfied. So φ is simple. □

Because of Proposition 3, we only need to give an algorithm for the simple fragment to decide both simple Semënov and the iterated powers fragments; we introduce the iterated powers fragment only to highlight the natural generalisation of the predicate fragment.

Unary-lifted Presburger. The simple Semënov fragment also contains the *unary-lifted Presburger* fragment. To define that we first define the *lifted Presburger* fragment.

Definition 9. Let $\varphi(x_1, \ldots, x_n)$ be a Presburger formula. The formula $\mathsf{Lift}_\varphi(x_1, \ldots, x_n)$ is defined as follows:

$$\mathsf{Lift}_\varphi(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \exists z_1 \ldots \exists z_n \bigwedge_{1 \leq i \leq n} (x_i = 2^{z_i}) \wedge \varphi(z_1, \ldots, z_n). \tag{4.2}$$

Lifted Presburger extends Presburger arithmetic by Lift_φ for each Presburger formula φ.

For instance, using the formula $\varphi(x, y, z) \stackrel{\text{def}}{=} x + y = z$, we define multiplication on powers of two with $\mathsf{Lift}_\varphi(x, y, z)$; we have that $\mathsf{Lift}_\varphi(x, y, z)$ holds *iff* x, y, and z are all powers of two with the exponents of x and y summing to the exponent of z. Similarly, using the formula $\psi(x) \stackrel{\text{def}}{=} \exists y\ 2 \cdot y = x$, we define the set of powers of four with $\mathsf{Lift}_\psi(x)$; we have that $\mathsf{Lift}_\psi(x)$ holds *iff* x is a power of two with an even exponent.

The iterated powers fragment is clearly a sub-fragment of stratified Semënov, but interestingly, in the unary case—that is, when we restrict to the extension of Presburger arithmetic with $\mathsf{Lift}_\varphi(x)$ for all Presburger formulae φ *in one free variable only*—we get a sub-fragment of simple Semënov, by the following proposition.

Proposition 4. *The unary-lifted Presburger fragment is a sub-fragment of simple Semënov.*

Proof. We argue that, using modular arithmetic, every unary-lifted predicate Lift_φ can be rewritten using only the powers-of-two predicate $P_1(x)$.

We start with a motivating example. Suppose we want to define the set $S = \{2^0, 2^2, 2^4, \ldots\}$, the set defined by the formula $\mathsf{Lift}_\varphi(x)$ where $\varphi(x) := \exists y\ x = 2y$. Each

value S is congruent to 1 mod 3. We can define both the set of numbers that are congruent to 1 mod 3 (by $\exists y\ x = 3y + 1$) and the set of powers of two (by $P_1(x)$). Together these formulae define the set of powers of two congruent to 1 mod 3, as we wanted.

Generally, all the elements of the set of powers $\{2^{0n}, 2^{1n}, 2^{2n}, \dots\}$ are congruent to 1 mod $(2^n - 1)$, and all the elements of the set of powers $\{2^{0n+k}, 2^{1n+k}, 2^{2n+k}, \dots\}$ are congruent to 2^k mod $(2^n - 1)$. We know Presburger arithmetic only defines unions of sets of the form $\{i \cdot n + k \mid i \in \mathbb{N}\}$ [GS64], so these congruences suffice to define all lifted sets. □

The upshot of this is that simple Semënov contains already interesting fragments. With the motivation done, we now prepare to analyse an algorithm for the simple fragment.

4.2 The complexity of Simple Semënov

Here we specialise the analysis of the destratification algorithm from chapter 3 to the simple fragment, and argue that regardless of the number of strata in the input formula we get an elementary algorithm.

Lemma 9. *On the first outer iteration of the destratification algorithm, including the call to Linearise, the following parameter table obtains:*

	#hom	*s0 heft*	*max s0 coef.*	*# s0 vars.*	*# s1 vars.*
φ	h	k_0	a_0	v_0	v_1
φ'	$h^{2^{v_0}} \cdot 2^{20k^2} \cdot \log c^{k+1} + v_0^2$	2	1	$v_0 + (2a_0 + 1)^{\min\{k, v_0\}}$	v_2

where:

- φ' *is the result of applying the 2–9 loop to* φ*,*
- v_0 *is the initial number of stratum-0 variables,* v_1 *the initial number of stratum-1 variables, and* v_2 *the initial number of stratum-2 variables, and*
- k *is the overall heft of the formula;* k_0 *is the heft of the stratum-0 atomic formulae.*

Proof. We cover each parameter in turn.

