Tiling 3D Space by (Translates of) a Single Polycube is Undecidable

Erik D. Demaine* Stefan Langerman†

Warning: This technical article was written by a large language model (LLM) and is not intended for consumption by humans. See the repository [DL26] for the human-readable overview and the complete Lean code that verifies the proof.

1 Definitions and Main Theorem

Lean references below give declaration names and link their source files on GitHub. Names omit the common prefix `TranslationTiling.`; file paths are relative to `lean/TranslationTiling/`.

For an abelian group G, a ***prototile*** is a finite nonempty subset $F \subset G$. Its translated copies are the sets $a + F$. Following the tiling-equation notation of [GT25, Section 1.1], for subsets A, F of an abelian group G, the notation $A \oplus F = G$ means that every element of G has exactly one representation as $a + f$ with $a \in A$ and $f \in F$. We say that F ***tiles*** G if such an A exists, and call A its ***set of translations***. The prototile is finite, but its set of translations is infinite when G is infinite. We allow translations only. Intermediate prototiles may be disconnected; the final prototile will be connected. A finite set $F \subset \mathbb{Z}^d$ is ***connected*** if every pair of its points can be joined by a sequence of points of F, successive points differing by a signed coordinate unit vector. Connectedness of polycubes and polyominoes always refers to this lattice adjacency, including when they are viewed as unions of unit cubes. A ***polycube*** is a finite nonempty subset of $\mathbb{Z}^3$, or its corresponding union of unit cubes; a ***polyomino*** is its two-dimensional counterpart, following the terminology of [Kim26, Section 1.2]. A tiling using a single prototile is a ***translational monotiling***. More generally, $B \oplus C = E$ denotes unique representation of every point of E as a sum, with no sums outside E; $B + C$ denotes the set of sums without a uniqueness condition, and $B - C$ is the set of differences $b - c$. For a vector u and an integer n, write $u + B = \{u + b : b \in B\}$ and $nB = \{nb : b \in B\}$. The notation $\sqcup$ denotes a disjoint union. A ***simultaneous tiling system*** consists of a finite family of prototiles $F_1, \ldots, F_s$ in a common target space G and asks for one set of translations A satisfying $A \oplus F_\nu = G$ for every ν, as in [GT24, Section 3]. Each prototile must individually tile the entire target space using the same set of translations.

For a finite nonempty $F \subset \mathbb{Z}^d$, its ***unit-cube union*** is $F + [0,1]^d$. In $\mathbb{R}^d$, a ***tiling by translations*** means that each point belongs to exactly one translated copy, except on a set of Lebesgue measure zero; this is coverage with multiplicity one almost everywhere [GT24, Section 1.1].

A subgroup Λ has ***finite index*** when there are finitely many ***cosets*** $g + \Lambda$; the group of these cosets is the ***quotient*** G/Λ. A set A is ***invariant*** under Λ if $A + \lambda = A$ for every $\lambda \in \Lambda$. We call a set of translations ***periodic*** if it is invariant under a finite-index subgroup, following [GT24, Section 1.1].

*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, edemaine@mit.edu

†Computer Science Department, Université libre de Bruxelles, Belgium, stefan.langerman@ulb.ac.be. S. Langerman is Directeur de Recherche du F.R.S.-FNRS.

A ***Wang tile*** is a unit square with a color on each of its east, north, west, and south edges; a ***Wang tiling*** assigns tiles from a finite set to $\mathbb{Z}^2$, without rotating or reflecting them, with equal colors on touching edges. A Wang tile set tiles $\mathbb{Z}^2$ when such an assignment exists [GT25, Definition 2.6].

A decision problem is ***decidable*** if an algorithm halts on every input with the correct yes or no answer. It is ***recursively enumerable*** (RE) if an algorithm halts exactly on its yes instances, and ***co-recursively enumerable*** (co-RE) if an algorithm halts exactly on its no instances. A ***computable many-one reduction*** is a computable map from inputs of one problem to inputs of another, preserving yes and no answers. A problem is ***co-RE-hard*** if every co-RE problem has such a reduction to it, and ***co-RE-complete*** if it is also in co-RE. For an encoding, ***completeness*** means that an original solution yields an encoded solution; ***soundness*** means that an encoded solution implies an original solution.

Theorem 1.1. *There is a computable map from finite nonempty Wang tile sets $\mathcal{T}$ to finite nonempty connected sets $F_{\mathcal{T}} \subset \mathbb{Z}^3$ such that*

$$\mathcal{T} \text{ tiles } \mathbb{Z}^2 \quad \Longleftrightarrow \quad F_{\mathcal{T}} \text{ tiles } \mathbb{Z}^3 \text{ by translations.}$$

Consequently, translational monotiling of $\mathbb{Z}^3$ is co-RE-complete under computable many-one reductions, even when the input prototile is required to be connected.

The reduction is `Compiler.connectedCompile`, with `Compiler.connectedCompile_computable`, `Compiler.connectedCompile_connected`, and `Compiler.connectedCompile_correct` in `Compiler/Main.lean`. The completeness conclusion is `Abstract.tiling_coRE_complete` in `Abstract.lean`.

2 The Two-Prime Sudoku Rule

We use $\mathbb{N} = \{0, 1, 2, \ldots\}$ and the domino-system formulation of [GT25, Definition 2.2]. A ***domino system*** consists of a finite nonempty alphabet $\mathcal{T}$ and two allowed-pair relations $H, V \subseteq \mathcal{T}^2$. A ***solution*** on a subset of $\mathbb{Z}^2$ assigns a symbol $\tau(u, v)$ to each point, with

$$(\tau(u, v), \tau(u + 1, v)) \in H, \qquad (\tau(u, v), \tau(u, v + 1)) \in V$$

whenever both points of the pair are in the domain. For a finite Wang tile set, take its tiles as $\mathcal{T}$, put $(t, t') \in H$ when the east color of t equals the west color of t', and put $(t, t') \in V$ when the north color of t equals the south color of t'. This correspondence is recorded in [GT25, Definition 2.6 and equations (2.1)–(2.2)].

Solvability on $\mathbb{N}^2$ is equivalent to solvability on $\mathbb{Z}^2$ [GT25, Lemma 2.5]. Indeed, a solution on the quadrant provides solutions on every finite square by restriction and translation; compactness then gives a solution on the plane. Restriction proves the reverse implication. We use the classical co-RE-completeness of this domino problem [Ber66, GT25].

For a prime r, write $\mathbb{F}_r = \mathbb{Z}/r\mathbb{Z}$ for its field of residues and $\mathbb{F}_r^\times$ for the nonzero residues. Fix the primes $p = 5$ and $q = 7$ once and for all, and set

$$N = 2q + 2 = 16, \qquad I = \{0, \ldots, N - 1\}, \qquad \Sigma = \mathbb{F}_p^\times \times \mathbb{F}_q^\times \times \mathcal{T}.$$

The board width N and coefficient period p^2q^2 are fixed independently of the Wang tile set. The alphabet has $(p - 1)(q - 1)|\mathcal{T}|$ symbols and therefore depends on that set. The finite-width board with constraints on all integer-slope lines is the Sudoku framework of Greenfeld and Tao [GT24, Definition 7.4]; we use its general word-rule formulation [GT25, Definition 3.1]. A ***word*** on I is a function $I \to \Sigma$; Σ^I denotes the set of all such words. A ***word rule*** is a subset $\mathcal{W} \subseteq \Sigma^I$. An ***array*** is a function $W : I \times \mathbb{Z} \to \Sigma$. It is a ***Sudoku array*** for the word rule $\mathcal{W}$ if

$$(W(n, dn + e))_{n \in I} \in \mathcal{W} \qquad (d, e \in \mathbb{Z}). \tag{2.1}$$

Write $W = (W_p, W_q, W_{\mathcal{T}})$ for its three components. The first two are its ***integer components***, taking values in $\{1, \dots, p-1\}$ and $\{1, \dots, q-1\}$, respectively, identified with the corresponding nonzero residues. The third is its ***Wang tile label***. The ***column*** at n is the function $m \mapsto W(n, m)$ on $\mathbb{Z}$; ***nonconstancy*** of an integer component means that this component takes at least two different values in the column; repetitions are allowed. We additionally require both integer components to be nonconstant on every column. This requirement is separate from the Sudoku word rule.

For a nonzero integer m, let $v_p(m)$ denote the largest integer $k \geq 0$ for which p^k divides m, called its ***p-adic valuation***, and let $f_p(m)$ be the residue of $m/p^{v_p(m)}$ in $\mathbb{F}_p^\times$. Set $v_p(0) = +\infty$ and $f_p(0) = 1$, and use the analogous notation for q. The ***last-nonzero-digit function*** and the convention at zero follow [GT25, Definition 4.1].

2.1 The Finite Word Rule

We now define $\mathcal{W}$ explicitly, following [GT25, Definition 5.1]. For integers a, b, call $\lambda(n) = an + b$ ***nondegenerate modulo r*** if a, b are not both divisible by r, and put

$$t_r(a) = \begin{cases} 1, & r \nmid a, \\ 0, & r \mid a. \end{cases}$$

A word $w = (w_p, w_q, w_{\mathcal{T}}) \in \Sigma^I$ is ***allowed*** if there are integers a, b such that λ is nondegenerate modulo both primes, and a domino solution

$$\sigma \colon \{0, \dots, t_p(a)\} \times \{0, \dots, t_q(a)\} \to \mathcal{T},$$

such that, for every $n \in I$,

$$\begin{aligned} w_r(n) &= f_r(\lambda(n)) && \text{if } v_r(\lambda(n)) \leq t_r(a), \quad r = p, q, \\ w_{\mathcal{T}}(n) &= \sigma(v_p(\lambda(n)), v_q(\lambda(n))) && \text{if both valuation bounds hold.} \end{aligned} \tag{2.2}$$

The coordinates not covered by these conditions are unrestricted. The domain of σ has one, two, or four points, and its domino constraints are exactly the H, V tests for adjacent points defined above.

This is a finite, effective definition. All valuation tests and required residues in (2.2) depend only on a, b modulo p^2q^2: a valuation at most one and the corresponding last nonzero digit are determined modulo r^2. Enumerate these coefficient residues, all assignments σ on the corresponding rectangle, and all words obeying their constraints. Taking their union computes $\mathcal{W}$ without an infinite search or a domino solution. The bounded search is justified by `Sudoku.allowed_iff_bounded`; its word enumeration is `Sudoku.allowedWords` in `Proofs/FiniteWordRule.lean`.

The valuations provide the two coordinates of the simulated domino plane. Given a domino solution τ on $\mathbb{N}^2$, decorate each height $m \neq 0$ with the tile $\tau(v_p(m), v_q(m))$. After removing common prime powers from the coefficients of a line, its low-valuation points test a rectangle of adjacent domino positions. The integer components record last nonzero digits so that the tests remain consistent across different lines and valuation scales. The reverse assertion, that these local tests and nonconstant columns force a global domino solution, is the substantive soundness theorem below.

2.2 The Soundness Theorem and a Canonical Solution

Theorem 2.1 (Decorated Sudoku, shorter-board variant)**.** *The word rule just defined has the following properties.*

(i) *If an array satisfies* (2.1) *and both W_p and W_q are nonconstant on every column, then the domino system has a solution on $\mathbb{N}^2$.*

(ii) *If $\tau\colon \mathbb{N}^2 \to \mathcal{T}$ is a domino solution, choose any $t_* \in \mathcal{T}$ and define the array*

$$W(n,m) = \begin{cases} \big(f_p(m), f_q(m), \tau(v_p(m), v_q(m))\big), & m \neq 0, \\ (1,1,t_*), & m = 0, \end{cases} \tag{2.3}$$

Then W satisfies (2.1)*, and both integer components are nonconstant on every column.*

The array in (2.3) is called the ***canonical Sudoku array*** associated to τ. Part (i) is a shorter-board refinement of [GT25, Proposition 5.2], formalized as `Sudoku.soundness` in `SudokuArithmetic/Soundness.lean`. Greenfeld–Tao assume primes greater than 48, and use decorated board width p^2q^2 [GT25, Theorem 4.6 and Definition 5.1]. To optimize the reduction size, our refinement reduces the prime requirement from $r > 48$ to $r \geq 5$ and needs only $M = 2r + 2$ arithmetic columns. We replace their clean-patch argument [GT25, Lemma 4.8] by five-point affine recovery, using line compatibility to handle the smaller primes. For $r = 5, 7$, seek a translate of

$$(0,0), (2,0), (2,h), (3,h), (4,2h), \qquad h \in \{2,4\},$$

whose four horizontal checks at the first four points and two diagonal triples avoid exceptional residues. The triples have slopes $h/2, h$; there are ten checked incidences in total.

A finite compatibility argument guarantees such a clean seed inside the $(2r+2)\times 13$ patch. Give each cell a nonzero field value, and each integer line of slope $-4, \ldots, 4$ meeting the patch a nonzero affine witness. Wherever a witness is nonzero, it must give the cell's value. Suppose every candidate seed of both spacings has an exceptional checked incidence. These conditions form finite propositional formulas with 58,969 clauses for $r = 5$ and 163,261 clauses for $r = 7$. Both formulas are unsatisfiable. Lean's kernel checks ordinary propositional proofs reconstructed from their LRAT certificates, including agreement of the certificate inputs with the defined formulas; no native-evaluation axiom is used. The formulas and their interpretation as array constraints are checked in `SudokuArithmetic/SmallPrimeSAT.lean`. The source modules `SudokuArithmetic/SmallPrimeCertificate5.lean` and `SudokuArithmetic/SmallPrimeCertificate7.lean` reconstruct the respective proof certificates. Thus every admissible array has a clean seed. For primes $r \geq 11$, the previous counting argument with spacing $h = 6$ remains in `SudokuArithmetic/FivePointPattern.lean`.

Write the first two horizontal affine forms as $U_j x + V_j$, for $j = 0, 1$, in coordinates relative to the chosen origin, and let $T_0 \neq 0$ be the actual value at the fifth point. The diagonal checks give

$$T_0 = 2(2U_1 + V_1) - V_0 = 2(3U_1 + V_1) - (2U_0 + V_0).$$

These force $U_0 = U_1 = A$. There is an affine form $F(n,m) = An + Bm + C$ agreeing with the first two rows wherever it is nonzero, whose value at the fifth point is T_0.

To recover the third row, write its horizontal form as $H(x) = Ux + V$ in absolute coordinates. Let D_0, D_1 be the intercepts of F on the first two rows, and put $D_2 = 2D_1 - D_0$. For each step $s \in \{-2, -1, 1, 2\}$, use a slope-h/s line through columns $x - 2s, x - s, x$ on the three rows, and set

$$f_0(x) = A(x - 2s) + D_0, \qquad f_1(x) = A(x - s) + D_1, \qquad f_2(x) = Ax + D_2.$$

When f_0, f_1, f_2, H are nonzero, the diagonal rule gives $H(x) \in \{f_2(x), 2f_1(x), -f_0(x)\}$, allowing for its own exceptional residue at the first or second intersection. An exceptional residue at the third intersection would force $f_2(x) = 0$. Each step has at least r valid consecutive column positions, so these implications hold on every field residue. For $r = 5, 7$, a finite field check forces $(U, V) = (A, D_2)$; when H is constant, the fifth seed point supplies its value. Thus F agrees with all three spaced rows wherever it is nonzero.

For spacing $h = 2$, use both slope-one and slope-minus-one lines to recover the intervening rows at offsets $t = 1, 3$ from the known rows $0, 2, 4$. Here is the finite check explicitly. On such a diagonal write the

predicted form as $g(y) = \alpha y + \beta$, put $f = g(t)$, and let η be the target horizontal witness's value. If $\eta, f \neq 0$, comparison at the three known rows permits $\eta = f$ and only these additional possibilities:

condition	additional values of η
$\beta = 0$	$2f - 4\alpha,\ -f + 4\alpha$
$2\alpha + \beta = 0$	$(f + 2\alpha)/2,\ (f - 2\alpha)/2$
$4\alpha + \beta = 0$	$-f - 4\alpha,\ 2f + 4\alpha$

Indeed, unless the predicted form vanishes at a known intersection, two nonzero comparisons identify the diagonal witness with it. In the remaining cases, its possible exceptional intersection gives the listed alternatives. Checking these conditions for every horizontal residue and both directions forces the target row's coefficients to equal those of F, whenever F is not identically zero on that row. Identically zero rows are already good.

For spacing $h = 4$, first recover rows $2, 6$ from rows $0, 4, 8$ by the same check after doubling the vertical coordinate; the diagonal slopes are now ± 2. Then use either the triple $0, 2, 4$ or the triple $4, 6, 8$. At least one triple has three nonzero row forms, because F cannot vanish identically on both rows $2, 6$. We again obtain four consecutive good rows, and four-line closure propagates the affine approximation throughout the board. The finite field checks, column ranges, and global recovery are in `SudokuArithmetic/SmallPrimeFields.lean`, `SudokuArithmetic/SmallPrimeRows.lean`, and `SudokuArithmetic/SmallPrimeRecovery.lean`. The rescaling obstruction uses three columns $n_0 + rk$, with $n_0 \in \{0, 1\}$ and $k = 0, 1, 2$. To compare vertical coefficients after rescaling, a vertical shift makes one of the affine forms take the values $B_1, 2B_1, 3B_1$ on the three prime-spaced columns, where $B_1 e 0$ is its vertical coefficient; the other can vanish at most once, leaving two comparisons. Once the ordinary slope is fixed, the higher-digit coefficient test needs only columns $r, 2r$. These refinements replace the quadratic board width in the arithmetic soundness proof; the two-prime decoration and valuation-plane extraction then apply as before. For completeness, we verify part (ii) directly. On the line $m = dn + e$ with $(d, e) \neq (0, 0)$, put

$$h_r = \min\{v_r(d), v_r(e)\}, \qquad d_r = d/r^{h_r}, \qquad e_r = e/r^{h_r} \quad (r = p, q).$$

The Chinese remainder theorem allows residues to be prescribed independently modulo coprime integers. Choose a, b so that $a \equiv d_r$, $b \equiv e_r \pmod{r^2}$ for both primes. The form $an + b$ is nondegenerate modulo each prime. Whenever its r-valuation is at most $t_r(a)$, that valuation and its last nonzero digit agree with those of $d_r n + e_r$. Multiplication by r^{h_r} does not change the last nonzero digit, and adds h_r to the valuation. Thus

$$\sigma(u, v) = \tau(h_p + u, h_q + v) \quad (0 \le u \le t_p(a),\ 0 \le v \le t_q(a))$$

witnesses every constraint in (2.2). At a point where $dn + e = 0$, both valuation bounds fail, so its arbitrary decoration causes no problem. For the line $d = e = 0$, choose $a = 0, b = 1$ and $\sigma(0, 0) = t_*$. This proves the line rule for the entire array. Both integer components are nonconstant on every column: heights 1 and 2 already give the distinct values 1 and 2. The corresponding Lean results are `Sudoku.canonical_lineRule` in `Proofs/CanonicalLineRule.lean` and `Sudoku.canonical_nonconstantColumns` in `Proofs/Canonical.lean`.

2.2.1 A Small Arithmetic Example

Let A have east and north colors 0 and west and south colors 1, and let B have the opposite colors. These two Wang tiles force a checkerboard, with $\tau(u, v) = A$ when $u + v$ is even and B otherwise. Figure 1 shows how the valuation pair addresses this tiling in the canonical array. For readability, the figure uses the toy primes $p = 3, q = 5$, as in the arithmetic illustration of [GT25, Figure 5.1]. The canonical formula and its line-rule verification above apply to these primes, but we make no soundness claim for them: Theorem 2.1

uses $p, q > 10$. Every displayed height has the same symbol in every column n. For example, height 45 gives the full symbol $(2, 4, B)$, because $f_3(45) = 2$, $f_5(45) = 4$, and $(\nu_3(45), \nu_5(45)) = (2, 1)$. This illustrates the arithmetic interface, rather than enumerating the much larger lattice prototile produced by the full reduction.