#hom. The same analysis from Lemma 7 works just as well here.

stratum-0 heft. In line 13 all stratum-1 variables are in PowCmp; hence they're all of the form $a \cdot 2^x < b \cdot 2^y$ or $a \cdot 2^x < c$. After linearising these become $x + y + \log(\frac{a}{b}) < 0$ or $x + \log(\frac{a}{c}) < 0$; these have heft at most 2. All stratum-0 formulae after linearising come into being this way, so we have our bound.

max stratum-0 coefficient. After linearising the maximum coefficient over all stratum-0 formulae is 1; this is immediate from the Linearise procedure. So we get our bound of 1.

 # stratum-0 variables. The analysis here is the same as in Lemma 7.

 # stratum-1 variables. Since we introduce no new stratum-2 variables before linearising (by Lemma 8), all the new stratum-1 variables are just the old stratum-2 variables from the input formula. So nothing interesting goes on here. □

Now we know that the parameter table introduces strict bounds on the heft and coefficients of the terms of the formula. Our next lemma says that this heft is preserved during the main loop of the destratification algorithm.

Lemma 10. *If φ has heft 2, then* PresQE(φ) *and* SemCover(φ) *both also have heft 2.*

Proof. We show that both PresQE and SemCover by themselves keep us in this fragment.

PresQE. For heft, note that all formulae in our fragment have at most two variables appearing in them; so all formulae relevant to PresQE substitutions are of the form $\pm x \pm t(y) < 0$, where $t(y)$ is a term where only y appears as a variable. Hence we only ever replace x with terms $t(z)$ where only z appears as a variable, and these substitutions leave the heft of the formula unchanged.

SemCover. For heft, we look at the lines where SemCover edits the formula.

Line 9: We introduce a heft-1 subformula $2^x > 2^g$, which is harmless, and replace 2^x by the constant 2^j, which doesn't increase the heft.

Line 10: Again we introduce harmless heft-1 formulae, or replace occurrences of x by v; this doesn't increase the heft.

Lines 11–15: The only reason we might get a heftier formula here is if a $\lambda(\sigma)$-term is replaced by one with too many variables. But by Line 27 all substitutions for $\lambda(\sigma)$-terms contain just one fresh variable, so the heft does not increase in these substitutions.

Line 24: Since we start in the heft-2 fragment, each σ-term has at most one variable in it. So the formulae introduced here have heft 2 and are harmless.

Line 27: As mentioned above the substitutions here are harmless, and as in Line 24 the new formulae introduced have heft 2 as well.

That covers all cases, so SemCover preserves heft-2 formulae. □

We now argue that the maximum coefficient seen in an iteration of the destratification algorithm is bounded by a single exponential.

Lemma 11. *The maximum coefficient immediately before linearising, in an iteration of the destratification algorithm, is at most* $2^{7+\log c} = c \cdot 2^7$.

Proof. In PresQE, a coefficient only increases in a formula $\pm x \pm t(y) < 0$ if $t(y)$ is substituted for x. But then we have $\pm 2t(y) < 0$ and any further substitutions must delete y. So the maximum increase in coefficients from PresQE is 2.

In SemCover, the main increase in coefficient come from Lines 12 and 13; nowhere else in the substitutions do we increase coefficients by more than 2 (since $\lambda(a)$ is always 1 in the thin fragment). When dealing with σ-terms, likewise, we only introduce coefficients of 2. So the increase is bounded by $2^{7+\log c}$ overall. □

With all this we can give a parameter table for an iteration of the destratification algorithm on a formula in the 2-heft-1-coefficient fragment.

Lemma 12. *Any iteration of the destratification algorithm on a 2-heft-1-coefficient formula is elementary. Further, the following parameter table obtains:*

	#hom	*heft*	*max s0 coef.*	*# s0 vars.*
φ	h	2	1	v_0
φ'	$2^{14} \cdot v^2$	2	1	$2v_0 + 3v_1$

where v is the total number of variables (of any stratum) in the formula.

Proof. Again, we go through each of the parameters in turn.

#hom. Since we remain in the heft-2 fragment, and since the greatest coefficient appearing in our formula is 2^7, we know all our atomic formulae are of the form $a \cdot t_1(x) < b \cdot t_2(y)$ where t_1 and t_2 are either linear variables or towers of powers, and $a, b \in [-1, 2^7]$. Combinatorically there are $2^{14} \cdot v^2$ ways of assembling such a formula, so $2^{14} \cdot v^2$ homogeneous terms at the end of an iteration of the destratification algorithm.

heft and max stratum-0 coefficient. The bound on heft follows from Lemma 10 and the bound on stratum-0 coefficients follows from the fact we've just linearised our formula.

 # stratum-0 variables. We first consider the new stratum-0 variables introduced by SemCover. We introduce one of these for each σ-term we see, and by considering the structure of our formulae there are at most $2(v_0 + v_1)$. We also linearise all the old stratum-1 variables, so add a v_1 term. In total this gets us the bound of $2v_0 + 3v_1$ as we wanted.

 # stratum-1 variables. As before we don't introduce any stratum-2 variables in the destratification algorithm, so after linearising all our stratum-1 variables are just those we started out our formula with. □

With this, we now reach the main result of this section:

Theorem 2. *The destratification algorithm is elementary on the simple fragment.*

Proof. The parameter table of Lemma 9 obtains on the first iteration of the destratification algorithm, runs in elementary time, and each parameter of the algorithm blows up at most doubly-exponentially.

Then we iterate the destratification algorithm $n - 1$ more times, where n is the number of strata in the original input formula. By iterating the parameter table from Lemma 12, we have at most $V = 2^n v_0 + \sum_{i \leq n} 3 \cdot 2^{n-i} v_i$ variables at the end of these iterations, and these bound the number of homogeneous terms in the output by $2^{14} \cdot V^2$. So we have an elementary number of homogeneous terms, and evaluating the quantifier-free formula will take elementary time. □

4.3 A note on the general stratified fragment

Finally we note why the results above do not generalise to the general stratified fragment. The analysis of the simple fragment from Chapter 4 does not generalise, because there are no guarantees on the heft or coefficient of the formula being processed after an iteration; so the parameter table from Lemma 12 cannot be used in the analysis.

The analysis of the fixed Semënov fragment does not make these assumptions on heft or coefficient, but the key to the elementary bound is that the number of iterations of the algorithm is fixed. In the general case we get a non-elementary blowup in the number of stratum-0 variables, which we see after two iterations of the parameter table:

Initially:	v_0	a
After one iteration:	$v_0 + 2a^k$	$2^{(2^{v_0}-1)} \cdot a^{2^{v_0}}$
After two iterations:	$(v_0 + 2a^k) + 2(2^{2^{v_0}-1} \cdot a^{2^{v_0}})^k$	$2^{2^{v_0+2a^k}-1} \cdot a^{2^{v_0+2a^k}}$

The problematic term in the blowup of v_0 is $2^{2^{v_0}-1}$; certainly $2^{v_0} \leq 2^{2^{v_0}-1}$, so together with its coefficients and exponents we see that v_0, after two iterations of the destratification algorithm, blows up faster than $v_0 \mapsto 2^{v_0}$. So for every $2i$ strata in the input formula, there is a tower of powers i high lower-bounding the number of variables in the output formula.

Because the runtime of the destratification algorithm depends on the number of variables produced, our analysis gets us a non-elementary upper bound on the stratified fragment.

This does not mean that the general stratified fragment is non-elementary, nor that the destratification algorithm is non-elementary on the general stratified fragment. But given that, in the stratified fragment, the form of terms in exponents is essentially unrestricted, it seems unlikely that the destratification algorithm as-is could get us an elementary bound.

Chapter 5

Other results

I have tried to include all of the known ideas ... that are both beautiful and easy to state.
– Donald E. Knuth

Here we simplify exponential equations into Presburger arithmetic (5.1), and briefly consider model-theoretic reducts of Semënov arithmetic (5.2).

5.1 Presburger-definable exponential equations

Here we go over solutions to equations of the following form:

$$\sum_{i \geq 0} a_i \cdot 2^{x_i} = c \cdot 2^z.$$

For equations as above mentioning no linear terms, an equivalent Presburger formula can be given. Indeed, for any equality as above that's just a sum of non-linear tower terms of k strata (see Definition 7), there exists an equivalent Semënov formula with $k-1$ strata. These results are therefore useful for implementing a generalised Linearise algorithm.

We begin with the restricted case where there are only three variables in the equation.