2.2.2 The Two Seed Symbol Sets

Parametrize the lines by $x = (x_1, x_2) \in \mathbb{Z}^2$, and put

$$L_n(x) = x_2 + nx_1, \qquad u_n = (1, -n).$$

Then L_n is onto $\mathbb{Z}$ and its ***kernel***, the set of points mapped to zero, is $\ker L_n = \mathbb{Z}u_n$. The two auxiliary constraints will use the symbol sets

$$B_t = \{(t, t, a) : a \in \mathcal{T}\} \subset \Sigma, \qquad t = 1, 2. \tag{2.4}$$

Here the two occurrences of t are interpreted in the respective fields. In particular, B_1 and B_2 are disjoint.

The canonical array has a useful uniform property:

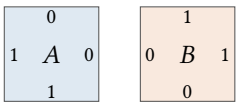
$$x_1 \equiv 0,\ x_2 \equiv t \pmod{pq} \quad \Longrightarrow \quad W(n, L_n(x)) \in B_t \quad (n \in I). \tag{2.5}$$

Indeed, $L_n(x)$ has the nonzero residue t modulo both primes. No constraint on the third coordinate is needed in this implication.

Remark 2.2. The nonconstancy hypothesis concerns each integer component, not merely the triple W. Variation of the Wang label alone would not suffice. This is the reason for imposing (2.4).

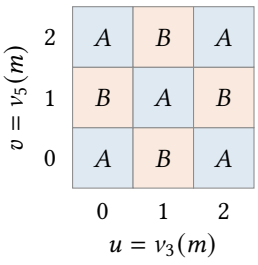
3 Encoding Finite Constraints with a Cyclic Factor

The general method of encoding finite-valued functional constraints by tiling equations was developed by Greenfeld and Tao [GT23, Sections 4–8] and [GT24, Theorem 4.1]. The graph and cycle gadgets in this section adapt [Ope26, Section 3]. We give their definitions and proofs for the decorated alphabet, and keep the prime choices explicit. OpenAI used this encoding for the one-prime Sudoku rule of [GT25, Section 4]. Here we adapt it to the two-prime Sudoku array problem, including the Wang tile constraints and both nonconstant-column conditions.



Colors 0, 1 on edges;
touching colors agree.

(a) Wang tiles



(b) Valuation addresses

m	$(\nu_3(m), \nu_5(m))$	$W_{\mathcal{T}}(n, m)$
1	(0, 0)	A
3	(1, 0)	B
5	(0, 1)	B
9	(2, 0)	A
15	(1, 1)	A
25	(0, 2)	A
45	(2, 1)	B
75	(1, 2)	B

(c) Sampled heights

Figure 1: A two-tile example of the canonical decoration (2.3). (a) Edge colors specify the Wang tiles. (b) Their checkerboard tiling supplies the decoration at each valuation address. (c) Sample heights for the toy primes 3, 5; the integer components are suppressed in the table. These small primes illustrate the formula and do not instantiate the soundness theorem. The two-prime addressing method is from [GT25, Section 5].

3.1 Coordinates and the Graph Equation

A ***channel*** is an indexed collection of input and output coordinates. Let $\mathcal{I} = I \sqcup \{\eta_1, \eta_2\}$ be their index set. Indices $n \in I$ are ***ordinary channels***; η_1, η_2 are ***seed channels***, used to enforce the two auxiliary constraints. The ***label set*** of channel i is denoted by J_i: put $J_n = \Sigma$ for ordinary channels and $J_{\eta_t} = \{*\}$ for seeds. Set

$$M = 5pq = 175, \qquad S = (\mathbb{Z}/M\mathbb{Z})^2, \qquad D = M^2, \qquad s(x) = x \bmod M. \tag{3.1}$$

For each channel i choose primes a_i, b_i, all mutually distinct, different from $5, p, q$, with

$$a_{\eta_1} = 2, \qquad a_{\eta_2} = 3, \qquad b_{\eta_1}, b_{\eta_2} > D.$$

Write $P_i = (\mathbb{Z}/a_i\mathbb{Z}) \times (\mathbb{Z}/b_i\mathbb{Z})$. For every $j \in J_i$ choose nonnegative integers A_{ij}, B_{ij} and a further prime r_{ij}, distinct from all other chosen primes, such that

$$r_{ij} = A_{ij}a_i + B_{ij}b_i, \qquad A_{ij}, B_{ij} \geq 0. \tag{3.2}$$

This is effective: if $r \geq (a_i - 1)b_i$, choose $B \in \{0, \ldots, a_i - 1\}$ with $Bb_i \equiv r \pmod{a_i}$, and put $A = (r - Bb_i)/a_i$. Searching for sufficiently large fresh primes therefore terminates.

All products indexed by channels range over $\mathcal{I}$. Let

$$r_i = \prod_{j \in J_i} r_{ij}, \qquad K_i = \mathbb{Z}/r_i\mathbb{Z} = \prod_{j \in J_i} \mathbb{Z}/r_{ij}\mathbb{Z}, \qquad K = \prod_i K_i, \qquad P = \prod_i P_i,$$

where the product description of K_i uses the Chinese remainder isomorphism. Each coordinate in this product is called an ***input digit***. Denote by U_{ij} the subgroup with arbitrary jth digit and all other digits zero; subtraction by an element of U_{ij} changes only that digit. The ***target space*** is the abelian group

$$G = \mathbb{Z}^2 \times V, \qquad V = (\mathbb{Z}/D\mathbb{Z}) \times \prod_i \big((\mathbb{Z}/r_i^2\mathbb{Z}) \times P_i\big). \tag{3.3}$$

The cyclic orders D, r_i^2, a_i, b_i appearing here are pairwise coprime, so V is cyclic. The residue set S is only an indexing set for functions; it is not a factor of V. Identifying V with $\mathbb{Z}/m\mathbb{Z}$, where $m = |V|$, gives the target space $\mathbb{Z}^2 \times (\mathbb{Z}/m\mathbb{Z})$: a 3D integer grid with a finite coordinate that wraps around modulo m. This coordinate encodes all the finite factors displayed above; the auxiliary output z below takes values in just the factor $\mathbb{Z}/D\mathbb{Z}$.

Write coordinates of G as $(x, z, (v_i, c_i)_i)$, and project to the ***base*** $(x, k) \in \mathbb{Z}^2 \times K$ by setting $k_i = v_i \bmod r_i$. We call $x \in \mathbb{Z}^2$ the ***horizontal coordinates***. The k_i are the ***low coordinates***; $c_i \in P_i$ are the ***useful outputs***, and $z \in \mathbb{Z}/D\mathbb{Z}$ is the ***auxiliary output***. A ***fiber*** over a base point is the set of all points projecting to it; a ***lift*** of a base point is any point in that fiber. The kernel of this projection is the finite subgroup

$$H_0 = \{0\} \times (\mathbb{Z}/D\mathbb{Z}) \times \prod_i \big(r_i(\mathbb{Z}/r_i^2\mathbb{Z}) \times P_i\big).$$

Our first equation is

$$A \oplus H_0 = G. \tag{3.4}$$

It says exactly that A has one point over each base point (x, k). We call such an A a ***graph over the base***; it is specified by its output functions z, c_i, v_i. For a residue t modulo a positive integer r, write $[t]_r$ for its least nonnegative integer representative. It is useful to replace v_i by an arbitrary function $\beta_i : \mathbb{Z}^2 \times K \to K_i$, called its ***high output***, via

$$v_i(x, k) = x_1 + [k_i - x_1]_{r_i} + r_i\beta_i(x, k) \quad \text{in } \mathbb{Z}/r_i^2\mathbb{Z}. \tag{3.5}$$

This parametrizes the possible lifts of k_i bijectively. It is a choice of coordinates on fibers, not a splitting of cyclic groups. The graph characterization is `Compiler.kernelTile_iff_graph` in `Compiler/GraphTests.lean`.

3.2 Controlling Dependence

We next require functions $\widetilde{c}_n : \mathbb{Z} \times K_n \to P_n$ and $\widetilde{c}_{\eta_t} : S \times K_{\eta_t} \to P_{\eta_t}$ such that

$$c_n(x,k) = \widetilde{c}_n(L_n(x), k_n), \qquad c_{\eta_t}(x,k) = \widetilde{c}_{\eta_t}(s(x), k_{\eta_t}). \tag{3.6}$$

These will not restrict z or any β_i. The dependence gadget is [Ope26, Lemma 3.1], with our decorated channel coordinates.

Lemma 3.1. *For a channel i and a base shift $\sigma \in \mathbb{Z}^2 \times K$, a finite prototile can impose $c_i(b-\sigma) = c_i(b)$ for every base point $b = (x,k)$ on graphs satisfying* (3.4)*, without restricting their other outputs.*

Proof. Let $\mathrm{pr}_i : G \to P_i$ be the coordinate projection. Partition H_0 into $E_0 = \{h : \mathrm{pr}_i(h) = 0\}$ and $E_* = H_0 \setminus E_0$. Choose a lift g of σ with $\mathrm{pr}_i(g) = 0$, and use $F = E_* \sqcup (g + E_0)$. The union is disjoint by its c_i coordinates. At base b, the first part covers all slices of the fiber except the one whose ith useful output is $c_i(b)$. The second part covers exactly the slice with ith useful output $c_i(b-\sigma)$. Every other coordinate is filled freely and once in either contribution. Exact coverage is therefore equivalent to the stated identity. This is `Compiler.invarianceTile_iff` in `Compiler/GraphTests.lean`. □

Apply this lemma to generators of K_ℓ for every $\ell \neq i$, then to the horizontal shift u_n in ordinary channel n, and to the two shifts $(M,0), (0,M)$ in each seed channel. These finitely many equations are equivalent to (3.6). Their combined equivalence is `Compiler.dependenceTiles_iff` in `Compiler/Dependence.lean`.

For a graph with these dependencies, put $g_{i,x}(k_i) = c_i(x,k)$. Following [Ope26, Section 3.3], define its ***active labels*** by

$$\mathcal{A}_i(x) = \{j \in J_i : \exists v, v' \in K_i,\ v - v' \in U_{ij},\ g_{i,x}(v) \neq g_{i,x}(v')\}. \tag{3.7}$$

A label is ***inactive*** when the function ignores that input digit. A channel is ***active*** at x if $\mathcal{A}_i(x) \neq \varnothing$, and ***inactive*** at x otherwise. If all labels are inactive, the function is constant: change the digits one at a time between any two inputs. For ordinary n, the active set depends only on $L_n(x)$, and for a seed it depends only on $s(x)$.

3.3 Excluding a Tuple of Active Labels

The next test is the cycle construction of [Ope26, Lemma 3.2].

Lemma 3.2. *Let C be a nonempty cyclically ordered list of distinct channels, and choose one label $j(i) \in J_i$ for each $i \in C$. A finite family of finite prototiles imposes the condition that these labels are not all active at the same x. On graphs with* (3.6)*, this condition is equivalent to the family of tiling equations, regardless of z and the high outputs.*

Proof. Here $\mathrm{prev}(i)$ is the preceding channel in the chosen cyclic order, including the wrap from the first channel to the last. For every choice of maps $d_i : P_{\mathrm{prev}(i)} \to U_{i,j(i)}$, define $d_i[e] = d_i(e_{\mathrm{prev}(i)})$ on the cycle and $d_i[e] = 0$ off the cycle. Include the prototile

$$F_d = \{(0, \zeta, (w_i, e_i)_i) : e \in P,\ \zeta \in \mathbb{Z}/D\mathbb{Z},\ w_i \bmod r_i = d_i[e]\}.$$

Each w_i ranges over all lifts of its specified low residue. For a target base (x,k), the useful-output map is

$$e \longmapsto \big(e_i + g_{i,x}(k_i - d_i[e])\big)_i. \tag{3.8}$$

The remaining output offsets have unique choices after e is fixed. Thus $A \oplus F_d = G$ precisely when this map is a permutation for every (x,k).

If the chosen label at i_0 is inactive, its component in (3.8) becomes $e_{i_0} + g_{i_0,x}(k_{i_0})$. It determines e_{i_0} uniquely from the target. Successive components around the cycle then determine all other e_i uniquely; the initial component has no dependency left to check. Off-cycle components are translations and can be solved independently.

If all chosen labels are active at some x, choose witnesses v_i, v_i' with $v_i - v_i' \in U_{i,j(i)}$ and $\epsilon_i = g_{i,x}(v_i) - g_{i,x}(v_i') \neq 0$. Take $k_i = v_i$ on the cycle and choose maps with

$$d_i(0) = 0, \qquad d_i(\epsilon_{\mathrm{prev}(i)}) = v_i - v_i'.$$

The arguments are distinct, so such maps exist and are among those tested. Extend ϵ by zero off the cycle. The inputs 0 and ϵ have the same image in (3.8), a contradiction to bijectivity. All maps and all prototiles used are finite. The equivalence is `Compiler.cycle_exclusion_iff` in `Compiler/CycleTests.lean`. □

Figure 2 illustrates the dependency cycle in this argument.

Apply Lemma 3.2 to every word outside $\mathcal{W}$, using the cycle of all ordinary channels. Also apply it to the pair (η_t, n) with labels $*$ and j, for every $j \in \Sigma \setminus B_t$. The resulting equations are exactly the conditions

$$\prod_{n \in I} \mathcal{A}_n(x) \subseteq \mathcal{W}, \qquad * \in \mathcal{A}_{\eta_t}(x) \implies \mathcal{A}_n(x) \subseteq B_t \quad (n \in I). \tag{3.9}$$

These conditions allow empty active sets. The next section excludes that degeneracy where it matters.

4 Activation and Soundness

Activation tests are tiling equations designed to force nonempty active sets. We adapt the activation tests of [Ope26, Section 4] to the residue set S in (3.1). Their purpose is to make every ordinary active set nonempty and each seed active somewhere.

4.1 A Nonuniform List with Uniform Marginals

On P_i, let $\mathbf{1}_e$ be the function equal to one at e and zero elsewhere, and let 1 denote the constant function equal to one. For each channel i, define

$$\omega_i = \mathbf{1}_{(0,0)} - \mathbf{1}_{(1,0)} - \mathbf{1}_{(0,1)} + \mathbf{1}_{(1,1)}, \qquad \mu_i = 1 + \omega_i.$$



(a) All selected labels active

(b) Selected label at 1 inactive

Figure 2: A three-channel instance of the cycle test, adapted from [Ope26, Lemma 3.2]. Arrows indicate dependence of an output component on another input in (3.8). (a) When all selected labels are active, some tested maps produce a collision. (b) An inactive selected label removes one dependency, allowing the components of the preimage to be recovered in order for every choice of the maps.

We will use $\mu_i(e)$ as the ***multiplicity***, or number of occurrences, of e in a list. These multiplicities are nonnegative and not all equal. Their ***marginals***, the sums over either coordinate, are uniform:

$$\sum_{\alpha\in\mathbb{Z}/a_i\mathbb{Z}} \mu_i(\alpha,\xi) = a_i, \qquad \sum_{\xi\in\mathbb{Z}/b_i\mathbb{Z}} \mu_i(\alpha,\xi) = b_i.$$

Let Δ_i index a list $(e_\delta)_{\delta\in\Delta_i}$ of length a_ib_i having these multiplicities, with $e_\delta = (e^a_\delta, e^b_\delta)$. For each fixed ξ, choose a bijection from the indices with $e^b_\delta = \xi$ to $\mathbb{Z}/a_i\mathbb{Z}$, and express it as $\delta \mapsto e^a_\delta + \rho^a_i(\delta)$. Independently, for each fixed α, choose a bijection from the indices with $e^a_\delta = \alpha$ to $\mathbb{Z}/b_i\mathbb{Z}$, expressed as $\delta \mapsto e^b_\delta + \rho^b_i(\delta)$. Thus both maps

$$\begin{aligned} \{\delta\colon e^b_\delta = \xi\} &\longrightarrow \mathbb{Z}/a_i\mathbb{Z}, \quad \delta \longmapsto e^a_\delta + \rho^a_i(\delta), \\ \{\delta\colon e^a_\delta = \alpha\} &\longrightarrow \mathbb{Z}/b_i\mathbb{Z}, \quad \delta \longmapsto e^b_\delta + \rho^b_i(\delta) \end{aligned} \tag{4.1}$$

are bijections. Repeated entries in the list have different indices.

Figure 3 gives a small example of this list on $(\mathbb{Z}/2\mathbb{Z})\times(\mathbb{Z}/3\mathbb{Z})$. Only the multiplicity pattern is illustrated; the prime choices in the full encoding remain those specified in Section 3.

4.2 Finite Activation Prototiles

Given a horizontal offset $h = (h_1, h_2)$, a useful-output offset $e \in P_i$, and a set $E \subseteq \mathbb{Z}/D\mathbb{Z}$, define a ***batch of offsets*** by

$$\begin{aligned} \mathcal{B}_i(h,e;E) = \{(h,\zeta,(w_\ell,d_\ell)_\ell) : \zeta\in E,\ w_i = h_1 \pmod{r_i^2},\ d_i = e, \\ w_\ell \in r_\ell(\mathbb{Z}/r_\ell^2\mathbb{Z}),\ d_\ell \in P_\ell \quad (\ell\neq i)\}. \end{aligned} \tag{4.2}$$

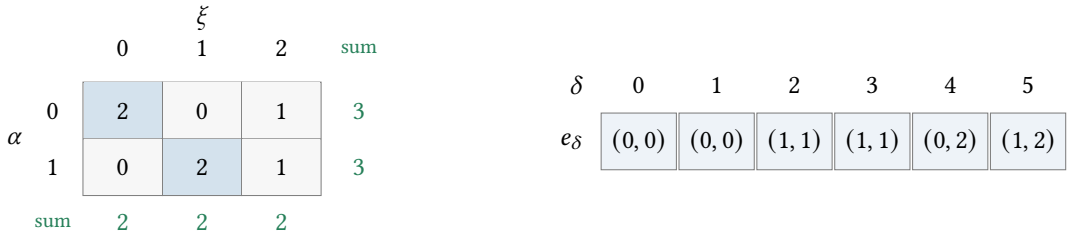
All unrestricted coordinates vary independently.

For an ordinary channel i, choose, for every $(l,\delta)\in K_i\times\Delta_i$, an integer multiple $h = h^{(i)}_{l,\delta}$ of u_i satisfying

$$h_1 \equiv l \pmod{r_i}, \qquad h_1 \equiv \rho^a_i(\delta) \pmod{a_i}, \qquad h_1 \equiv \rho^b_i(\delta) \pmod{b_i}. \tag{4.3}$$

Choose these horizontal offsets distinctly, and define

$$F^{\mathrm{act}}_i = \bigcup_{l,\delta} \mathcal{B}_i(h^{(i)}_{l,\delta}, e_\delta; \mathbb{Z}/D\mathbb{Z}).$$



(a) Multiplicities $\mu(\alpha,\xi)$ (b) Six distinct list indices

Figure 3: The activation multiplicity pattern on $(\mathbb{Z}/2\mathbb{Z})\times(\mathbb{Z}/3\mathbb{Z})$, adapted from [Ope26, Section 4]. (a) The entries are nonuniform, but their marginals are uniform. (b) The same multiplicities written as an indexed list. Separate horizontal offsets for repeated entries make the activation prototile a set. An inactive ordinary channel cannot satisfy its required uniform coverage, whereas the marginal bijections (4.1) support the block inverse in Lemma 5.1.

The Chinese remainder theorem gives infinitely many choices for each h_1, and then $h_2 = -ih_1$. In particular, $L_i(h) = 0$.

For a seed i, let i' denote the other seed. Choose distinct offsets $h = h^{(i)}_{l,\delta,\tau}$, indexed by $K_i \times \Delta_i \times S$, satisfying (4.3) and

$$s(h) = \tau, \qquad h_1 \equiv 0 \pmod{a_{i'}}. \tag{4.4}$$

The moduli $r_i, a_i, b_i, M, a_{i'}$ are pairwise coprime, so these choices are again possible. Put

$$F_i^{\mathrm{act}} = \bigcup_{l,\delta,\tau} \mathcal{B}_i(h^{(i)}_{l,\delta,\tau}, e_\delta; \{0\}).$$

For ordinary channels the entire z-coordinate is available in each batch; for seeds the z-offset is zero. Distinct horizontal offsets make the batches disjoint, even when their e_δ values coincide. Thus these are sets of offsets, not weighted prototiles. Each has cardinality

$$|F_i^{\mathrm{act}}| = D \prod_{\ell} r_\ell |P_\ell| = |H_0|.$$

The activation equation for channel i is $A \oplus F_i^{\mathrm{act}} = G$.