Lemma 13. *14 The equation $a \cdot 2^x + b \cdot 2^y = c \cdot 2^z$ has solutions just when, for i, j, k in the range $\lceil \max \log a, \log b, \log c \rceil$, we have $a \cdot 2^i + b \cdot 2^j = c \cdot 2^k$.*

Proof. Consider the bitstrings $(a)_2, (b)_2$, and $(c)_2$. We can add arbitrarily many zeroes to the end of each of these—we scale them by $2^x, 2^y$, and 2^z respectively—so we need only ensure that, for some offsets, $(a)_2$ and $(b)_2$ add up to $(c)_2$.

But if such a solution i, j, k exists, then assuming all of i, j, and k are greater than 0, we can subtract 1 from all three values and still get another solution. Further, it must be that i, j, and k are all within at most $\lceil \max\{\log a, \log b, \log c\} \rceil$ of each other; otherwise

$(a)_2 + (b)_2$ will be a longer bitstring than $(c)_2$, so not a solution.

Putting these results together, we get that the equation has solutions *iff* it has solutions in the range $\lceil \max\{\log a, \log b, \log c\} \rceil$, as required. □

Example 2. For example, we know the equation $4 \cdot 2^x + 2^y = 7 \cdot 2^z$ has no solutions, since $(a)_2 = \texttt{100}$ and $(b)_2 = \texttt{001}$. There is no way to add these together to get `111`, since we need three `1`s and only have two to work with. Hence the equation has no solutions.

We then find solutions of these equations with the following lemma.

Lemma 14. *If the equation $a \cdot 2^x + b \cdot 2^y = c \cdot 2^z$ has the solution $x = i, y = j$, and $z = k$ (in the range defined by Lemma), then all solutions of the equation are defined by the formula $y = x + (j - i) \wedge z = x + (k - i)$.*

Proof. This essentially follows from Lemma 14. Given minimal solutions i, j, and k such that the bits of $(a)_2$ and $(b)_2$ add up to the bits of $(c)_2$, all we can do is scale all of these each by 1. So the solution set is defined by a (two-dimensional) line, and we can define this in Presburger arithmetic. □

So, given a formula $a \cdot 2^x + b \cdot 2^y = c \cdot 2^z$, we can in polynomial time find solutions i, j, k and write down a Presburger formula that defines these solutions. Notably, we've decreased the stratum of the input formula by one.

Now we generalise, and consider the case with arbitrarily many variables.

Lemma 15. *The equation $\sum_i^n a_i \cdot 2^{x_i} = c \cdot 2^z$ has solutions just when it has solutions in the range $\lceil \max\{\log a_i\} \rceil$.*

Proof. Essentially the same proof as Lemma 14, where we consider the binary expansions of each coefficient a_i and how we might scale them to add up to c. □

For the sake of a linearisation procedure, the brute-force method of finding solutions (simply enumerate and check all $n \cdot \log(\max\{a_i\})$ possibilities) is efficient enough. We can again express the solution set of the equation $\sum_i a_i \cdot 2^{x_i} = c \cdot 2^{z_i}$ as a Presburger formula, by the following lemma.

Lemma 16. *The solutions $i_1, \ldots, i_p, k$ to the equation $\sum_i a_i \cdot 2^{x_i} = c \cdot 2^z$, if they exist, are defined by the Presburger formula*

$$\bigwedge_{1 \leq j \leq p} (x_j = x_0 + (i_j - i_0)) \wedge z = x_0 + (k - i_0).$$

Proof. Much the same as the proof of Lemma 14, we notice that we can only scale our solutions linearly so the solution set is defined by a $(p+1)$-dimensional plane. □

Finally, we note that if larger towers of powers appear in the formula, then we can use the same Presburger formulae and just substitute in these towers in for variables. The upshot of that is that we get an equivalent formula of a lower stratum; so we can linearise all equations of this form as desired.

5.2 Non-elementary unary reducts of Semënov arithmetic

One way to find elementary fragmennts of a non-elementary theory is to take (model-theoretic) *reducts* of that theory. We can think of taking reducts of a theory as follows: take all the definable sets of a theory, then forget some of them. The resulting theory may be semantically equivalent to the original theory—if all removed sets were definable using other sets—or it may be weaker than the original theory.