Lemma 4.1 (Activation criterion)**.** *Fix a graph satisfying* (3.4) *and a target base* (x, k)*. For a batch indexed by* l *and with horizontal offset* h*, let*

$$x' = x - h, \qquad k_i' = k_i - l, \qquad k_\ell' = k_\ell \quad (\ell \neq i).$$

The ordinary activation equation is equivalent to bijectivity of

$$(l, \delta) \longmapsto \big(c_i(x', k') + e_\delta, \beta_i(x', k')\big) \tag{4.5}$$

from $K_i \times \Delta_i$ *to* $P_i \times K_i$*, at every target base. For a seed the corresponding condition is bijectivity of*

$$(l, \delta, \tau) \longmapsto \big(c_i(x', k') + e_\delta, \beta_i(x', k'), z(x', k')\big) \tag{4.6}$$

onto $P_i \times K_i \times \mathbb{Z}/D\mathbb{Z}$ *at every target base.*

Proof. We adapt the fiber calculation of [Ope26, Lemma 4.1]. The base of the source graph point is forced by the batch offsets. Since $h_1 \equiv l \pmod{r_i}$, we have $[k_i' - x_1']_{r_i} = [k_i - x_1]_{r_i}$. After adding the offset h_1 to v_i, formula (3.5) gives

$$v_i(x', k') + h_1 = x_1 + [k_i - x_1]_{r_i} + r_i \beta_i(x', k').$$

Hence the batch preserves the sheared high output and adds e_δ to the useful output. All other high and useful coordinates are filled freely and exactly once. The same holds for z in an ordinary batch, whereas a seed batch retains the source value of z. Exact coverage is therefore precisely the stated bijection condition. These two characterizations are `Compiler.ordinaryActivationTile_iff` and `Compiler.seedActivationTile_iff` in `Compiler/ActivationTiles.lean`. □

Lemma 4.2. *In every common solution of the graph, dependence, and activation equations,* $\mathcal{A}_n(x) \neq \varnothing$ *for every ordinary channel* n *and every* x*. Each seed has a nonempty active set at some* x*.*

Proof. The two counting obstructions below adapt [Ope26, Lemmas 4.2–4.3] to our two-prime residue set. Suppose an ordinary channel i is inactive at x. Its useful function of k_i is a constant c_0. Every source of its activation test has the same value of L_i, so also has useful output c_0. The number of batches producing

useful output c is $r_i\mu_i(c - c_0)$. Bijectivity in (4.5) requires this number to be r_i for every c, contrary to nonuniformity of μ_i.

Now suppose a seed i is nowhere active. Its useful output then has the form $q_i(s(x))$, independent of k_i. Define the ***histogram***

$$Q(\alpha, \xi) = |\{s \in S : q_i(s) = (\alpha, \xi)\}|, \qquad \sum Q = D.$$

For functions on P_i, define their ***convolution*** by $(f * g)(c) = \sum_{u \in P_i} f(u)g(c - u)$. At any target base, the pairs $(s(x) - \tau, k_i - l)$ run through $S \times K_i$. Counting only the useful output in (4.6) yields $Q * \mu_i = D$, with the right side the constant function on P_i. Since $Q * 1 = D$, we have $Q * \omega_i = 0$:

$$Q(\alpha, \xi) - Q(\alpha - 1, \xi) - Q(\alpha, \xi - 1) + Q(\alpha - 1, \xi - 1) = 0.$$

Thus $Q(\alpha, \xi) - Q(\alpha - 1, \xi)$ is independent of ξ. Choose α_0 minimizing $Q(\alpha, 0)$, and set

$$u(\alpha) = Q(\alpha, 0) - Q(\alpha_0, 0), \qquad v(\xi) = Q(\alpha_0, \xi).$$

Summing the differences along the first coordinate gives $Q(\alpha, \xi) = u(\alpha) + v(\xi)$, with u, v nonnegative integers. Consequently

$$D = b_i \sum_{\alpha} u(\alpha) + a_i \sum_{\xi} v(\xi).$$

Because $b_i > D$, the first sum is zero, so a_i divides D. But a_i is 2 or 3, and $D = (5pq)^2$ is divisible by neither. This proves that each seed must activate somewhere. The two conclusions are `Compiler.ordinary_active_of_tile` and `Compiler.seed_active_somewhere_of_tile` in `Compiler/Activation.lean`. □

Let $\mathscr{F}$ consist of H_0, all dependence prototiles, all exclusion prototiles, and all activation prototiles. This finite family is computable from the domino system. Its size depends on the input Wang tile set: the exclusion tests range over forbidden words in the input-dependent alphabet Σ.

Proposition 4.3 (Soundness)**.** *If the equations $A \oplus F = G$, $F \in \mathscr{F}$, have a common solution, then the input domino system tiles $\mathbb{Z}^2$.*

Proof. Fix any total order on Σ and define

$$W(n, m) = \min \mathcal{A}_n((0, m)).$$

The minimum exists by Lemma 4.2. Dependence on L_n means that $W(n, L_n(x))$ belongs to $\mathcal{A}_n(x)$. Thus the first exclusion in (3.9) proves the line rule at every $x = (d, e)$.

For $t = 1, 2$ choose $x^{(t)}$ where seed η_t is active. The second exclusion gives

$$W(n, L_n(x^{(t)})) \in B_t \qquad (n \in I).$$

Both integer components therefore take the distinct values 1 and 2 in every column. Theorem 2.1 gives a domino solution on $\mathbb{N}^2$, and compactness gives one on $\mathbb{Z}^2$. This implication is `Compiler.wang_of_solves` in `Compiler/Soundness.lean`. □

5 One Common Solution

We prove the converse by lifting the canonical array from (2.3). The construction uses the block permutations of [Ope26, Section 5], with the two-prime residue set and the relaxed seed restrictions B_t.

5.1 Block Functions

For channel i and label j, partition the jth digit set $\mathbb{Z}/r_{ij}\mathbb{Z}$ into A_{ij} sets of size a_i and B_{ij} sets of size b_i, using (3.2). Label each such set bijectively by the cyclic group of its size. Copy these partitions and labels identically for every setting of the other digits in K_i. This partitions K_i into ***blocks***, each fixing all digits other than j.

If k_i has label y in a block of size d, define

$$C_{i,j}(k_i) = \begin{cases} (y, 0), & d = a_i, \\ (0, y), & d = b_i. \end{cases}$$

Let $T_{i,j}(t)$ be the permutation that adds t to the label in each block, preserving the block itself. These are permutations of sets; the blocks need not be subgroups. The function $C_{i,j}$ depends on exactly the jth digit, so its active set is $\{j\}$.

Lemma 5.1 (Block inverse)**.** *Suppose integers $h_1(l, \delta)$ satisfy* (4.3)*. For any fixed i, j, x_1, k_i, the map*

$$(l, \delta) \longmapsto \big(C_{i,j}(k_i - l) + e_\delta, T_{i,j}(x_1 - h_1(l, \delta))(k_i - l)\big) \tag{5.1}$$

is a bijection from $K_i \times \Delta_i$ to $P_i \times K_i$.

Proof. This is the block calculation of [Ope26, Lemma 5.1]. Fix a target $((\alpha^*, \xi^*), \beta^*)$. The point β^* identifies a block and a label z_0 in that block. Any preimage of this target must have $k_i' = k_i - l$ in that block.

If its size is a_i, the target equations force

$$e^b_\delta = \xi^*, \qquad e^a_\delta + \rho^a_i(\delta) = \alpha^* + x_1 - z_0 \quad \text{in } \mathbb{Z}/a_i\mathbb{Z}. \tag{5.2}$$

The first bijection in (4.1) determines δ. The source label is $y = \alpha^* - e^a_\delta$, which fixes k_i', and then $l = k_i - k_i'$. The high label is correct because

$$y + x_1 - h_1(l, \delta) = \alpha^* + x_1 - (e^a_\delta + \rho^a_i(\delta)) = z_0.$$

These necessary choices also suffice.

For a block of size b_i, use instead

$$e^a_\delta = \alpha^*, \qquad e^b_\delta + \rho^b_i(\delta) = \xi^* + x_1 - z_0 \quad \text{in } \mathbb{Z}/b_i\mathbb{Z}. \tag{5.3}$$

The second bijection fixes δ; the source label is $\xi^* - e^b_\delta$, and again k_i' and l follow uniquely. The explicit inverse is packaged as the equivalence `Compiler.BlockShiftData.blockInverseEquiv` in `Compiler/BlockActivationInverse.lean`. □

5.2 Ordinary Outputs

Suppose the domino system has a solution, and let W be the canonical array of Theorem 2.1. For ordinary channel i set

$$j_i(x) = W(i, L_i(x)), \qquad c_i(x, k) = C_{i,j_i(x)}(k_i), \qquad \beta_i(x, k) = T_{i,j_i(x)}(x_1)(k_i). \tag{5.4}$$

The active set is exactly $\{j_i(x)\}$, and the useful output has the required dependence. Every activation offset for this channel satisfies $L_i(h) = 0$, so all sources in its test use the same label $j_i(x)$ and the same block partition. The test map is therefore (5.1). Lemmas 4.1 and 5.1 verify every ordinary activation equation. This calculation permits any valid Sudoku array; the canonical choice will only be needed to accommodate the seeds.

5.3 Seed Regions and the Shared Output

For $t = 1, 2$, choose disjoint sets $S_t \subset S$ of respective sizes 3 and 4, contained in

$$\{(s_1, s_2) \in S : s_1 \equiv 0,\ s_2 \equiv t \pmod{pq}\}. \tag{5.5}$$

Each eligible residue class has 25 points, so the choices are possible. Let $S_0 = S \setminus (S_1 \cup S_2)$. Since $D = (5pq)^2 \equiv 1 \pmod 6$, the number $D - 7$ is divisible by 6. Equip these regions with the following set-theoretic label spaces:

region	label space	number of copies
S_1	$\mathbb{Z}/3\mathbb{Z}$	1
S_2	$\mathbb{Z}/2\mathbb{Z}$	2
S_0	$(\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/3\mathbb{Z})$	$(D-7)/6$.

The labels on S_1 belong to seed η_2, those on S_2 to seed η_1, and those on S_0 to both seeds. Wherever the label for seed i is present, denote its coordinate in $\mathbb{Z}/a_i\mathbb{Z}$ by $y_i(s)$. These will be precisely the regions where that seed is inactive. A point s is specified by its region, its copy of the label space, and the labels of the inactive seeds.

Let $R(t)$ be the permutation of S preserving each label-space copy and adding t to all its coordinates. Fix any bijection $\lambda\colon S \to \mathbb{Z}/D\mathbb{Z}$, and define a single auxiliary output

$$z(x, k) = \lambda(R(x_1)s(x)). \tag{5.6}$$

There is no requirement that these labelings respect the additive structure of S; that structure is used only for $s(x - h) = s(x) - s(h)$.

For seed $i = \eta_t$, abbreviate $C_{i,*}$ and $T_{i,*}$ by C_i, T_i, and put

$$(c_i(x, k), \beta_i(x, k)) = \begin{cases} (C_i(k_i), T_i(x_1)(k_i)), & s(x) \in S_t, \\ ((y_i(s(x)), 0), k_i), & s(x) \notin S_t. \end{cases} \tag{5.7}$$

This seed is active precisely on S_t. Its useful output depends only on $s(x)$ and k_i, as required. Its high output need not have this dependence.

Lemma 5.2. *With Equations* (5.6) *and* (5.7)*, both seed activation equations hold for the same graph outputs.*

Proof. Fix a seed $i = \eta_t$, its companion i', and a target base (x, k). A source batch has residue $s' = s(x) - \tau$, low coordinate $k_i' = k_i - l$, and first horizontal coordinate $x_1 - h_1$. Fix a target (c_i^*, β_i^*, z^*), with $c_i^* = (\alpha^*, \xi^*)$, and put $s'' = \lambda^{-1}(z^*)$. The equation for z is

$$s'' = R(x_1 - h_1)s'. \tag{5.8}$$

It forces s' to have the same region and label-space copy as s''.

Suppose first that $s'' \in S_t$, so this seed is active at the source. The only label belongs to i'. Because $h_1 \equiv 0 \pmod{a_{i'}}$, it satisfies $y_{i'}(s') = y_{i'}(s'') - x_1$. This uniquely determines s' and $\tau = s(x) - s'$ before l or δ has been chosen. For this fixed τ, apply Lemma 5.1 to the offsets $h_1(l, \delta, \tau)$. It gives unique l, δ producing (c_i^*, β_i^*). Their congruences preserve the already recovered value of z.

Suppose instead that $s'' \notin S_t$, so this seed is inactive at the source. The high output first forces $k_i' = \beta_i^*$ and $l = k_i - \beta_i^*$. Let $z_0 = y_i(s'')$. The useful-output and z equations imply

$$e_\delta^b = \xi^*, \qquad y_i(s') = \alpha^* - e_\delta^a, \qquad z_0 = y_i(s') + x_1 - \rho_i^a(\delta).$$

Thus (5.2) determines δ uniquely, and the second identity determines $y_i(s')$. If i' is also inactive, its label is determined by $y_{i'}(s') = y_{i'}(s'') - x_1$; otherwise it has no label in this region. Together with the known copy of the label space, these labels specify exactly one s', hence exactly one τ.

In both cases the recovered indices determine a batch whose actual offset satisfies every congruence used in the recovery. There is no further condition on its integer lift. Every target therefore has exactly one preimage in (4.6). The calculation works for either choice of i, using the same function z. The representing batch may differ between the two equations, as is permitted for a common set of translations. The simultaneous construction is checked by `Compiler.seedActualForward_bijective` in `Compiler/CommonSeedOutputs.lean`. □

Proposition 5.3 (Completeness)**.** *If the input domino system tiles $\mathbb{Z}^2$, then the equations $A \oplus F = G$, $F \in \mathscr{F}$, have a common solution.*

Proof. Restrict a domino solution to $\mathbb{N}^2$ and use its canonical Sudoku array. Equations (5.4), (5.6) and (5.7) define all useful, high, and auxiliary outputs of one graph. Equation (3.5) supplies its actual v_i coordinates, so (3.4) holds.

The useful outputs have the required dependencies. Lemmas 5.1 and 5.2 prove every activation equation. At each x, the singleton ordinary active sets form the allowed word $(W(n, L_n(x)))_{n\in I}$, so all word exclusions hold. When seed η_t is active, (5.5) and (2.5) put every ordinary active label in B_t. Thus all seed exclusions hold as well. Lemma 3.2 verifies their tiling equations on this same graph. In Lean, `Compiler.CommonModel.common_solution` in `Compiler/CommonSolution.lean` verifies the graph, and `Compiler.exists_solution_of_wang` in `Compiler/Completeness.lean` constructs it from a Wang tiling. □

Theorem 5.4. *There is an algorithm taking a domino system to a finite cyclic group V and a finite family of prototiles $\mathscr{F}$ in $\mathbb{Z}^2 \times V$ such that the system has a domino solution if and only if there is one set of translations A satisfying $A \oplus F = \mathbb{Z}^2 \times V$ for every $F \in \mathscr{F}$.*

Proof. Soundness and completeness are Propositions 4.3 and 5.3. All choices in the construction concern finite sets, maps, prime searches, or Chinese remainder calculations. They can be made by fixed enumeration orders. In particular, no knowledge of a domino solution is needed to construct the prototiles. The infinite solution is used only to prove completeness. The semantic equivalence is `Compiler.finite_system_iff_wang` in `Compiler/Completeness.lean`; the effective finite family is verified by `Compiler.Effective.family_tiling_iff` in `Compiler/EffectiveSystem.lean`. □

6 A Single Prototile in Three Dimensions

6.1 Stacking the Prototiles

The stacking step is an existence equivalence, not merely a way to transfer aperiodicity. We record its proof to make that distinction explicit. The construction originates in [GT24, Theorem 3.1 and Lemma 3.2], following earlier consolidation results in [GT23, Theorem 1.15 and Section 3]. We use the prime-cyclic version in [Ope26, Section 6].

Lemma 6.1. *For every positive integer s and finite set of forbidden primes, one can effectively find a prime ℓ outside that set and a partition $\mathbb{Z}/\ell\mathbb{Z} = E_1 \sqcup \cdots \sqcup E_s$ with $E_\nu - E_\nu = \mathbb{Z}/\ell\mathbb{Z}$ for every ν.*

Proof. Color the elements of $\mathbb{Z}/\ell\mathbb{Z}$ independently and uniformly with s colors. For any nonzero d, the cycle $0, d, 2d, \dots, (\ell-1)d$ contains $\lfloor \ell/2 \rfloor$ disjoint consecutive pairs. A specified pair receives color ν at both ends with probability s^{-2}. The probability that some color fails to realize some nonzero difference is at most

$$s(\ell-1)(1-s^{-2})^{\lfloor \ell/2 \rfloor}.$$

Choose a fresh odd prime large enough that this is less than one, and search the finite set of colorings. A successful coloring exists; its color classes are nonempty and realize zero as well. This is `Stacking.exists_fresh_prime_fullDifference_partition` in `Proofs/FullDifferencePartition.lean`. □

Lemma 6.2. *Let $F_1, \dots, F_s$ be prototiles in an abelian group H, and choose a partition as in Lemma 6.1. Set*

$$F^{\mathrm{st}} = \bigcup_{\nu=1}^{s} (F_\nu \times E_\nu).$$

Then F^{st} tiles $H \times \mathbb{Z}/\ell\mathbb{Z}$ if and only if one set of translations A satisfies $A \oplus F_\nu = H$ for all ν.

Proof. A common solution A gives the stacked solution $A \times \{0\}$. Conversely, let $B \oplus F^{\mathrm{st}} = H \times \mathbb{Z}/\ell\mathbb{Z}$. At a fixed horizontal point $g \in H$, a color-ν contribution consists of $(a, t) \in B$ and $f \in F_\nu$ with $a + f = g$; it covers $t + E_\nu$ in that fiber. Two distinct contributions of the same color would overlap, because $E_\nu - E_\nu = \mathbb{Z}/\ell\mathbb{Z}$. Thus there is at most one per color. If $n_\nu \in \{0, 1\}$ counts them, exact coverage implies

$$\sum_\nu n_\nu |E_\nu| = \ell = \sum_\nu |E_\nu|.$$

Every E_ν is nonempty, so all n_ν are one.

Let $\mathrm{pr}\colon H \times \mathbb{Z}/\ell\mathbb{Z} \to H$ be the coordinate projection. This projection is injective on B: two points $(a, t), (a, t')$ would give two color-1 contributions at $a + f$ for any $f \in F_1$. Consequently $A = \mathrm{pr}(B)$ has exactly one representation of every g with each F_ν, as required. The equivalence is `Stacking.exists_tiles_stack_iff` in `Proofs/Stacking.lean`. □

Corollary 6.3. *From a domino system one can compute a positive integer m and a prototile $F \subset \mathbb{Z}^2 \times \mathbb{Z}/m\mathbb{Z}$ that tiles the whole group if and only if the domino system is solvable.*

Proof. Apply Lemma 6.2 to Theorem 5.4, choosing ℓ coprime to $|V|$. The group $V \times \mathbb{Z}/\ell\mathbb{Z}$ is cyclic of order $m = \ell|V|$. Thus stacking enlarges the modulus while keeping the finite coordinate cyclic. All steps are effective. The semantic construction is `Compiler.exists_cyclic_tile` in `Compiler/Consolidation.lean`; `Compiler.cyclicCompiler_computable` and `Compiler.cyclicCompiler_correct` in `Compiler/Main.lean` verify the effective version. □

6.2 Removing the Finite Coordinate

We apply the quotient-to-lattice reduction of [MSS25, Theorem 1.1] in the special case needed here. We define its component shapes and prove their rigidity before assembling the lattice prototile.

Put

$$\Gamma = \mathbb{Z}^2 \times \mathbb{Z}/m\mathbb{Z}, \qquad \pi(x_1, x_2, x_3) = (x_1, x_2, x_3 \bmod m), \qquad w = (0, 0, m).$$

The kernel is $L = \mathbb{Z}w$. A ***fundamental domain*** for L is a set containing exactly one representative of every coset of L; we use $D_m = \mathbb{Z}^2 \times \{0, \dots, m-1\}$. Thus Γ is the quotient of $\mathbb{Z}^3$ that identifies (x_1, x_2, x_3) with $(x_1, x_2, x_3 + m)$.

6.2.1 Rigid Component Shapes

Let e_1, e_2, e_3 be the standard unit vectors in $\mathbb{Z}^3$, and define the ***boxes*** and their ***one-cell-thick frames*** by

$$B_k = \{-k, \dots, k\}^3, \qquad H_k = B_k \setminus B_{k-1} \quad (k \geq 1).$$

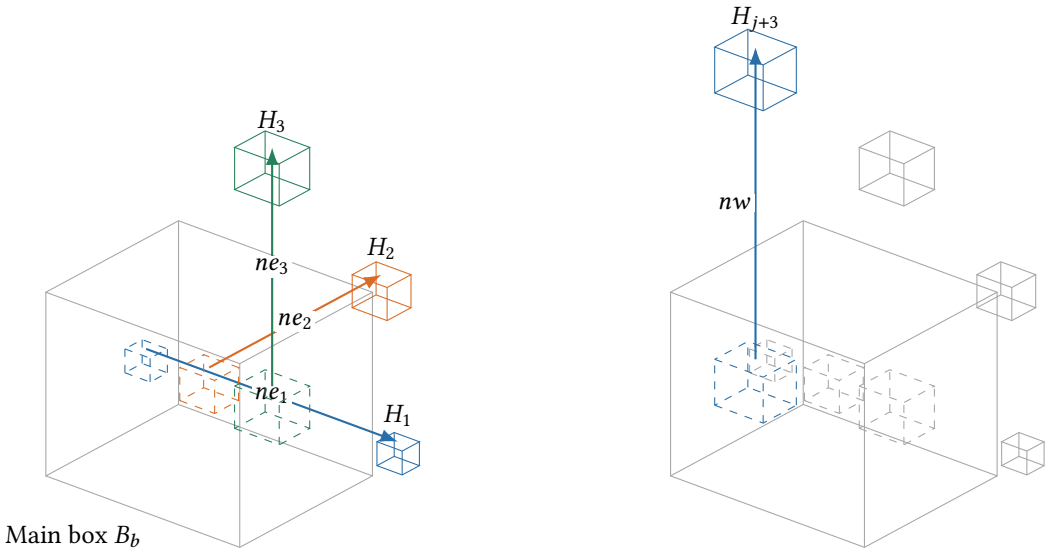
Take $b = 100$, $n = 2b + 1 = 201$, and $v_j = (12j, 0, 0)$ for $1 \le j \le 5$. The five boxes $v_j + B_5$ are disjoint, lie strictly inside B_b, and avoid zero. Define the following shapes, illustrated in Figure 4:

$$\begin{aligned} T &= \left(B_b \setminus \bigcup_{j=1}^{3} (v_j + H_j) \right) \sqcup \bigcup_{j=1}^{3} (ne_j + v_j + H_j), \\ T_1 &= (T \setminus (v_4 + H_4)) \sqcup (nw + v_4 + H_4), \\ T_2 &= (T \setminus (v_5 + H_5)) \sqcup (nw + v_5 + H_5). \end{aligned} \tag{6.1}$$

Thus each removed frame is reattached at a displacement that is a multiple of n. The first three frames distinguish the coordinate directions; the extra frame distinguishes the type 1 or 2 and links copies separated by nw. The removed frames form ***dents***, and the reattached frames form matching ***bumps***. These are finite subsets of the lattice, including their detached bumps.

Each shape has exactly one point in every residue class modulo $n\mathbb{Z}^3$: it is obtained from the box B_b by moving selected points by vectors in $n\mathbb{Z}^3$. Moreover, the representatives in T_1 and T_2 of any given residue differ by an element of nL, because their type-specific moves are by nw. These observations are direct checks on the shapes; forcing every tiling to respect their intended placement is the content of the next lemma. By a ***mixed tiling*** with prototiles T_1, T_2 and sets of translations C_1, C_2 we mean that all copies $c + T_j$, $c \in C_j$, $j = 1, 2$, together cover $\mathbb{Z}^3$ exactly once.

Lemma 6.4 (Rigid components, Meyerovitch–Sanadhya–Solomon)**.** *For the shapes in* (6.1)*, let* $C_1, C_2 \subset \mathbb{Z}^3$



(a) Base shape T: move three frames

(b) Component T_j: move one extra frame

Figure 4: The rigid component shapes in (6.1), shown schematically with frame sizes and separations exaggerated. Colored wireframes indicate hollow, one-cell-thick shells, not solid boxes. Dashed frames are removed; solid frames are reattached at the indicated displacements. (a) Three different frame sizes distinguish the coordinate directions. (b) For $j = 1, 2$, moving the extra frame H_{j+3} distinguishes T_1 from T_2 and enforces repetition along $w = (0, 0, m)$; the rest of T is unchanged. The lattice prototile $\widetilde{F}$ is assembled from translated copies of T_1, T_2 as in (6.2).

with $0 \in C_1 \cup C_2$. *The copies* $c + T_j$, $c \in C_j$, $j = 1, 2$, *cover* $\mathbb{Z}^3$ *exactly once if and only if*

$$C_1 \sqcup C_2 = n\mathbb{Z}^3, \qquad C_j + nw = C_j \quad (j = 1, 2).$$

This is [MSS25, Lemma 2.1, its construction and part (c)] with $d = 3$, two component types, and the rank-one subgroup $L = \mathbb{Z}w$. The normalization $0 \in C_1 \cup C_2$ is harmless: translate any tiling so that one of its component translation vectors is zero. Here invariance under nL means exactly $C_j + nw = C_j$; it imposes no periods in the first two coordinates. The matching bumps and dents force alignment to the scaled grid $n\mathbb{Z}^3$ and repetition every m grid steps in the third coordinate.

The local forcing argument in the proof of the more general claim [MSS25, Lemma 2.1] has a gap. For the base shape T above, T and $(-214, -1, 200) + T$ are disjoint, yet the latter covers $ne_3 = (0, 0, 201)$ via the point $(214, 1, 1) \in T$ in a detached bump, despite not being translated by ne_3. Figure 5 shows this placement and magnifies the covering bump. The Lean theorem `MSS.unintended_neighbor` in `MSS/LocalForcingCheck.lean` verifies this counterexample to the local claim, which concerns a partial placement. The following proof of Lemma 6.4 uses further coverage constraints; its Lean formalization culminates in `MSS.rigidity` in `MSS/GridAssembly.lean`.

Proof of Lemma 6.4. Suppose first that the copies form a mixed tiling, and put $C = C_1 \cup C_2$. Call a translation vector the ***center*** of its copy. Since zero belongs to both component shapes, exact coverage gives $C_1 \cap C_2 = \varnothing$.

First-coordinate rows. In either component shape, v_1 is the only lattice point with no face-adjacent point in the same shape. Indeed, removing $v_1 + H_1$ isolates its center, whereas every other retained body point and every detached frame point has a neighbor in its piece. For a copy centered at c, consider $p = c + ne_1 + v_1$, the center of its first detached frame. All six face-neighbors of p belong to that frame, while p itself must be covered by another copy. The covering point in that copy must therefore be isolated, so it is its v_1 point. Its center is $c + ne_1$. Thus $c + ne_1 \in C$ for every $c \in C$.

These forward rows give a useful separation property:

$$c, d \in C, \quad |d_2 - c_2|, |d_3 - c_3| \le 200 \quad \Longrightarrow \quad d_2 = c_2,\ d_3 = c_3,\ d_1 - c_1 \in n\mathbb{Z}.$$

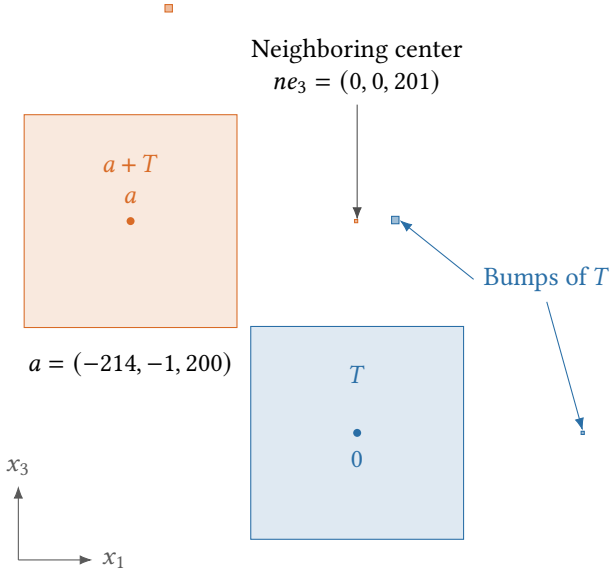
To prove it, shift one of the two centers forward by a nonnegative multiple of ne_1 so that their first-coordinate difference lies in $[-100, 100]$. The negative-first-coordinate halves of their main boxes contain no dents. Under the displayed transverse bounds, those two retained halves intersect, so exact coverage forces the shifted centers to coincide.

The rows also extend backward. Otherwise, if $c - ne_1 \notin C$, the separation property and forward closure exclude every center strictly to the left of c with the same transverse coordinates. Consider the two points

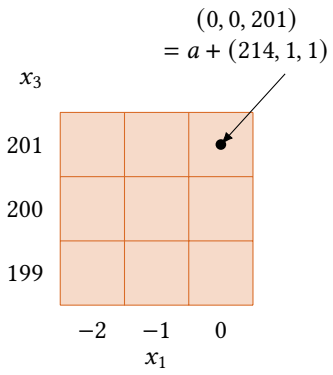
$$c + (-301, 100, 100), \qquad c + (-201, 100, 100).$$

The separation property rules out coverage by a main body or any of the three common bumps. Both points must therefore lie in type-specific bumps. The corresponding centers differ by at most 10 in each transverse coordinate, since these bumps all move by the same vector nw. Their first-coordinate difference lies in $[79, 121]$: the target points differ by 100, and the first coordinates within a type-specific bump range from 44 to 65. This contradicts divisibility by $n = 201$ in the separation property. Consequently $C + ne_1 = C$.

Coverage by the projected main boxes. Every $(y, z) \in \mathbb{Z}^2$ lies in the projection of some $c + B_b$, $c \in C$. Otherwise, cover the four points $(50r, y, z)$, $r = 0, 1, 2, 3$. None can lie in a main body or a first bump, so each is in a second, third, or type-specific bump. Two points use the same category. The corresponding centers are within 10 in both transverse coordinates, and hence their first-coordinate difference is a multiple of 201. But their target coordinates differ by 50, 100, or 150, whereas the first-coordinate range within any one category has width at most 21. Their center difference therefore lies between 29 and 171, a contradiction.



(a) View along the second coordinate



(b) Magnified layer of the covering bump in the plane $x_2 = 0$; each square is one unit cube.

Figure 5: Failure of the local forcing claim in [MSS25, Lemma 2.1] for the base shape T illustrated in Figure 4(a). (a) Exact projections of T (blue) and $a + T$ (orange), drawn to scale as unions of unit cubes centered at lattice points. Their projections are disjoint, so the copies are disjoint in $\mathbb{Z}^3$. Dents and the second bump are hidden by the main body in this view. (b) The translated first bump nevertheless covers the neighboring center ne_3, at the marked lattice point. This is a partial placement, not a tiling; the rigidity theorem and the reduction remain correct in the setting used here.

Forcing the remaining coordinate steps. For $i = 2, 3$, take the whole common bump $p + H_i$, where $p = c + ne_i + v_i$. Projected coverage gives a main box containing the transverse coordinates of p; shifting its center along its first-coordinate row gives $d \in C$ with $p \in d + B_b$. Here $d \neq c$. The bump cannot intersect the retained body at d. Choose the point of $p + H_i$ displaced in the first coordinate toward d. It is still inside $d + B_b$, so it lies in a dent. All dents have first coordinate between 11 and 65 and other coordinates between -5 and 5, relative to d. These bounds put $p - d$ in $[8, 68] \times [-5, 5]^2$, so the entire frame $p + H_i$ lies inside $d + B_b$. It must therefore lie entirely in the union of that copy's dents.

Each frame is face-connected, and the dents are separated by more than one lattice step, so this whole frame lies in a single dent. A translate of H_i can be contained in H_k only when $i = k$ and the translation is zero: the extreme coordinates give $i \leq k$, and if $i < k$, a corner directed toward the center lies strictly inside B_{k-1}, contradicting membership in H_k. Thus the matching dent is $d + v_i + H_i$, giving $d = c + ne_i$. This step uses coverage to fit an entire frame into a dent.

We now have forward closure of C in all three coordinate directions. For any two centers, shift them forward in directions e_2, e_3 to reduce both transverse differences to $[-100, 100]$. The separation property shows that the reduced differences are zero and their first-coordinate difference is divisible by n. Hence all centers are congruent modulo $n\mathbb{Z}^3$, and normalization gives $C \subseteq n\mathbb{Z}^3$. Conversely, projected coverage supplies a center with the transverse coordinates of any prescribed point of $n\mathbb{Z}^3$, since two such coordinates differing by at most 100 must agree. First-coordinate row invariance then supplies the prescribed center itself. Therefore $C = n\mathbb{Z}^3$.

Periods of the individual types. For a type-j copy at c, the grid contains a copy at $c + nw$. If its type were different, its body would retain the entire frame $v_{j+3} + H_{j+3}$, overlapping the type-j bump from c. Thus moving by nw preserves the type. Every grid point has a predecessor under this move, so $C_j + nw = C_j$ for $j = 1, 2$.

Converse. Suppose $C_1 \sqcup C_2 = n\mathbb{Z}^3$ and each C_j is invariant under nw. Start with the disjoint tiling by the boxes $c + B_b$, $c \in n\mathbb{Z}^3$, assigning type j at centers in C_j. Every common dent is filled exactly by the corresponding bump from the center $c - ne_i$. Every type-j dent is filled exactly by the type-j bump from $c - nw$, whose type agrees by invariance. These moves replace pairwise disjoint frame regions, so coverage remains exact and yields the required mixed tiling.

The row and separation arguments are `MSS.mixed_first_forward` in `MSS/FirstBump.lean`, `MSS.mixed_centers_near` in `MSS/RayPacking.lean`, and `MSS.mixed_first_period` in `MSS/FirstPeriod.lean`.

Projected coverage and whole-frame matching are `MSS.mixed_transverse_cover` in `MSS/TransverseCoverage.lean`, `MSS.translated_frame_subset_holes` in `MSS/FrameMatching.lean`, and `MSS.mixed_common_forward` in `MSS/CommonForcing.lean`.

Grid alignment and type periods are `MSS.mixed_grid_centers` in `MSS/GridCenters.lean` and `MSS.mixed_color_periods` in `MSS/ColorPeriods.lean`. Finally, `MSS.mixed_rigidity_sufficient` in `MSS/GridAssembly.lean` proves the converse and `MSS.rigidity` combines both directions.

□

6.2.2 Assembly and the Tiling Correspondence

Theorem 6.5 (Meyerovitch–Sanadhya–Solomon)**.** *Given $m \geq 1$ and a finite nonempty $F \subset \mathbb{Z}^2 \times \mathbb{Z}/m\mathbb{Z}$, one can compute a finite nonempty $\widetilde{F} \subset \mathbb{Z}^3$ such that*

$$F \text{ tiles } \mathbb{Z}^2 \times \mathbb{Z}/m\mathbb{Z} \quad \Longleftrightarrow \quad \widetilde{F} \text{ tiles } \mathbb{Z}^3.$$

Proof. Translate F first to contain zero, and take its finite set of representatives $U = \pi^{-1}(F) \cap D_m$. In particular $0 \in U$ and π is injective on U. The prototile is explicitly

$$\widetilde{F} = T_2 \sqcup \bigsqcup_{u \in U \setminus \{0\}} (nu + T_1). \tag{6.2}$$

These component pieces are disjoint. If two points from different pieces were equal, their component points would have the same residue modulo n. The corresponding representatives in the shapes differ by an element of nL, so the two elements of U would differ by an element of L, contradicting injectivity of π on U. Formula (6.2) is therefore a computable finite set of size $n^3|F|$.

Suppose $A \oplus F = \Gamma$, translating A if necessary so that $0 \in A$. Its lift $B = \pi^{-1}(A)$ satisfies

$$B + w = B, \qquad B \oplus U = \mathbb{Z}^3.$$

For the second identity, descend a target point by π, use its unique representation in $A \oplus F$, lift the F term to its unique representative in U, and obtain the unique B term by subtraction. Set

$$C_2 = nB, \qquad C_1 = \bigsqcup_{u \in U \setminus \{0\}} n(B + u).$$

These sets partition $n\mathbb{Z}^3$ and are invariant under nw. Lemma 6.4 gives an exact mixed tiling by T_1, T_2. Grouping its components according to (6.2) gives $nB \oplus \widetilde{F} = \mathbb{Z}^3$.

Conversely, let $\widetilde{A} \oplus \widetilde{F} = \mathbb{Z}^3$ and translate $\widetilde{A}$ so that $0 \in \widetilde{A}$. Decompose every translated copy into its component pieces. Their sets of translations are

$$C_2 = \widetilde{A}, \qquad C_1 = \bigsqcup_{u \in U \setminus \{0\}} (\widetilde{A} + nu).$$

The displayed union is disjoint, because a repeated translation vector would repeat a nonempty copy of T_1 in an exact tiling. Lemma 6.4 implies that C_1, C_2 partition $n\mathbb{Z}^3$ and are invariant under nw. In particular $\widetilde{A} \subset n\mathbb{Z}^3$. Writing $B = \{a/n : a \in \widetilde{A}\} \subset \mathbb{Z}^3$, the partition and invariance give

$$B \oplus U = \mathbb{Z}^3, \qquad B + w = B.$$

Hence B is a union of whole cosets of L, and $A = \pi(B)$ satisfies $\pi^{-1}(A) = B$. The identity $B \oplus U = \mathbb{Z}^3$ now descends to $A \oplus F = \Gamma$. Explicitly, lift a target of Γ to $\mathbb{Z}^3$ and use its unique U term; any other choice of an F term would lift to a different U term with the remaining point still in B, a contradiction. This proves both directions without assuming periods of the original tiling. The assembled-tile equivalence is `Compiler.LatticeGeometry.assembled_iff_quotient` in `Compiler/LatticeCompleteness.lean`, using the proved `MSS.rigidity`. □

6.3 Making the Prototile Connected

Kim's connectedness reduction preserves the number of prototiles, including one [Kim26, Theorem 2.5]. We give its finite construction in our notation and use its rigidity assertion as a black box. For an integer $a \geq 0$, write $C_a = \{0, \ldots, a-1\}^3$, with $C_0 = \varnothing$. Given a prototile $E \subset \mathbb{Z}^3$, translate it into C_l for a computable integer $l \geq 2$, and set

$$c = l^3 - (l-2)^3, \qquad s = 3c + 6.$$

Here c counts the points in the ***outer layer*** $\partial C_l = C_l \setminus ((1,1,1) + C_{l-2})$; enumerate them as $x_1, \ldots, x_c$ in lexicographic order.