Here we want to know general properties of reducts of Semënov arithmetic that still extend Presburger arithmetic; let's call such reducts *Presburger-reducts*. For instance, the predicate fragment introduced in [BCM23] is a Presburger-reduct: all sets other than those either definable by Presburger arithmetic or the set of powers of two have been forgotten.

Because the predicate fragment is elementary, and because unary predicates are typically add less expressive power when added to a theory, we might hope that any unary-predicate-reduct of Semënov arithmetic is elementary. This turns out to be false.

Proposition 5. *The Presburger-reduct of Semënov arithmetic containing the unary predicate R, defined as follows, is non-elementary:*

$$R(x) \stackrel{\text{def}}{=} (\exists z\ x = 2^z) \vee (\exists z\ x = 2^z + z). \tag{5.1}$$

Proof. Define the predicates $P(x) \stackrel{\text{def}}{=} \exists z\ x = 2^z$ and $Q(x) \stackrel{\text{def}}{=} \exists z\ x = 2^z + z$. We can extract

these predicates out of the R-fragment by the following equivalence:

$$P(x) \iff (R(x) \wedge \neg\exists y\ (3x < 4y < 4x \wedge R(y)). \tag{5.2}$$

We then define $Q(x)$ as $R(x) \wedge \neg P(x)$. Finally we define a formula that holds *iff* $x = 2^y$:

$$\varphi(x, y) \stackrel{\text{def}}{=} P(y) \wedge Q(x + y) \wedge \neg\exists z\ (z < x \wedge Q(y + z)). \tag{5.3}$$

So we've defined the powers-of-two function, and have the full power of Semënov arithmetic. But Semënov arithmetic is non-elementary, so our Presburger-reduct is as well. □

Chapter 6

Conclusion

Beware of bugs in the above code; I have only proved it correct, not tried it.
– Donald E. Knuth

In this thesis we have seen algorithms for the simple and fixed-stratum fragments; in both cases we use the destratification algorithm (Algorithm 1), and have an in-depth complexity analysis of the algorithm. We have also seen results on the linearisation of Semënov-definable equations, and problematic unary reducts of Semënov arithmetic.

6.1 Future work and limitations

Generalising to the stratified fragment. The current analysis of the destratification algorithm does not generalise to the whole stratified fragment (by Section 4.3), so the complexity of the stratified fragment is open. Analysing the destratification algorithm on the stratified fragment may give an elementary algorithm, but if so it would be far more involved than any analysis above. Dealing with arbitrarily high-stratum formulae at each iteration seems particularly difficult.

Against an elementary analysis, there are no obvious reductions of the decision problem for stratified Semënov to a non-elementary problem. Defining a pairing function, like $2^{x+y} + x$, seems tricky: all candidates seem to require a variable to appear both linearly and exponentiated. Likewise an arithmetisation of a non-elementary Turing machine (in the style of [Mey75]), or a reduction to automatic structures (in the style of [BG00]), seem difficult; it is unclear how to encode the structure of either of these models of computation whilst keeping each variable in its own stratum.

Reducts of Semënov arithmetic Here we focused on the stratified fragment of Semënov arithmetic, restricting our theory at the formula level. It would also be interesting to consider model-theoretic *reducts* of Semënov arithmetic, restricting at a semantic level instead. For instance, we may restrict to the extension of Presburger arithmetic with a powers-of-two function defined only on powers of two themselves.

We have mentioned that there are non-elementary reducts of Semënov arithmetic that consist of Presburger arithmetic extended by a single unary predicate; so it doesn't seem like there are obvious structural results on which reducts of Semënov arithmetic are elementary. Exploring these in future work would be interesting.

Structural restrictions of Semënov arithmetic. The stratified fragment is a natural restriction of Semënov arithmetic when considering why Semënov arithmetic is non-elementary: given that all the problematic pairing functions, such as $2^{x+y} + x$, have variables that appear both in exponents and linearly in the same atomic formulae, it is natural to restrict our theory so that this sort of thing does not happen.

What other natural restrictions of Semënov arithmetic there are, such that the restriction is at least plausibly elementary, is unclear. More work in this area would make clear what exactly it is—intuitively—about Semënov arithmetic that is non-elementary.

Generalisation to the real case. Throughout the project we have worked in the integers, but there has been interesting work on the reals with powers of two. Avigad and Yin [AY07] give a primitive recursive non-elementary algorithm for deciding the real field (with multiplication) with a predicate for powers of two. It is unknown if this bound is tight, or whether there is an elementary algorithm for this extension of the real field.