6.3.1 The Interlocking Partition

Define a function $\chi_c : C_{c+2} \to \{1, \ldots, c\}$ by

$$\chi_c(x,y,z) = \begin{cases} y, & 0 \leq x \leq c,\ 1 \leq y \leq c,\ z = 0, \\ x, & 1 \leq x \leq c,\ 0 \leq y \leq c,\ z = c+1, \\ y, & x = 0,\ 1 \leq y \leq c,\ 1 \leq z \leq c, \\ x, & 1 \leq x \leq c,\ y = 0,\ 1 \leq z \leq c, \\ z, & 1 \leq x, y \leq c+1,\ 1 \leq z \leq c, \\ 1, & \text{otherwise.} \end{cases} \tag{6.3}$$

Let $Q_i = \chi_c^{-1}(i)$. This is Kim's partition from [Kim26, Section 2.1]: the pieces Q_i are connected, and every two different pieces meet along a face both inside one box and across corresponding faces of neighboring boxes. More precisely, disjoint lattice sets are ***adjacent*** when some point of one differs from a point of the other by a signed coordinate unit vector; the partition satisfies

$$Q_i \text{ is adjacent to } Q_j \text{ and to } Q_j + (c+2)v \quad (i \neq j,\ v \text{ a signed coordinate unit vector}).$$

These are the internal and external adjacency properties of [Kim26, Lemmas 2.1–2.2].

Replace each point of Q_i by a $3\times3\times3$ block: $R_i = 3Q_i + C_3$. Define the three ***bumps*** and their matching ***dents*** by

$$B_{\text{bump}} = \{(-1,1,1),(1,-1,1),(1,1,-1)\}, \qquad D_{\text{dent}} = \{(s-1,1,1),(1,s-1,1),(1,1,s-1)\},$$

and put $Q_1' = (R_1 \setminus D_{\text{dent}}) \cup B_{\text{bump}}$, $Q_i' = R_i$ for $i > 1$. The thickened pieces retain connectedness and their adjacency properties. The bumps and dents enforce a common grid in tilings of the assembly

$$S_l = \bigsqcup_{i=1}^{c} (sx_i + Q_i'). \tag{6.4}$$

The sum in this definition is disjoint: moving points by sx_i does not change their residues modulo s, and the pieces Q_i' together contain exactly one representative of every such residue. Thus $|S_l| = s^3$ and $S_l \oplus s\mathbb{Z}^3 = \mathbb{Z}^3$.

6.3.2 The Rigidity Input and the Connected Prototile

Lemma 6.6 (Kim's connected assembly)**.** *The set S_l in* (6.4) *is connected. For distinct $u, v \in C_l$, the sets $su + S_l$ and $sv + S_l$ are disjoint and adjacent. Moreover, if $A \oplus S_l = \mathbb{Z}^3$, then $A = t + s\mathbb{Z}^3$ for some $t \in \mathbb{Z}^3$.*

These are [Kim26, Lemmas 2.3–2.4 and the connectivity argument in Theorem 2.5]. The last assertion is the rigidity statement we import. It concerns arbitrary tilings, so it suffices for an existence reduction. The contact property explains how separated points of the original prototile become joined: their replacements have interlocking pieces in overlapping boxes, while their lattice points remain disjoint. The Lean counterparts are `Kim.shell_connected` in `Proofs/KimShellConnectivity.lean`, `Kim.shell_geometry` in `Proofs/KimShellGeometry.lean`, and `Kim.cosetRigidity` in `Proofs/KimRigidity.lean`.

Proposition 6.7 (Connectedness reduction, Kim)**.** *From any finite nonempty $E \subset \mathbb{Z}^3$ one can compute a finite nonempty connected $E^{\text{conn}} \subset \mathbb{Z}^3$ such that*

$$E \text{ tiles } \mathbb{Z}^3 \quad \Longleftrightarrow \quad E^{\text{conn}} \text{ tiles } \mathbb{Z}^3.$$

The construction preserves the number of prototiles (one) and the dimension.

Proof. After the harmless translation $E \subset C_l$ above, define

$$E^{\text{conn}} = sE \oplus S_l = \bigsqcup_{e\in E} (se + S_l). \tag{6.5}$$

Every piece is connected and every pair of different pieces is adjacent by Lemma 6.6, so their union is connected. The representation is unique because S_l is a fundamental domain for $s\mathbb{Z}^3$. All sets in the construction are finite and explicit.

If $A \oplus E = \mathbb{Z}^3$, then

$$sA \oplus E^{\text{conn}} = s(A \oplus E) \oplus S_l = \mathbb{Z}^3.$$

Conversely, suppose $A' \oplus E^{\text{conn}} = \mathbb{Z}^3$. Decomposing every copy using (6.5) gives a tiling by S_l with set of translations $A' \oplus sE$. By Lemma 6.6,

$$A' \oplus sE = t + s\mathbb{Z}^3$$

for some $t \in \mathbb{Z}^3$. Fix $e_0 \in E$; for every $a' \in A'$, the point $a' + se_0$ lies in this coset, so $a' - t \in s\mathbb{Z}^3$. Dividing the displayed unique-representation identity by s gives

$$\{(a' - t)/s : a' \in A'\} \oplus E = \mathbb{Z}^3.$$

This proves the tiling equivalence, the $k = 1$ case of [Kim26, Theorem 2.5]. The conversion is verified by `Compiler.connectedTile_connected` and `Compiler.connectedTile_correct` in `Compiler/ConnectedCompiler.lean`, using `Kim.cosetRigidity`. □

Proof of Theorem 1.1. Construct the domino relations of the input Wang tiles, apply Corollary 6.3, apply Theorem 6.5, and finally apply Proposition 6.7. The last step makes the output connected. The composition is a computable many-one reduction and keeps the ambient dimension equal to three for every input. The classical domino problem gives co-RE-hardness.

For membership, introduce Boolean variables $b_a \in \{0,1\}$ for $a \in \mathbb{Z}^3$ and write the tiling equations

$$\sum_{f \in F} b_{x-f} = 1 \qquad (x \in \mathbb{Z}^3). \tag{6.6}$$

For each r, test all assignments to the finitely many variables in the equations with $x \in [-r, r]^3 \cap \mathbb{Z}^3$. If no assignment exists, report that F does not tile. Conversely, if every such finite system is satisfiable, compactness of $\{0,1\}^{\mathbb{Z}^3}$ yields an assignment satisfying every equation, and its support $\{a : b_a = 1\}$ tiles $\mathbb{Z}^3$. Thus translational monotiling is in co-RE, completing the proof. The finite-obstruction search is `tiles_coRE` in `Proofs/FiniteSearch.lean`; the connected conclusion follows from `connected_complete_of_reduction` in `Proofs/Complexity.lean` applied to `Compiler.connectedWangReduction`. □

6.3.3 An Explicit Connected Example

Take $E = \{(0,0,0), (1,1,1)\} \subset C_2$. Its two points are not face-adjacent. With $l = 2$, the outer layer is all of C_2, so $c = 8$ and $s = 30$. Thus the construction gives

$$E^{\text{conn}} = S_2 \sqcup ((30,30,30) + S_2), \qquad |S_2| = 27{,}000, \quad |E^{\text{conn}}| = 54{,}000.$$

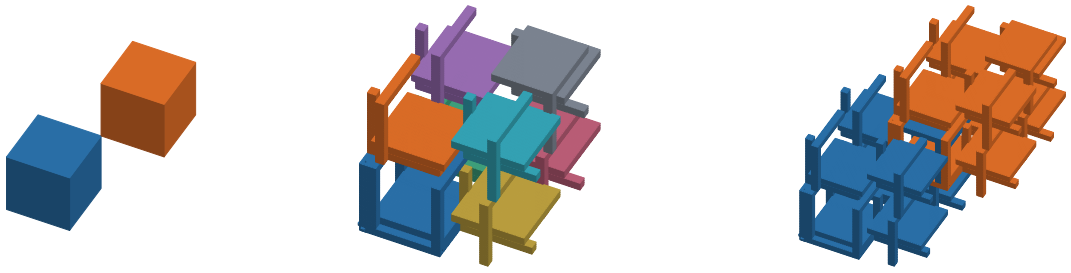
Figure 6 renders these exact sets as unit-cube unions. The two copies of S_2 are disjoint and share unit-square faces, joining the two components without adding a separate connector. This is an example of the connectedness step, rather than the output of the full Wang-to-polycube reduction.

6.4 Quantitative Size Bounds

Even the one-tile upper bound for this construction is enormous. The following bounds count distinct unit cubes in the final connected polycube, including the quotient-to-lattice and connectedness steps.

Corollary 6.8. *Given a finite set $\mathcal{T}$ of n Wang tiles, one can construct a finite nonempty face-connected $F \subset \mathbb{Z}^3$ with*

$$|F| \le (n+2)^{10^{101}(n+1)} = 2^{O(n \log n)},$$



(a) Input E (b) Assembly S_2: 27,000 cubes (c) Connected prototile: 54,000 cubes

Figure 6: An exact three-dimensional example of Kim's connectedness construction [Kim26, Section 2], with panels scaled separately. (a) The two input cubes meet only at a corner. (b) The eight colors identify the pieces $30x_i + Q_i'$ of S_2. (c) Blue and orange identify its two translated copies in E^{conn}; these copies have disjoint interiors and meet along faces. The rendering includes the bumps and dents, although they are small at this scale.

such that $\mathcal{T}$ tiles the plane if and only if F tiles $\mathbb{Z}^3$ by translations. For $n = 1$, there is such an F with the explicit bound

$$|F| \le 2^{10^{100}}.$$

The explicit constant is proved in `Compiler/ExplicitGeneralBound.lean` by `Compiler.Size.exists_connected_tile_explicit_power_bound`. `Compiler.Size.exists_connected_tile_explicit_exp_log_bound` gives the sharper binary-exponential bound

$$2^{10^{100}(n+1)(\lfloor \log_2(n+2) \rfloor + 1)}.$$

These constants are independent of the Wang tile set. The theorem uses bounded admissible parameter choices in the existing reduction. It does not bound the original compiler's search for a certificate with the least numerical encoding, or assert a lower bound on simulation size.

Proof. Write $N = |I|$, $C = N + 2$, and $A = (p-1)(q-1)$; these are fixed constants. The ordinary label sets have cardinality An and the seed label sets have cardinality one. Choose the useful primes a_i, b_i independently of the input, bounded by a fixed constant U.

Assign digit primes by their ranks in the finite set of channel-label pairs, so numerical color names have no cost. Chebyshev's elementary prime-counting bound gives the coarse estimate $p_k \le (8(k+1))^2$ for the zero-indexed kth prime. Consequently, all digit primes can be bounded by a fixed polynomial $Q(n)$, while remaining distinct and above the fixed threshold needed for the block decompositions. This is proved in `Compiler.Size.exists_polynomial_parameters` in `Compiler/PolynomialPrimes.lean`. Thus $r_i \le Q(n)^{An+1}$. There is a common budget $b = (n+2)^{O(n)}$ bounding every r_i, a_i, b_i, the channel count, and the label counts. The horizontal coordinate bound and $|V|$ are fixed polynomials in b.

To count equations, retain the fixed useful-output sizes $|P_i| \le U^2$. For any selected tuple of labels, each cycle-test map $d_i : P_{\mathrm{prev}(i)} \to U_{i,j(i)}$ has at most b^{U^2} choices, because $|U_{i,j(i)}| \le r_i \le b$. A cycle has at most C channels, so at most b^{CU^2} tests are needed for that tuple. There are at most $(An)^N$ ordinary words and $2NAn$ seed exclusions; graph, dependence, and activation tests add only a fixed number of equations. Hence the total number of equations is a fixed polynomial in b, as proved in `Compiler.Size.testTiles_card_le_direct` in `Compiler/DirectTestCount.lean`. Counting all subsets of an ambient support would lose this bound.

The stacking prime can be chosen polynomially bounded in the number of equations and $|V|$, so the final cyclic modulus is also polynomial in b. The quotient-to-lattice and connectedness steps have polynomial volume cost in that modulus and the horizontal coordinate bound. Therefore $|F| \le b^{O(1)} = (n+2)^{O(n)}$. `Compiler.Size.exists_connected_tile_single_bound` combines these estimates with nonemptiness, face connectedness, and both tilability directions; `Compiler.Size.singleCubeBound_le_power` supplies one constant K uniformly for every n, including zero. □

For the explicit constant, take $U = 10^{40}$ and digit primes at most $10^{164}(n+2)^2$. Then $b = (n+2)^{10^8(n+1)}$ is a common parameter budget. Writing $L = \lfloor \log_2(n+2) \rfloor + 1$, the equation count, horizontal coordinate bound, and $|V|$ are each at most $2^{10^{98}(n+1)L}$. Stacking and geometry give $|F| \le 2^{10^{100}(n+1)L}$. Finally, $2^L \le (n+2)^2$ gives $K = 10^{101}$ in the power bound.

The one-tile bound is proved in `Compiler/ExplicitOneBound.lean` by `Compiler.Size.exists_connected_tile_one_explicit_bound`. The useful primes are at most 10^{40} and the digit primes at most 10^{164}, giving a parameter budget $b = 2^{10^8}$. The direct equation count is at most $2^{10^{98}}$; stacking and the geometric steps give at most $2^{10^{100}}$ cubes. This specializes the full construction without first deciding whether the Wang tile tiles the plane. The earlier general numerical bound $2^{2^{10^{40}(n+1)^2}}$ remains proved in `Compiler/SimpleSizeBound.lean`. It uses a powerset estimate for the equation count and weaker prime bounds.

6.4.1 Ternary and Mixed-Radix Channels

The linear dependence on n in the exponent comes from assigning a separate prime to each possible label in a channel. Encoding the Wang label and arithmetic decorations over a fixed small alphabet changes the number of channels instead. We also optimize the cycle test using cyclicity of each digit subgroup.

Lemma 6.9 (One fixed cycle test)**.** *In Lemma 3.2, a single test suffices. Fix the generator* $u_i = r_i/r_{i,j(i)} \pmod{r_i}$ *of* $U_{i,j(i)}$ *and use*

$$d_i(e) = \begin{cases} 0, & e = 0, \\ u_i, & e \neq 0. \end{cases}$$

Proof. If one selected label is inactive, the same inverse argument works for this fixed map. If all selected labels are active at some x, then $g_{i,x}$ is not invariant under $U_{i,j(i)}$. Since u_i generates this subgroup, there is a v_i with $g_{i,x}(v_i) \neq g_{i,x}(v_i - u_i)$. Put

$$\epsilon_i = g_{i,x}(v_i) - g_{i,x}(v_i - u_i) \neq 0.$$

Extend ϵ by zero off the cycle and take $k_i = v_i$ on it. Then $d_i(0) = 0$ and $d_i(\epsilon_{\mathrm{prev}(i)}) = u_i$, so 0 and ϵ have the same image in (3.8). The generator, adjacent witnesses, and exclusion equivalence are checked in `FiniteCompiler/FixedCycleTests.lean` by `FiniteCompiler.digitGenerator_spans`, `FiniteCompiler.digitActive_step`, and `FiniteCompiler.fixed_cycle_exclusion_iff`. The numerical implementation is checked in `FiniteCompiler/Effective/FixedCorrect.lean`. □

Write $\lg = \log_2$ for the real logarithm to base 2. For $n \geq 1$, the Sudoku alphabet has $|\Sigma| = (p-1)(q-1)n = 24n$ symbols. Choose the least integer k with $24n < 3^k$ and define an injective code $\widehat{c} : \Sigma \to \{0,1,2\}^k$. Concretely, for arithmetic digits $a \in \{1,\dots,4\}$, $a' \in \{1,\dots,6\}$ and Wang tile index $j < n$, use the base-three expansion of

$$((a-1)6 + (a'-1))n + j < 24n.$$

The code and its computable numerical form are checked in `FiniteCompiler/Effective/JointCode.lean`.

Replace each ordinary channel $i \in I$ by channels (i,b), $b \in \{1,\dots,k\}$, all having the same line map L_i. The two seed channels stay unchanged. The new ordinary alphabet is $\Gamma = \{0,1,2\}$. A word over the new channels is allowed exactly when it encodes some word in $\mathcal{W}$. Keep the original graph, dependence, and activation equations and the fixed cycle tests for forbidden words. For each seed t, original column i, and symbol $s \notin B_t$, add one fixed cycle through seed t and the k channels (i,b), labeled by the seed's unique label and the codeword of s. This forbids their simultaneous activity. Unused codewords need no seed tests, because the word constraints already exclude them. The joint seed tests and their soundness and completeness are checked in `FiniteCompiler/JointSystem.lean`. The computable payload and numerical family are checked in `FiniteCompiler/Effective/JointInput.lean`, `FiniteCompiler/Effective/CycleSpec.lean`, and `FiniteCompiler/Effective/JointFamily.lean`.

Proposition 6.10 (Quasipolynomial channel bound)**.** *A set of* n *Wang tiles can be simulated by a nonempty face-connected polycube with at most*

$$B(n) = \max\{(n+4)^{8.83302\left(\lg\lg(n+4)+\lg\lg\lg(n+4)\right)+570}, 2^{2392}\} = n^{O(\log\log n)}$$

unit cubes. For $n = 1$*, the same construction uses at most* 2^{2112} *unit cubes.*

Proof. The guarded encoder requires a nonempty Wang alphabet. For an empty input we use an alphabet consisting of one tile with unequal north and south colors. It cannot tile the plane, and we pass it through the same encoder and geometric construction.

First derive the coefficient 2000 for the baseline joint-code compiler; then refine its channel system for 1350, restrict arithmetic columns for 1000 and 850, and retain two Wang-tile columns for 800. Check correctness of the expanded channel system. Activation gives a nonempty active set in every ordinary ternary channel. Choose one active symbol $V(i, b, m)$ at the common vertical section $(0, m)$. At every horizontal point x, the tuple $(V(i, b, L_i(x)))_{i,b}$ is an allowed encoded word, by the fixed word exclusions. In particular, at $(0, m)$ the digits for each i decode uniquely to a symbol $W(i, m) \in \Sigma$. At a general x, injectivity of the code shows that decoding its ith digit tuple gives this same $W(i, L_i(x))$. Hence W satisfies the original line rule. This coherence step is checked by `Compiler.BitChannels.array_of_encoded_line_rule` in `Compiler/BitChannels.lean`.

Each seed activates somewhere. If a decoded column symbol were outside B_t there, its joint seed cycle would have every selected label active, contradicting the fixed test. Thus both arithmetic digits of every decoded column symbol equal t. The original two-prime soundness argument applies. Conversely, encode the canonical Sudoku array with $\widehat{c}$. Each ternary channel has a singleton active set, using its own block functions and the same slope L_i. The ordinary activation inverse uses only that channel and its slope; it requires no distinctness of slopes between channels. The two-seed construction and its common auxiliary output remain valid. The encoded line and seed restrictions hold, so Lemma 6.9 proves all exclusion equations on this same graph. This proves both tilability directions.

Now write $C = Nk + 2$ for the total number of channels. The constants N, M, and D are independent of n. There are $O(C)$ useful prime tags. Choosing their ranks above the fixed seed threshold and using $p_k \le (8(k+1))^2$ gives a bound $U = C^{O(1)}$ for all useful primes. The digit-prime threshold is $O(U^2)$; there are again only $O(C)$ digit-prime tags because $|\Gamma| = 3$. Their primes thus have a bound $Q = C^{O(1)}$. Each channel has at most three labels, so $r_i \le Q^3 = C^{O(1)}$ and $|P_i| \le U^2 = C^{O(1)}$. Consequently the cyclic factor has order

$$|V| = D \prod_i r_i^2 a_i b_i = C^{O(C)}.$$

There are at most 3^{Nk} forbidden ternary words. Each needs one fixed cycle test. There are at most $2N3^k$ joint seed tests, $O(C^2)$ dependence equations, and C activation equations. The total number of equations is therefore $s \le 3^{Nk} + 2N3^k + O(C^2) = 2^{O(C)}$. Enumerating all maps would instead introduce the growing exponent $|P_i|$; the fixed test avoids that loss.