Similarly it would be interesting to consider extensions of the real field by other predicate-reducts we have considered here, for instance the iterated powers or lifted Presburger fragments (as given in Section 4.1.1).

Bibliography

[AY07] Jeremy Avigad and Yimu Yin. Quantifier elimination for the reals with a predicate for the powers of two. *Theoretical Computer Science*, 370(1):48–59, 2007.

[BCM23] Michael Benedikt, Dmitry Chistikov, and Alessio Mansutti. The complexity of Presburger arithmetic with power or powers. In *ICALP 2023*, volume 261 of *LIPIcs*, pages 112:1–112:18, Dagstuhl, Germany, 2023.

[Ber80] Leonard Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–77, 1980.

[BG00] Achim Blumensath and Erich Grädel. Automatic structures. In *Proceedings of the Fifteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2000)*, pages 51–62. IEEE Computer Society Press, June 2000.

[Bru85] Véronique Bruyère. Entiers et automates finis, Mémoire de fin d'études, 1985.

[Bè02] Alexis Bès. A survey of arithmetical definability. In *A Tribute to Maurice Boffa*, pages 1–54. Société Mathématique de Belgique, 2002.

[Bü60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, Apr 1936.

[CMS24] Dmitry Chistikov, Alessio Mansutti, and Mikhail R. Starchak. Integer linear-exponential programming in NP by quantifier elimination. *LIPIcs, Volume 297, ICALP 2024*, 297:132:1–132:20, 2024.

[Coo72] David C. Cooper. Theorem proving in arithmetic without multiplication. *Machine Intelligence*, 7:91–99, 1972.

[CP86] Gregory Cherlin and Françoise Point. On extensions of Presburger arithmetic. *4th Easter Conference on Model Theory*, pages 17–34, 1986.

[FR74] M.J. Fischer and M.O. Rabin. *Super-exponential Complexity of Presburger Arithmetic*. Massachusetts Institute of Technology Cambridge, Mass. Massachusetts Institute of Technology. Project MAC, 1974.

[FR75] Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, March 1975.

[FR79] Jeanne Ferrante and Charles W. Rackoff. *The Computational Complexity of Logical Theories*. Springer Berlin Heidelberg, 1979.

[Fü82] Martin Fürer. The complexity of Presburger arithmetic with bounded quantifier alternation depth. *Theoretical Computer Science*, 18(1):105–111, 1982.

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

[Gö31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38-38(1):173–198, December 1931.

[Hei76] Joos Heintz. Untere Schranken für die Komplexität logischer Entscheidungsprobleme. In Ernst Specker and Volker Strassen, editors, *Komplexität von Entscheidungsproblemen, Ein Seminar*, Lecture Notes in Computer Science, pages 127–137. Springer, 1976.

[HS26] Christoph Haase and Mikhail Starchak. Formal languages and arithmetic theories: Recent results and open problems. In *Developments in Language Theory*, pages 1–14. Springer Nature Switzerland, 2026.

[Mey75] Albert R. Meyer. Weak monadic second order theory of succesor is not elementary-recursive. In Rohit Parikh, editor, *Logic Colloquium*, pages 132–154, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[Opp73] Derek C. Oppen. Elementary bounds for Presburger arithmetic. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, page 34–37, New York, NY, USA, 1973. Association for Computing Machinery.

[Opp78] Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

[Pre29] Mojżesz Presburger. Über die Vollständigkeit eines gewissen systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du I congres de Mathematiciens des Pays Slaves*, pages 92–101, 1929.

[RL78] C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 320–325, New York, NY, USA, 1978. Association for Computing Machinery.

[Sem80] Alexei L. Semenov. On certain extensions of the arithmetic of addition of natural numbers. *Mathematics of the USSR-Izvestiya*, 15(2):401, April 1980.

[Sem84] Alexei L. Semënov. Logical theories of one-place functions on the set of natural numbers. *Mathematics of the USSR-Izvestiya*, 22(3):587, June 1984.

[Sta84] Ryan Stansifer. Presburger's article on integer arithmetic: Remarks and translation. Technical Report TR84-639, Cornell University, Computer Science Department, September 1984.

[Wei90] V. Weispfenning. The complexity of almost linear diophantine problems. *Journal of Symbolic Computation*, 10(5):395–403, 1990.