The activation CRT moduli, batch indices, and their separated lifts are polynomial in the individual channel budgets. The slopes are still bounded by the fixed N, so the horizontal support bound is $H = C^{O(1)}$. If P bounds the prime divisors of $|V|$, the partition argument below gives a stacking prime $\ell = O(s^2 \lg(2s)+P)$ exceeding P. It is coprime to $|V|$, and the stacked cyclic modulus $\ell|V|$ is $C^{O(C)}$. The quotient-to-lattice and connectedness steps have fixed polynomial volume cost in H and $\ell|V|$. The final cube count is therefore $C^{O(C)}$. Since $k = O(\lg(n+2))$, this gives the quasipolynomial dependence.

For the numerical coefficient, write $w = Nk = 16k$ for the number of ordinary channels and use the prime budget $P = \max\{w + 2, 256\}$. When $w + 2 < 256$, restrict a prime allocation for 254 ordinary channels to the actual channels; the construction itself gains no extra channels. This is checked in `FiniteCompiler/BudgetParameters.lean`. The two auxiliary seed primes must exceed $D = (5pq)^2 = 30{,}625$. The general prime allocation uses the admissible choices

$$b_0 = 511{,}237, \qquad b_1 = 511{,}243,$$

whose common positive-representation threshold is at most $4b_1 + 3$. Reserve these primes and the fixed primes $2, 3, 5, p, q$. Choose the integer h with $h^2 \le P < (h+1)^2$ and put $U = Ph$. Then $h \ge 16$, $U^2 \le P^3$, and $2U \le P^2$. To optimize the prime allocation, choose integers S, J satisfying

$$32S \le 15U < 32S + 32, \qquad 32J \le 15P^3 < 32J + 32.$$

For $P \geq 962$, Chebyshev's elementary prime-counting bound gives $\pi(S) \geq P + 7$ and $\pi(U) \geq 2P + 7$. To optimize the minimum prime budget from 962 to 256, we also verify these bounds on $256 \leq P < 962$ using seventeen monotone intervals. The first is $[256, 278]$, where $S \geq 1920$, $U \geq 4096$, $\pi(1920) \geq 285$, and $\pi(4096) \geq 563$. The remaining intervals are checked in `Compiler/SmallBudgetCounting.lean`. Their prime certificates use a strictly increasing list of primes at most 29,791, with primality and ordering checked in `Compiler/SmallBudgetPrimeList.lean`. Thus we can choose the ordinary a_i primes at most S and then distinct b_i primes at most U, excluding the reserved primes. Each ordinary pair satisfies $a_i b_i \leq J \leq P^3$ and $a_i b_i + a_i + b_i \leq J + 2U$. The seed primes satisfy $a_0 = 2, a_1 = 3$ and $a_i b_i \leq P^4$; their positive-representation thresholds are at most $4b_1 + 3$.

Set $L = J + 2U + 117$ and $Q = P^3$. We have $L \leq 19P^3/40$. Writing $\theta(x) = \sum_{r \leq x,\ r \text{ prime}} \ln r$, Chebyshev's upper and lower bounds give

$$\theta(Q) - \theta(L) \geq P^3/80 > 3P \ln Q, \qquad \pi(Q) - \pi(L) \geq 3P + 1.$$

Here the first estimate uses $\ln P \leq P/40$ and $\sqrt{P^3} \leq P^2/16$. Hence all ordinary digit primes can exceed L and be at most P^3. This replaces choosing primes by indices exceeding L, which unnecessarily skipped the first L primes, with choosing primes by their values. For the two seed digit primes, put

$$B = \max\{P^3, \max\{P^3 + 2U, 4b_1 + 3\} + 117\} \leq 2P^3.$$

Two applications of Bertrand's postulate give distinct primes $B < z_0 \leq 2B < 2z_0$ and $z_0 < z_1 \leq 2z_0$. Both exceed the ordinary digit primes and the seed representation thresholds, and are at most $8P^3 \leq P^4$. These prime counts and choices are checked in `Compiler/SmallBudgetInterval.lean` and `FiniteCompiler/SmallBudgetParameters.lean`. The effective certificate checks the ordinary digit-prime bound $r_{ij} \leq P^3$ as well as the seed bound $r_{ij} \leq P^4$, so the remaining estimates apply to any certified parameters. For an ordinary channel, $r_i \leq P^9$, so its block modulus $r_i^2 a_i b_i$ is at most P^{21}. The two seed block moduli remain at most P^{12}, and the reference modulus $M = 5pq$ is at most P^2. Consequently,

$$|V|^{10} \leq P^{210w+280}.$$

Each forbidden word or nonseed codeword needs one fixed cycle test. There are at most 3^w forbidden words and $2N3^k = 32 \cdot 3^k$ joint seed tests. Counting the dependence indices directly gives at most $(w+2)^2 + w + 4$ equations; the kernel and activation equations contribute at most $w + 3$. Since $w = 16k$ and $k \geq 3$, we have $w^2 + 6w + 11 \leq 3^w$ and $32 \cdot 3^k \leq 3^w$, so

$$s \leq 3^w + 32 \cdot 3^k + w^2 + 6w + 11 \leq 3^{w+1}.$$

Since $w \geq 48$, we also have $P^2 \leq 3^{w+1}$: the base case is $256^2 \leq 3^{49}$, and the induction step uses $(P+1)^2 \leq 3P^2$. Thus

$$X = \max\{s, P^2\} \leq 3^{w+1} \leq P^{8P}.$$

These counts are checked in `FiniteCompiler/JointBounds.lean` and `FiniteCompiler/Effective/FamilyCompiler.lean`.

Let λ be the binary length of $2X$. Choose a prime ℓ between $64X^2\lambda + P^4 + 3$ and twice that number. Every prime divisor of $|V|$ is at most P^4, so ℓ is coprime to $|V|$. To check the partition condition, write $\ell = 2m + 1$. The failure probability from Lemma 6.1 is at most

$$s(2m)(1 - s^{-2})^m \leq 132X^3\lambda e^{-32\lambda} \leq 132e^{-28\lambda} < 1,$$

using $X \leq 2^\lambda \leq e^\lambda$ and $\lambda \leq e^\lambda$. The case $s = 1$ is immediate. This sharper estimate is checked in `Proofs/LogDifferenceBound.lean`; the computable partition search and coprime stacking are in `FiniteCompiler/`

`Effective/FreshStack.lean`. Since $\lambda \le P^2$, we have $\ell \le 132X^2\lambda \le P^3X^2$. Thus $\ell \le P^3 3^{2w+2}$, and the stacked modulus $R = \ell|V|$ satisfies

$$R \le P^{21w+31}3^{2w+2}.$$

To optimize the displacements, recover the low activation index from the CRT residue instead of packing it a second time. Put $A_i = r_i a_i b_i$. An ordinary activation index (u, α, ξ, d) has $u < r_i$, $\alpha < a_i$, $\xi < b_i$, and $d < 2$, so use

$$c = 2(\alpha b_i + \xi) + d < 2a_i b_i.$$

For a seed index with extra residues $x, y < M = 175$, use $c_{\text{seed}} = (cM + x)M + y < 2a_i b_i M^2$. The CRT residue recovers u, and the code recovers the remaining indices, so the pair is injective. Adding the relevant CRT modulus times the code to the CRT residue preserves every required congruence and gives distinct displacements. The ordinary offsets have both coordinates bounded in absolute value by $32r_i(a_i b_i)^2$; the seed offsets have first coordinate at most $2a_{\text{other}} r_i (a_i b_i)^2 M^3$ and second coordinate less than M. Ordinary channels have $r_i \le P^9$ and $a_i b_i \le P^3$; seed channels have $r_i \le P^4$ and $a_i b_i \le P^4$. Since $a_{\text{other}} \le 3$ and $6M^3 \le P^4$, these bounds allow

$$H \le P^{16},$$

improving the previous P^{25} support estimate. Dependence shifts also fit this bound. The residue-aware packing and support estimates are checked in `FiniteCompiler/ActivationOffsets.lean`, `FiniteCompiler/CoreSupport.lean`, and `FiniteCompiler/CompactSupport.lean`.

To optimize the size estimate, count points in the test tiles rather than their bounding boxes. Put

$$J = D \prod_i r_i a_i b_i \le P^{12w+20}.$$

Every test tile has at most J points. For the graph-kernel and cycle tiles, fixing the useful outputs fixes each low residue, leaving r_i high residues per channel and D auxiliary choices. Dependence tiles rearrange the graph-kernel tile. An ordinary activation tile has $r_i a_i b_i$ batches, each with at most $D \prod_{j \ne i} r_j a_j b_j$ points. A seed activation tile has $D r_i a_i b_i$ batches, each with at most $\prod_{j \ne i} r_j a_j b_j$ points. These counts use the total shift multiplicity $a_i b_i$. For the displayed bound on J, each ordinary factor is at most $P^9 P^3 = P^{12}$, each seed factor at most $P^4 P^4 = P^8$, and $D \le P^4$. The counts are checked in `FiniteCompiler/MassBounds.lean`.

Stacking distributes the test tiles among a partition of ℓ layers, so the stacked tile has at most ℓJ points. Each MSS component has at most 2^{24} cubes, independently of its displacement, so the unconnected lattice tile has at most $2^{24}\ell J$ cubes. Its horizontal coordinates have absolute value at most $1000(2H + 1)$, and its vertical coordinates at most $1000(2R + 1)$. Use $\widehat{R} = \max\{H^2, R\}$ to size the connectedness construction. Since $w \ge 48$, this still satisfies $\widehat{R} \le P^{21w+31}3^{2w+2}$. Translate the lattice tile into an $A \times A \times B$ box with $A^2 \le B$, $A \le 8003H$, and $B \le 8003^2\widehat{R}$.

To optimize the connectedness construction, separate the short planar directions from the long vertical direction. Use the planar slots $[-A, A]^2 \times \{0\}$, indexed by $u_0, \dots, u_{r-1}$ with $u_0 = 0$, and the rail slots $v_i = (0, 0, i)$ for $0 \le i < c = B + 1$. The computable enumeration gives $r \le 10A^2$. Identify the two slots at the origin. Every displacement between input points is a rail slot minus a planar slot, up to sign. Thus only rail–planar internal contacts are needed, together with the contacts that connect the slot set itself. Choose permutations p, q of the planar indices satisfying $p(j) = i$ whenever $u_j = u_i + e_1$, and $q(j) = i$ whenever $u_j = u_i + e_2$.

Use planar colors $C_0, \dots, C_{r-1}$ and rail colors $D_0, \dots, D_{c-1}$, with $C_0 = D_0 = 0$. Partition an eight-layer box of dimensions

$$L = 4r + 7(c + 1) + 1, \qquad Y = 4r + 1, \qquad Z = 8.$$

Let $\rho_z(x) = D_i$ on $4r + z + 7i \le x < 4r + z + 7(i + 1)$ for $1 \le i < c$, and zero elsewhere. Layers zero and seven have color $\rho_z(x)$, except that the bottom row $y = 0$ is all zero. In layers one through six, reserve the

line $x \ge 4r$, $y = 2r$ for $\rho_z(x)$. Elsewhere, layers one and two have colors $C_{y \bmod r}$ and $C_{x \bmod r}$. Layer three changes $C_{y \bmod r}$ to $C_{p(y \bmod r)}$ when $x \ge r$; layers four through six change $C_{x \bmod r}$ to $C_{q(x \bmod r)}$ when $y \ge r$. Every color piece is connected: the planar stripes route around the reserved line, successive rail strips overlap in six cells, and color zero connects the remaining boundary regions. Each rail color touches every planar color internally. The opposite faces realize p, q and the successive rail-slot adjacencies. Scale each cell by three, relocate its color piece to the corresponding slot, and apply Kim's three bump-and-dent pairs to color zero. The resulting shell forces grid scales $(3L, 3Y, 24)$ in every tiling. Since $A^2 \le B$, we have $L \le 62B$ and $Y \le 41A^2$, so its volume is at most $216LY \le 549{,}072A^2B$. Padding the long dimension to at least A^2 therefore gives a volume expansion factor $O(A^4 + A^2B)$, improving the four-layer variant's $O(A^2B^2)$ factor for the long boxes used here. The assembly preserves tilability and is connected, as before. Consequently the final volume is at most

$$2^{24}\ell J \cdot 549{,}072A^2B \le KH^2\widehat{R}\ell J, \qquad K = 2^{24} \cdot 549{,}072 \cdot 8003^4 \le P^{12}.$$

Substituting the bounds on H, $\widehat{R}$, ℓ, and J, and using $3^4 \le P$, gives

$$|F| \le P^{33w+98}3^{4w+4} \le P^{34w+99}.$$

The partition, contacts, connectivity, arbitrary-tiling rigidity, and computable shell assembly are checked in `Rails/`.

`FiniteCompiler / Effective / CompactFamilyCompiler.lean` implements the geometric compiler. Its volume estimate is checked in `FiniteCompiler / Effective / CompactFamilyBounds.lean` and `Rails/SizeBound.lean`.

Put $t = \lg(n+4)$ and $g = \lg t$. The choice of k gives $3^{k-2} \le 8n$. Using $2^3 \le 3^2$ yields

$$k \le \tfrac{2}{3}\lg n + 4 \le \tfrac{2}{3}t + 4.$$

For $n \ge 252$, we have $t \ge 8$ and $g \ge 3$, so

$$w \le \tfrac{56}{3}t, \qquad P \le 32t, \qquad \lg P \le 5 + g \le \tfrac{8}{3}g.$$

Hence

$$\lg|F| \le (34w + 99)\lg P \le \frac{8}{3}\left(34 \cdot \frac{56}{3} + \frac{99}{8}\right)tg = \frac{15{,}529}{9}tg < 2000tg.$$

For $92 \le n \le 251$, the ternary code has $w \le 128$ and $P = 256$, because $24 \cdot 251 < 3^8$. The same raw volume estimate gives $|F| \le 2^{35{,}402}$. Also $t \ge 79/12$ and $g \ge 27/10$, by $2^{79} \le 96^{12}$ and $2^{27} \le (79/12)^{10}$. Therefore

$$\lg|F| \le 35{,}402 < 35{,}550 \le 2000tg.$$

To optimize the smaller cases, use explicit prime parameters for $w = 48, 64, 80, 96, 112$. The ordinary useful primes are the first $2w$ primes excluding $2, 3, 5, 7$, pairing the first half with the reverse of the second half. The seed useful pairs are $(2, 30{,}631)$ and $(3, 30{,}637)$, whose second entries exceed $M^2 = 30{,}625$. A digit prime only needs a positive representation $r_{ij} = ua_i + vb_i$ with $u, v > 0$; the earlier threshold $r_{ij} > a_ib_i + a_i + b_i$ was sufficient, but is unnecessary when the coefficients are supplied explicitly. The five tables supply distinct digit primes, all avoiding the useful and reference primes, together with their positive coefficients. All primes are at most 65,536 and satisfy the preceding prime budgets. Primality, distinctness, positive representations, and numerical support and volume bounds are checked in `FiniteCompiler/Effective/SmallProfiles.lean`. The computable parameter selector uses these tables at these five widths.

For $n = 4, 5, 6$, further optimize the code alphabets: use three positions with radices $(4, 4, 6)$, $(4, 5, 6)$, and $(4, 6, 6)$, respectively. Their capacities $96, 120, 144$ are at least the $24n$ decorated Sudoku symbols.

The graph, dependence, cycle, and activation arguments allow a different finite alphabet in each channel. Whole-codeword decoding and seed tests therefore give the same tilability equivalence, with only $w = 48$ channels. The channel-dependent interface is `FiniteCompiler.AlphabetFamily` in `FiniteCompiler/Symbols.lean`; correctness is checked in `FiniteCompiler/MixedAlphabetSystem.lean`. The three additional prime tables and their support and volume certificates are checked in `FiniteCompiler/MixedProfile4.lean`, `FiniteCompiler/MixedProfile5.lean`, and `FiniteCompiler/MixedProfile6.lean`.

Retain the actual values of $|V|$, J, and the maximum support bound H for each table. Let W be the product of the ordinary channel alphabet sizes, so $W = 3^w$ for ternary channels. In the mixed cases, the non-word tests add at most 9515 equations and $W \geq 10{,}000$, so again $s \leq 3W$. The tables also give $3W \geq 256^2$. Here $P = 256$, so $X = \max\{s, 256^2\} \leq 3W$, and the binary length λ of $2X$ is at most 192. Thus

$$\ell \leq 132X^2\lambda \leq 2^{15}(3W)^2.$$

The finite checks also give $H^2 \leq 2^{15}(3W)^2|V|$, so $\widehat{R} = \max\{H^2, \ell|V|\} \leq 2^{15}(3W)^2|V|$ and

$$|F| \leq KH^2|V|J\big(2^{15}(3W)^2\big)^2.$$

The following upper bounds $|F| \leq 2^e$ and lower bounds for t, g imply $\lg|F| \leq 2000tg$ for the remaining inputs. Here k counts code positions per original column; the rows $n = 4, 5, 6$ use the mixed alphabets above.

n	k	e	lower t	lower g
4	3	9460	3	19/12
5	3	10,019	31/10	163/100
6	3	10,580	33/10	17/10
$7, \ldots, 10$	5	11,678	17/5	7/4
$11, \ldots, 30$	6	14,302	39/10	49/25
$31, \ldots, 91$	7	17,001	5	23/10

Each rational logarithm bound is checked by an exact power comparison: for example, $2^{31} \leq 9^{10}$ and $2^{163} \leq (31/10)^{100}$ give the row $n = 5$. The ternary parameter identities and geometric estimates are checked in `FiniteCompiler/Effective/SmallMetrics.lean` and `FiniteCompiler/Effective/CompactFamilyBounds.lean`. The mixed-alphabet equation counts, geometric compiler, and volume bounds are checked in `FiniteCompiler/MixedAlphabetBounds.lean`, `FiniteCompiler/MixedGeometry.lean`, and `FiniteCompiler/MixedCompiler.lean`. For each exceptional positive input size, color normalization leaves a fixed finite input domain. Applying the proved construction on this domain gives a fixed finite table, whose computability is checked by `Compiler/FiniteNormalizedInput.lean` and `FiniteCompiler/MixedCompiler.lean`. The baseline logarithm conversion is checked in `Compiler/JointLogBound.lean`.

To reduce the coefficient from 2000 to 1350, separate the arithmetic labels from the Wang-tile labels and omit components unused by the extraction. Keep the 5-arithmetic labels on columns $0, \ldots, 11$, the 7-arithmetic labels on columns $0, \ldots, 15$, and Wang-tile labels only on columns $1, 5, 7$. Impose the original word rule after existentially filling the omitted components. Completeness follows by restricting the canonical Sudoku array.

For soundness, apply the arithmetic classification separately to the boards of widths $2p + 2 = 12$ and $2q+2 = 16$. No small-prime certificate changes. The finite-depth arithmetic argument works on each of these boards, and the same CRT argument chooses a common height shear for both components. On a horizontal word with constant arithmetic labels, coefficient recovery at columns $1, 2, 3, 4$ makes all three retained Wang labels equal. On the line through heights Mn, the arithmetic recovery samples are $1, 2, 3, 4, 5, 10$ for $p = 5$ and $1, 2, 3, 4, 7, 14$ for $q = 7$, all retained. The labels at columns $1, 5, 7$ then read the corners $(0, 0)$, $(1, 0)$, $(0, 1)$ of the valid rectangle witnessing the word rule. Thus

$$H\text{-matches}(C_1(M), C_5(5M)), \qquad V\text{-matches}(C_1(M), C_7(7M)).$$

Horizontal-word consistency replaces the second labels by $C_1(5M)$ and $C_1(7M)$. Reading $C_1(5^u7^v)$ gives arbitrarily large valid Wang rectangles, and compactness gives a plane tiling. The separate-board arithmetic is checked in `SudokuArithmetic/SparseArithmetic.lean`; the restricted rule and extraction are checked in `SudokuArithmetic/SparseRule.lean`, `SudokuArithmetic/SparseRecovery.lean`, and `SudokuArithmetic/SparseSoundness.lean`.

For the general-input compiler, retain both arithmetic components on all sixteen original columns and encode each of their 24 possible pairs by three trits. Encode a Wang tile by k trits with $3^{k-1} \le n < 3^k$, but keep this code only in columns 1, 5, 7. Each group of trits retains its original slope. Fix unused padding trits to zero to make the number w of ordinary channels a multiple of 16, with

$$48 + 3k \le w \le 48 + 3k + 15.$$

For each seed, the required arithmetic pair is one symbol. Excluding incorrect digits of its three-trit code therefore fixes the whole arithmetic pair without restricting any Wang-tile digit. These use at most 288 fixed two-channel cycle tests. Together with word, dependence, and activation tests, they give

$$s \le 3^w + 288 + w^2 + 6w + 11 \le 3^{w+1}.$$

All previous support, mass, stacking, and connectedness estimates apply, so again

$$|F| \le P^{33w+98}3^{4w+4}, \qquad P = \max\{w + 2, 256\}.$$

The projected word list is obtained by mapping the original finite list of allowed words; no new decision about tilability is used. The typed system's equivalence is checked in `FiniteCompiler/SparseTernaryModel.lean`. Its effective word and seed enumerations are combined in `FiniteCompiler/Effective/SparseFamily.lean`.

For $n \ge 252$, put $t = \lg(n + 4)$ and $g = \lg t$ as before. Minimality of k and $\lg 3 \ge 3/2$ give

$$w \le 2t + 66 \le \tfrac{41}{4}t, \qquad P \le 32t, \qquad \lg P \le \tfrac{8}{3}g.$$

Hence the general-input binary exponent is at most

$$(34w + 99)\lg P \le \frac{8}{3}\left(34 \cdot \frac{41}{4} + \frac{99}{8}\right)tg = \frac{2887}{3}tg < 1350tg.$$

For $4 \le n \le 10$, use the smaller separate arithmetic boards and a single tile-label channel in each of the three retained columns. There are 31 ordinary channels: twelve of arity 4, sixteen of arity 6, and three of arity n for $n = 4, 5, 6$, or arity 10 for $7 \le n \le 10$. The non-word tests add at most 2000 equations, so $s \le 3W$ again, where W is the product of the channel alphabet sizes. Four explicit prime tables provide the same positivity, separation, support, and volume certificates as above, with $\ell \le 2^{15}(3W)^2$ and $H^2 \le 2^{15}(3W)^2|V|$. They give the first four rows below. For $11 \le n \le 242$, the padded ternary width is 64, and for $243 \le n \le 251$ it is 80; the existing prime tables at these widths give the last two rows.

n	w	e	lower t	lower g
4	31	6300	3	19/12
5	31	6400	31/10	163/100
6	31	6500	33/10	17/10
$7, \dots, 10$	31	7000	17/5	7/4
$11, \dots, 242$	64	9124	39/10	49/25
$243, \dots, 251$	80	11,678	7	2

Every row satisfies $e \le 1350tg$. The rational logarithm bounds use the same exact power comparisons as before. The four new tables are kernel-checked in `FiniteCompiler/SparseProfile4.lean`, `FiniteCompil`

er/SparseProfile5.lean, FiniteCompiler/SparseProfile6.lean, and FiniteCompiler/SparseProfile10.lean. Their connection to the finite system, geometric construction, and computable finite input tables is checked in FiniteCompiler/SparseSmallSystem.lean, FiniteCompiler/SparseSmallBounds.lean, FiniteCompiler/SparseSmallGeometry.lean, and FiniteCompiler/SparseSmallCompiler.lean. The uniform conversion is checked in Compiler/SparseLogBound.lean. These estimates give the coefficient 1350.

To reduce the coefficient further from 1350 to 1000, restrict the arithmetic components to the column sets

$$I_5 = \{0, \dots, 8, 10\}, \qquad I_7 = \{0, \dots, 9, 14\}.$$

Keep Wang-tile labels at $1, 5, 7$ as before. This leaves 21 arithmetic channels and 3 tile-label channels, instead of 28 and 3. The initial affine classification now uses the first nine columns for $p = 5$ and the first ten for $q = 7$. New kernel-checked SAT certificates force the same clean five-point seed on these shorter boards. The third-row and midpoint implications are checked using only the bounded integer observations that remain inside each board. Four-row propagation needs only eight columns, and horizontal coefficient recovery extends the resulting affine approximation to the isolated column $2r$, for $r \in \{5, 7\}$. The classification is checked in SudokuArithmetic/ShortClassification.lean.

Rescaling needs a different proof that the vertical coefficient stays nonzero. Normalize the first affine approximation to Bm with $B \neq 0$, and write the approximation after $m \mapsto rm$ as $A_1x + B_1m + C_1$. Suppose $B_1 = 0$. On the line $m = x$, the columns $0, r, 2r$ force $C_1 = 0$: if $C_1 \neq 0$, at least two distinct nonzero predicted low digits would both equal C_1. Thus $A_1 \neq 0$. For $j = 1, 2$, use the line $m = x - j$ and the two retained columns $j, j + r$. The predicted low digits differ by B, whereas their actual values both equal $A_1 j \neq 0$. One predicted digit must vanish, so $A_1 j \in \{B, -B\}$. This requires both A_1 and $2A_1$ to belong to $\{B, -B\}$, impossible for $r = 5, 7$. Therefore $B_1 \neq 0$. The existing three-sample argument using $0, r, 2r$ then gives $B_1 = B$. The finite-depth induction and common CRT shear follow as before; these arguments are checked in SudokuArithmetic/ShortArithmetic.lean. All extraction samples $1, 2, 3, 4, r, 2r$ remain available, so the same three Wang decorations yield the tilability equivalence in SudokuArithmetic/ShortBoardSoundness.lean.

Use this smaller system for $4 \leq n \leq 20$. It has 24 ordinary channels: ten of arity 4, eleven of arity 6, and three tile-label channels. The tile-label arity is n for $n = 4, 5, 6, 10$ for $7 \leq n \leq 10$, and 20 for $11 \leq n \leq 20$. The non-word tests still add fewer than 2000 equations, giving $s \leq 3W$. Five explicit prime tables satisfy the same support and geometric inequalities, without changing the geometric construction. Their safe binary exponents are the first five rows below. The last two rows reuse the existing ternary profiles of widths 64 and 80.

n	e	lower t	lower g
4	4700	3	19/12
5	4800	31/10	163/100
6	4900	33/10	17/10
$7, \dots, 10$	5360	17/5	7/4
$11, \dots, 20$	6600	39/10	49/25
$21, \dots, 242$	9124	9/2	13/6
$243, \dots, 251$	11,678	7	2

Every row satisfies $|F| \leq 2^e$ and $e \leq 1000tg$. For the new logarithm witnesses at $n = 21$, use $2^9 \leq 25^2$ and $2^{13} \leq (9/2)^6$. For $n \geq 252$, the existing bulk estimate $(2887/3)tg$ is already less than $1000tg$. The five prime tables are in FiniteCompiler/ShortProfile4.lean, FiniteCompiler/ShortProfile5.lean, FiniteCompiler/ShortProfile6.lean, FiniteCompiler/ShortProfile10.lean, and FiniteCompiler/ShortProfile20.lean. Their computable compiler is checked in FiniteCompiler/ShortSmallCompiler.lean, and the uniform logarithm conversion in Compiler/ShortLogBound.lean.

To reduce the coefficient from 1000 to 850, retain nine columns in each arithmetic component:

$$I_r = \{0, 1, 2, 3, 4, r, r+1, r+2, 2r\}, \qquad r \in \{5, 7\}.$$

This gives 18 arithmetic channels and the same three Wang-tile channels. The Wang decoration at column 5 remains even though this column is absent from the $q = 7$ arithmetic component.

The initial affine classification uses a new finite implication. Normalize one symbol to 1 and consider a seven-row patch of nonzero field symbols on I_r. On each integer line of slope between -6 and 6, impose the first-digit affine rule at the patch cells it meets. Kernel-checked SAT certificates show that the entire patch admits a nonzero affine approximation $Ax + By + C$, with agreement wherever this expression is nonzero. Normalization preserves the line rules and can be undone by multiplying the coefficients by the original symbol.

Adjacent seven-row patches overlap on six rows. Four consecutive rows and the four columns 0, 1, 2, 3 suffice to identify their affine coefficients. Each nonzero affine form vanishes identically on at most one of those rows, so at least two rows have nonzero horizontal forms for both approximations. On each such row, at least two of the four columns have nonzero predictions for both forms. Agreement at those points identifies the horizontal coefficient and row intercept; two row intercepts then identify the vertical coefficient and constant. Thus all patch approximations glue to one global affine approximation. The normalization and gluing are checked in `SudokuArithmetic/NinePatchField.lean`, `SudokuArithmetic/NinePatchOverlap.lean`, and `SudokuArithmetic/NinePatchGlobal.lean`.

These sets retain all rescaling samples $0, r, 2r; 1, 1+r; 2, 2+r$ and extraction samples $1, 2, 3, 4, r, 2r$. The finite-depth proof in `SudokuArithmetic/NineArithmetic.lean` and the Wang-tile extraction in `SudokuArithmetic/NineBoardSoundness.lean` therefore apply unchanged. Use this 21-channel compiler for $4 \le n \le 30$, padding tile labels to 10, 20, or 30 in the indicated ranges. The extra tests number at most $(21+2)^2 + 21 + 4 + 2 \cdot 21 \cdot 30 + 1 + 21 + 2 = 1838 < 2000$, so $s \le 3W$.

Six tables give the first six bounds below; the others use the ternary compiler.

n	e	lower t	lower g
4	4010	3	19/12
5	4100	31/10	163/100
6	4200	33/10	17/10
$7, \dots, 10$	4580	17/5	7/4
$11, \dots, 20$	5670	39/10	49/25
$21, \dots, 30$	6810	9/2	13/6
$31, \dots, 242$	9124	5	23/10
$243, \dots, 1019$	11,678	7	2

Every row satisfies $|F| \le 2^e$ and $e \le 850tg$. For $n \ge 1020$, we have $t \ge 10$ and $g \ge 3$, so $w \le 2t+66 \le (43/5)t$ and $\lg P \le (8/3)g$. Hence

$$\lg |F| \le (34w + 99) \lg P \le \left(34 \cdot \frac{43}{5} + \frac{99}{10}\right) \frac{8}{3}\, tg = \frac{12092}{15}\, tg < 850tg.$$

The six tables are `FiniteCompiler/NineProfile4.lean`, `FiniteCompiler/NineProfile5.lean`, `FiniteCompiler/NineProfile6.lean`, `FiniteCompiler/NineProfile10.lean`, `FiniteCompiler/NineProfile20.lean`, and `FiniteCompiler/NineProfile30.lean`. The compiler and logarithm conversion are checked in `FiniteCompiler/NineSmallCompiler.lean` and `Compiler/NineLogBound.lean`.

To reduce the coefficient from 850 to 800, keep both nine-column arithmetic boards but retain Wang-tile labels only at columns 0, 2. After the common finite-depth shear, put $M = 5^u 7^v$. The line $m = M(1 + 2x)$

compares heights M and $5M$, and $m = M(1+3x)$ compares M and $7M$. Thus the same pair of columns tests both Wang directions.

Here is the shifted-word coefficient recovery. Fix $r \in \{5,7\}$ and $d \in \{2,3\}$. The canonical first digits on the line $m = M(1+dx)$ are $c(1+dx)$, wherever this expression is nonzero, for a nonzero phase c. Comparing the primitive word witness $ax+b$ at the four retained columns $1,2,3,4$ gives $a \equiv cd$ and $b \equiv c \pmod r$. In particular, the witness has valuation zero at column 0. When $r \nmid 1+2d$, its valuation at column 2 is also zero.

In the remaining cases $r = 2d+1$, write $2a+b = rz$. The canonical values at the retained columns 2 and $2+r$ are c and $c(d+1)$, respectively, since $1+2d = r$ and $1+d(2+r) = r(d+1)$, with $0 < d+1 < r$. If $r \mid z$, the word witness at $2+r$ has valuation one and low digit $a \equiv cd$, forcing $cd = c(d+1)$, a contradiction. Hence the witness at column 2 has valuation exactly one. For $d = 2$, the two tile columns therefore read rectangle entries $(0,0),(1,0)$; for $d = 3$, they read $(0,0),(0,1)$. The constant-row argument identifies both decorations at the queried heights, so these comparisons give a common Wang grid. Completeness restricts the original Sudoku array to the retained arithmetic and tile columns. These proofs are checked in `SudokuArithmetic/PairSamples.lean` and `SudokuArithmetic/PairBoardSoundness.lean`.

The resulting system has 20 ordinary channels: nine of arity 4, nine of arity 6, and two tile-label channels. Use it for $4 \le n \le 30$, with the same tile-alphabet padding ranges as above. The extra tests number at most $(20+2)^2 + 20 + 4 + 2 \cdot 20 \cdot 30 + 1 + 20 + 2 = 1731 < 2000$, so the same bound $s \le 3W$ holds.

Six new prime tables give the first six rows; the last two reuse the ternary profiles.

n	e	lower t	lower g
4	3770	3	19/12
5	3890	31/10	163/100
6	3950	33/10	17/10
$7,\dots,10$	4220	17/5	7/4
$11,\dots,20$	4980	39/10	49/25
$21,\dots,30$	5780	9/2	13/6
$31,\dots,242$	9124	5	23/10
$243,\dots,2043$	11,678	7	9/4

Every row satisfies $|F| \le 2^e$ and $e \le 800tg$. For the last logarithm witness, use $2^9 \le 7^4$. For $n \ge 2044$, we have $t \ge 11$ and $g \ge 3$, so $w \le 2t+66 \le 8t$ and $\lg P \le (8/3)g$. Consequently,

$$\lg|F| \le (34w+99)\lg P \le \left(34\cdot 8 + \frac{99}{11}\right)\frac{8}{3}\,tg = \frac{2248}{3}\,tg < 800tg.$$

The $n = 4$ table also improves the useful-prime pairing. The tables are `FiniteCompiler/PairProfile4.lean`, `FiniteCompiler/PairProfile5.lean`, `FiniteCompiler/PairProfile6.lean`, `FiniteCompiler/PairProfile10.lean`, `FiniteCompiler/PairProfile20.lean`, and `FiniteCompiler/PairProfile30.lean`. The compiler and logarithm conversion are checked in `FiniteCompiler/PairSmallCompiler.lean` and `Compiler/PairLogBound.lean`.

Extend the 20-channel construction through $n = 40$. For $31 \le n \le 40$, pad the tile alphabet to 40. The prime profile in `FiniteCompiler/FortyProfile40.lean` gives at most 2^{6586} unit cubes in that range.

With $t = \lg(n+4)$ and $g = \lg t$, the following bounds use the same dyadic caps for $6 \le n \le 30$ and the

ternary construction for $n \geq 41$. Each row satisfies $|F| \leq 2^e \leq 2^{700tg}$.

n	e	lower bound on t	lower bound on g
6	3950	33/10	12/7
$7, \ldots, 10$	4220	69/20	7/4
$11, \ldots, 20$	4980	39/10	49/25
$21, \ldots, 30$	5780	9/2	13/6
$31, \ldots, 40$	6586	5	23/10
$41, \ldots, 242$	9124	49/9	12/5
$243, \ldots, 8187$	11678	7	12/5

For $n \geq 8188$, we have $t \geq 13$, $g \geq 3$, and $w \leq 2t+66 \leq (92/13)t$. Since $P \leq 32t$, it follows that $\lg P \leq (8/3)g$, and the ternary bound gives

$$\lg |F| \leq (34w + 99) \lg P \leq \frac{25816}{39}\, tg < 700tg.$$

The size inequalities are checked in `Compiler/SevenLogBound.lean`.

To reduce the coefficient from 700 to 660, replace the full word constraint by four tests on subsets of columns. Two of them check the two nine-column arithmetic boards separately. The other two check Wang adjacency using the two Wang positions, arithmetic positions $0, 1, 4$ on both boards, and one additional position $2 + r$ on the board for the requested direction ($r = 5$ or 7). At each stage, a test forbids a partial word exactly when it cannot extend to an allowed word. A fixed cycle through the selected channels rules out that partial word.

On a canonical query line $m = M(1 + dx)$ with $d = 0, 2, 3$, the three samples at $0, 1, 4$ have nonzero first digits for both primes. A primitive affine witness has at most one root among the three samples, so two agreements recover its coefficients modulo r. When $r = 2d + 1$, position $2 + r$ distinguishes valuation one from a higher valuation at position 2. The two Wang positions thus recover row equality for $d = 0$, horizontal adjacency for $d = 2$, and vertical adjacency for $d = 3$. This is checked in `SudokuArithmetic/ProjectedBoard.lean` and `FiniteCompiler/ProjectedSystem.lean`.

With Wang alphabet size D, the number of word tests is at most $4^9 + 6^9 + (4^4 6^3 + 4^3 6^4)D^2$. Allowing for the other tests and one spare index gives $X = 4^9 + 6^9 + (4^4 6^3 + 4^3 6^4)D^2 + 80D + 550$. The same geometry bound applies with the stacking cap at this X. Using the checked prime profiles, including a new assignment at $n = 6$, gives the following dyadic bounds $|F| \leq 2^e \leq 2^{660tg}$.

n	D	e	lower bound on t	lower bound on g
6	6	3788	83/25	173/100
7	7	3912	17/5	7/4
$8, \ldots, 10$	10	4065	7/2	9/5
$11, \ldots, 20$	20	4862	39/10	49/25
$21, \ldots, 30$	30	5663	9/2	13/6
$31, \ldots, 40$	40	6476	5	23/10
$41, \ldots, 44$	44	6807	5	23/10
$45, \ldots, 242$	—	9124	28/5	247/100
$243, \ldots, 8187$	—	11678	7	11/4

The last two rows use the ternary compiler. For $n \geq 8188$, we have $t \geq 13$, $g \geq 11/3$, and $P \leq 20t$, so $\lg P \leq g + 9/2 \leq (49/22)g$. The existing bound $(34w + 99) \lg P$ on $\lg |F|$ is then at most $(68 + 2343/13)(49/22)tg < 660tg$ using $w \leq 2t + 66$. To reduce the coefficient further to 635, we use 460 forbidden partial patterns on the 5-board and 1853 on the 7-board, each specifying three to five columns. Every disallowed arithmetic word matches at least one pattern, and every allowed word avoids all of them. Lean checks the

arithmetic equivalence using finite computations and unsatisfiability certificates in `SudokuArithmetic/PartialWordCertificate5.lean` and `SudokuArithmetic/PartialWordCertificate7.lean`. For each of the three canonical query slopes, we test just the arithmetic patterns that recover the two Wang symbols at the required valuations. There are $(5-1)(7-1) = 24$ choices of the nonzero arithmetic digits at height M, so the Wang rules require at most $72D^2$ test prototiles. The implications and their soundness are checked in `SudokuArithmetic/QueryGuards.lean` and `FiniteCompiler/GuardedSystem.lean`.

We also reduce the seed modulus from 175 to 35 in this branch. The first seed allows digits $1, 2$ modulo 5 and $1, 2, 3$ modulo 7, while the second allows the remaining nonzero digits. Thus activation of both seeds still forces two distinct values in every arithmetic column. The first seed can use three residue points $(0, 1)$, $(0, 2)$, $(0, 16)$, and the second can use four $(0, 4)$, $(0, 13)$, $(0, 18)$, $(0, 19)$. Their complement has 1218 points, a multiple of 6, so the same shared seed labeling works. Inactivity would require writing $35^2 = au + bv$ for nonnegative integers u, v, where a, b are the useful primes for that seed. We exclude this by choosing $a = 2$ and $b > 1225$ for the first seed, and $a = 3$, $b \equiv 2 \pmod 3$ and $2b > 1225$ for the second. See `FiniteCompiler/GuardedSeeds.lean`.

The new tests give $X = 2313 + 72D^2 + 80D + 550$. With $\ell = \lfloor \lg(2X) \rfloor + 1$ and prime bound $P = 65{,}536$, the stacking modulus is at most $2(6X^2\ell + P + 3)$. For a prime $2m+1$ above $6X^2\ell + P + 3$, the probability of missing any required difference is bounded by $s(2m)(1 - 1/s^2)^m$ for $s \le X$ colors. The exponential factor is at most $e^{-3\ell}$, and the prefactor is less than 16^ℓ for the chosen prime, so this probability is less than one. We count the final rail shell directly from its side lengths and use the box, bump, and frame counts for the quotient-to-lattice pieces. See `Proofs/TightDifferenceBound.lean` and `FiniteCompiler/GuardedVolume.lean`.

The resulting profiles give $|F| \le 2^e \le 2^{635tg}$.

n	D	e	lower bound on t	lower bound on g
6	6	3651	93/28	71/41
7	7	3779	24/7	7/4
$8, \dots, 10$	10	3932	7/2	9/5
$11, \dots, 20$	20	4731	31/8	25/13
$21, \dots, 30$	30	5533	9/2	2
$31, \dots, 40$	40	6348	5	2
$41, \dots, 44$	44	6680	5	9/4
$45, \dots, 48$	48	7449	11/2	7/3
$49, \dots, 242$	—	9124	63/11	73/29
$243, \dots, 8187$	—	11678	7	8/3

The ternary estimate above also gives $(68 + 2343/13)(49/22)tg < 635tg$ for $n \ge 8188$.

To reduce the coefficient from 635 to 410, we use fewer and shorter corridors in the connectedness construction. Write $R = \prod_i r_i$ for the product of the channel digit products, N for the cyclic order before stacking, and q for the stacking prime. The kernel test includes every multiple of R in $\mathbb{Z}/N\mathbb{Z}$ at horizontal position zero. Any stacking layer assigned to the kernel test therefore contains a whole coset of the multiples of Rq in $\mathbb{Z}/(Nq)\mathbb{Z}$. After taking lattice representatives, both the main components and their displaced marker frames supply points along this column, spaced $201Rq$ apart in height. Each cube lies within $201Rq + 400$ in height of one of these points, so we can connect it to the column and join the successive points using a rail of height $201Rq + 401$.

The required horizontal displacements have $|x| \le 201H + 465$, where H bounds the horizontal coordinates of all test prototiles. A seed or dependence offset fits in $|y| \le 201 \cdot 175 + 400$, and an ordinary activation offset of slope s fits in $|y + sx| \le 465s + 400$ for $0 \le s \le 16$. These bounds include the offsets inside each lattice component. Each strip is connected to the origin by horizontal and vertical steps, so we can use their union as the horizontal slot set. This uses at most $a = (402H + 931) \cdot 210{,}447$ horizontal slots. For the

existing stacking cap L on q and mass bound J, we use $B = 201RL + 401$ in the shell estimate and obtain at most

$$(201^3 + 3 \cdot 7^3 + 11^3)JL \cdot 216(4a + 7(B + 2) + 1)(4a + 1)$$

unit cubes. See `Rails/Strips.lean`, `FiniteCompiler/StripStack.lean`, `FiniteCompiler/StripLift.lean`, and `FiniteCompiler/StripVolume.lean`.

With the same prime profiles, the bounds are $|F| \le 2^e \le 2^{410tg}$.

n	e	lower bound on t	lower bound on g
6	2355	83/25	45/26
7	2438	17/5	7/4
$8, \dots, 10$	2536	7/2	9/5
$11, \dots, 20$	3041	27/7	25/13
$21, \dots, 30$	3544	9/2	2
$31, \dots, 40$	4055	5	2
$41, \dots, 44$	4263	5	9/4
$45, \dots, 48$	4741	11/2	7/3
$49, \dots, 242$	5856	40/7	5/2
$243, \dots, 8187$	7488	7	8/3

For $n \ge 8188$, the ternary branch satisfies $R \le P^{9w+8}$, where w counts its channels and $P = \max(w + 2, 256)$. Applying the previous coarse estimate on the horizontal area of the shell with $\max(H^2, Rq)$ in the vertical factor gives $|F| \le P^{21w+78}3^{4w+4} \le P^{22w+79}$. Using $w \le 2t + 66$, $t \ge 13$, and $\lg P \le (49/22)g$ gives $\lg|F| \le (103047/286)tg < 410tg$. See `FiniteCompiler/ProjectedCompiler.lean`, `FiniteCompiler/Effective/SparseStrip.lean`, and `Compiler/ProjectedLogBound.lean`.

To reduce the coefficient from 410 to 401, we reserve layers indexed by $3Rj \bmod q$ for the kernel test, for $0 \le 3j < q$. For each nonzero difference there is a matching of $(q - 1)/2$ disjoint pairs realizing it. Discarding pairs that meet reserved layers leaves at least $(q - 7)/6$ pairs. The same counting argument on the free layers finds a partition whose color classes all realize every difference. With X bounding the number of tests and P bounding the original prime factors, we take $q > 20X^2(\lfloor \lg(2X) \rfloor + 1) + P + 3$ and at most twice that threshold. The ternary branch can use its existing upper bound on q. After stacking, the resulting column has gaps at most $3R$ around the cycle. The lifted pieces can therefore use a rail of height $201(3R + 1) + 401$. See `Proofs/ReservedDifferenceBound.lean`, `FiniteCompiler/Effective/ReservedStack.lean`, and `FiniteCompiler/ReservedStripLift.lean`.

In the smaller-input compiler, each ordinary activation offset is a Chinese-remainder representative plus at most one copy of its modulus $r_i a_i b_i$. In each fixed slice, doubling the ranks of the shift copies preserves their permutation because the ordinary useful primes are odd. The residues and the copy flag then distinguish every shift copy. For each slope s we bound the horizontal radius using only its own channels. The seed and dependence offsets fit in the slope-zero strip. We use the sum of these strip sizes in the same shell estimate. See `FiniteCompiler/CompactShift.lean` and `FiniteCompiler/ProjectedCompiler.lean`.

With the same prime profiles, these estimates give $|F| \le 2^e \le 2^{401tg}$ using the following certificates.

n	e	lower bound on t	lower bound on g
6	2307	289/87	168/97
7	2389	24/7	7/4
$8, \dots, 10$	2487	7/2	9/5
$11, \dots, 20$	2988	31/8	25/13
$21, \dots, 30$	3489	9/2	2
$31, \dots, 40$	3999	5	2
$41, \dots, 44$	4206	5	9/4
$45, \dots, 48$	4679	11/2	7/3
$49, \dots, 242$	5636	40/7	5/2
$243, \dots, 8187$	7218	7	8/3

The same coarse estimate for $n \ge 8188$ gives $\lg |F| \le (103047/286)tg < 401tg$.

The guarded compiler keeps the eighteen arithmetic columns and encodes each Wang label by several digits sharing its original slope. A forbidden partial pattern specifies every digit of each label it mentions, so a single cycle tests the expanded pattern. Decoding commutes with sampling a line because all digits of a label use the same slope. The seeds impose their predicates on the arithmetic columns and allow every Wang digit. See `FiniteCompiler/ColumnRefinement.lean` and `FiniteCompiler/GuardedRefinement.lean`.

Through $n = 3^{21}$, we use the following radices and checked prime tables for the guarded construction. The volume checks give $|F| \le 2^e$. The first two rows fit the fixed term 2^{2392}, and the other rows fit $(n+4)^{10\lg\lg(n+4)+19\lg\lg\lg(n+4)+557}$.

n	radices per label	e
1	(1)	2112
$2, \dots, 12$	(3, 4)	2392
$13, \dots, 15$	(3, 5)	2438
$16, \dots, 36$	(3, 3, 4)	2573
$37, \dots, 242$	5 ternary digits	2876
$243, \dots, 3^9$	9 ternary digits	3594
$3^9+1, \dots, 3^{21}$	21 ternary digits	5954

See `FiniteCompiler / PackedCompiler.lean`, `FiniteCompiler / PackedExtendedProfiles.lean`, and `Compiler/GuardedFiniteBound.lean`.

For $n > 3^{21}$, let $k = \lceil \log_4 n \rceil$. Each of the two Wang labels uses k base-four digits, so there are $2k$ Wang columns. We retain the checked primes for the eighteen arithmetic columns and the two seed channels, reserving their primes together with 5 and 7. Write $\Pi(s) = 4s(1 + \lfloor \lg s \rfloor)$, so $\pi(\Pi(s)) \ge s$ for $s \ge 16$. For each Wang column we first reserve a useful prime $b \le x = \Pi(2k + 134)$. The cap $P = \Pi(4x + 12k + 134)$ then leaves more than $4b$ unused primes. By the pigeonhole principle, five of them have the same residue modulo b. The smallest is the column's other useful prime a, and the other four serve as digit primes, each of the form $a + vb$ with $v > 0$. See `Compiler/GuardedPrimeCounting.lean` and `FiniteCompiler/Effective/ResidueRows.lean`.

Let R be the product of the digit primes and U the product of the useful primes, so the mass is $J = 1225RU$. Each Wang column contributes a factor of at most P^9x to RJ: four digit primes each appear twice, and each useful prime appears once. The checked arithmetic and seed rows contribute a factor of at most 2^{1878}. With $t = \lg(n+4)$ and $g = \lg t$, the prime caps satisfy $\lg x \le g + \lg g + 6$ and $\lg P \le g + 2\lg g + 13$. The $2k \le t + 2$ Wang columns therefore give the terms $10g + 19\lg g$ in the exponent of $n + 4$.

We use a test cap of $2048(n+4)^2$, giving a reserved stacking prime cap L with $\lg L \le 4t + g + 29$. The strip radii are at most $2^{70} + 2P^5x$, so the number of horizontal slots is at most $A = 2^{100}P^5x$. The

volume bound $2^{49}RJLA^2$ then gives $\lg|F| \le t(10g + 19\lg g + 557)$. See `Compiler/ResidueLogBound.lean` and `FiniteCompiler/Effective/PackedTailCore.lean`. □

The full construction is in `Compiler/CleanQuasipolynomialBound.lean`, proved by `Compiler.Size.exists_connected_tile_clean_quasipolynomial_bound`. The generalized graph, dependence, fixed cycle, and activation equations are in `FiniteCompiler/`; their common-solution and soundness proofs allow repeated column slopes and channel-dependent finite alphabets. The shorter-board equivalence is `Sudoku.PairBoard.wang_iff` in `SudokuArithmetic/PairBoardSoundness.lean`. `FiniteCompiler/Effective/PackedTotalCompiler.lean` selects the prime table by alphabet size and generates the primes and tests for larger inputs. The public theorem includes nonemptiness, face connectedness, both tilability directions, and the numerical bound. Duplicate input entries have no cost in the existence bound. The explicit bounds in Corollary 6.8 remain available. The sharper count for one Wang tile is `Compiler.Size.exists_connected_tile_one_quasipolynomial_bound`.

6.4.2 Sharing a Useful Prime Across Wang Columns

For $n > 3^{21}$, the compiler shares one useful prime per Wang slope. The uniform bound is

$$B(n) = \max\left\{(n+4)^{8.83302(\lg\lg(n+4)+\lg\lg\lg(n+4))+570}, 2^{2392}\right\}.$$

The computable compiler is in `FiniteCompiler/Effective/SharedTailCore.lean`, with its correctness and size bound proved in `FiniteCompiler/Effective/SharedDeckCorrect.lean`.

Use $k = \lceil \log_3 n \rceil$ ternary digits per Wang label. At slope $s \in \{0, 2\}$, share one useful coordinate α_s of prime order a_s, with $a_0 = 199$ and $a_2 = 163$. Each Wang column i keeps its private useful coordinate ξ_i of prime order b_i and its three digit primes, whose product is r_i. Its dependence and cycle tests read ξ_i. Its activation tile reads the pair (α_s, ξ_i) and the high coordinate for column i. Two extra dependence tests make each α_s constant along its slope for each fixed vector of low coordinates. All arithmetic and seed coordinates retain their private pairs.

For soundness, fix a line and the low coordinates outside column i. If ξ_i were constant there, the activation histogram would be supported on one value of its second coordinate. The activation shift multiplicities force its mixed difference to vanish. The histogram is therefore uniform in the first coordinate on its support, implying $a_s \mid r_i$. The primes are distinct, so each line has an active digit for the private coordinate, which is the coordinate used in the cycle tests. For completeness, take α_s to be the sum of the first block coordinates of the columns at slope s. Within the activation test for column i, the other summands are constant because those columns have the same slope and their low coordinates stay fixed. The existing block inverse works after translating the shared target by that constant. A positive number of blocks of size b_i ensures the private coordinate detects the designated digit.

Write $x = \Pi(2k + 136)$, and choose the $2k$ private useful primes at most x, excluding the fixed primes. The explicit Chebyshev bounds supply the $6k$ digit primes in $(200x, 1600x]$. Each exceeds $a_s b_i$, so it is a positive integer combination of a_s and b_i. With $P = 1600x$, the cyclic order is at most $2^{1893}P^{12k}x^{2k}$. Each Wang column contributes three squared digit primes and one private useful prime. Both shared primes are in the fixed factor. Consequently the leading coefficient is $2(2 \cdot 3 + 1)/\lg 3 = 14/\lg 3$, rounded up to 8.83302 in the displayed bound. The strip radius is at most $2^{70} + 400P^3x$, and the test cap is $2048(n+4)^2$. Each test individually complements a graph, so its mass equals the size of one fibre of the low-coordinate projection. The stacking and assembly estimates give the displayed bound for $n > 3^{21}$. For the other inputs the guarded prime tables above still fit, including the sharper 2^{2112} count when $n = 1$.

6.4.3 The Remaining Cost of Cyclicity

Distinct private useful primes already incur a quasipolynomial cost. For the variant with two per channel, let C be the number of channels and H_0 the graph-kernel prototile. Its useful-output space embeds in H_0: set all other coordinates to zero. Consequently,

$$|H_0| \geq \prod_i a_i b_i \geq (2C+1)!.$$

For the second inequality, sort the $2C$ distinct useful primes; the kth entry is at least $k+1$. These inequalities are checked in `FiniteCompiler/SizeObstruction.lean` by `FiniteCompiler.factorial_le_kernelTile_card`. Since $C = \Theta(\lg n)$ in this channel construction, even this intermediate prototile has $n^{\Omega(\log\log n)}$ elements. The shared-prime variant still has $2k$ distinct private Wang useful primes, so their product alone gives the same asymptotic cost. Thus polynomial size requires changing the encoding. This is a limitation of the present construction, with no corresponding lower bound for arbitrary reductions.

A possible change is to reuse a fixed prime table across ordinary ternary channels. At the level of the size estimates, let every such channel have fixed moduli a, b, r, and keep the two seed channels fixed. With $m = Nk$ ordinary channels the finite factor would have order

$$|V| = D(r^2ab)^m \prod_{t=1}^{2} r_{\eta_t}^2 a_{\eta_t} b_{\eta_t} = n^{O(1)}.$$

The fixed-test count would also be $n^{O(1)}$, because $m = O(\lg n)$. However, this finite factor is noncyclic; its quotient $(\mathbb{Z}/a\mathbb{Z})^m$ already needs m generators when a is prime. The present quotient-to-lattice step, which uses one finite cyclic coordinate and two infinite coordinates, therefore does not apply. A polynomial conversion of this shared-prime system into a cyclic system remains an open step; the displayed estimates do not assert a polynomial three-dimensional construction.

6.5 Why the Dimension Is Optimal

Proposition 6.11 (Bhattacharya [Bha20, Corollary 1.2], Greenfeld–Tao [GT21, Corollary 1.6])**.** *Given a finite nonempty $F \subset \mathbb{Z}^2$, it is decidable whether F tiles $\mathbb{Z}^2$ by translations.*

Proof. Bhattacharya [Bha20, Theorem 1.1] proved that every such prototile admitting a tiling also admits a tiling invariant under a finite-index subgroup of $\mathbb{Z}^2$, extending Beauquier and Nivat's earlier result for simply connected polyominoes [BN91]. Every finite-index subgroup contains $m\mathbb{Z}^2$ for some positive integer m. Run two searches in parallel. The first enumerates m and all Boolean assignments on $(\mathbb{Z}/m\mathbb{Z})^2$, checking the periodic versions of (6.6). It halts if a periodic tiling exists. The second searches for an unsatisfiable finite subsystem, as in the membership argument. It halts if no tiling exists. The periodicity theorem ensures that exactly one search eventually halts, so this is a decision algorithm. Greenfeld and Tao [GT21, Corollary 1.6] later gave a quantitative version with an explicit bound on the running time. Planar periodicity is `planarPeriodicity_proved` in `Proofs/PlanarPeriodicity.lean`; the two searches are combined in `planar_decidable` in `Proofs/PeriodicCertificates.lean`. □

The one-dimensional case reduces to this planar problem by replacing $F \subset \mathbb{Z}$ by $F \times \{0\} \subset \mathbb{Z}^2$. For every fixed $d > 3$, the three-dimensional hardness result extends by replacing F with $F \times \{0\}^{d-3}$ and taking slices. Thus three is precisely the least lattice dimension with undecidable translational monotiling, even with a connected input prototile. The dimension statement is `dimension_optimality` in `Statements.lean`.

6.6 Arbitrary Real Translations of Unit-Cube Unions

We now prove the Euclidean formulation using the tiling convention from Section 1. For $v \in \mathbb{R}^d$, write $\|v\|_\infty = \max_j |v_j|$ for the ***sup norm***. Discrete-to-Euclidean tiling reductions also appear in [GT24, Section 2]; the unit-cube existence equivalence needed here has the direct sampling proof below.

Lemma 6.12. *For every finite nonempty $F \subset \mathbb{Z}^d$, the set $\Omega = F + [0,1]^d$ tiles $\mathbb{R}^d$ by real translations if and only if F tiles $\mathbb{Z}^d$ by integer translations.*

Proof. An integer tiling thickens to a Euclidean tiling. Conversely, suppose translates $a + \Omega$, $a \in A \subset \mathbb{R}^d$, form an almost-everywhere tiling. Distinct points of A have sup-norm distance at least one, since otherwise translates of a fixed unit cube of Ω overlap in positive volume. Hence A is countable.

Choose $\theta \in (0,1)^d$ so that every point of $\theta + \mathbb{Z}^d$ has coverage multiplicity one and $\theta_j \not\equiv a_j \pmod 1$ for every $a \in A$ and every coordinate j. Such a choice exists: the excluded coverage sets, translated by integer vectors, form a countable union of null sets; the excluded coordinate hyperplanes are also countably many null sets. Define $b(a) \in \mathbb{Z}^d$ by $b(a)_j = \lceil a_j - \theta_j \rceil$. Intersecting a translate with the sampled grid gives exactly

$$(a + \Omega) \cap (\theta + \mathbb{Z}^d) = \theta + b(a) + F.$$

Every sampled point lies in exactly one such set. The map $a \mapsto b(a)$ is injective, because equal rounded vectors would make two nonempty sampled copies coincide. Consequently $\{b(a) : a \in A\} \oplus F = \mathbb{Z}^d$. This is `realTiles_iff_tiles` in `Proofs/Euclidean.lean`. □

Corollary 6.13. *Given a finite nonempty $F \subset \mathbb{Z}^3$, the problem of deciding whether its unit-cube union $F + [0,1]^3$ tiles $\mathbb{R}^3$ by arbitrary real translations is co-RE-complete. This remains true when F is required to be connected, in which case its unit-cube union is connected and has connected interior. For the analogous class of unit-square unions in $\mathbb{R}^2$, the problem is decidable.*

Proof. Apply Lemma 6.12 to Theorem 1.1 and Proposition 6.11. The lattice paths in F give chains of cubes sharing faces, so both the union and its interior are connected. Connected Euclidean completeness is stated as `Abstract.real_tiling_coRE_complete` in `Abstract.lean`. □

The planar Euclidean decidability conclusion is also known directly from [dDPGGM26, Corollary 1.1].

Remark 6.14. The sampling argument produces a new tiling with integer translations. It does not assert that the original set of real translations was integral, and no preservation of its period group is used.

7 Allowing Grid Rotations

A ***grid isometry*** is a signed permutation of the three coordinates, followed by an integer translation. Of the $3!2^3 = 48$ signed permutations, the 24 with determinant 1 are the ***proper grid rotations***; the others reverse orientation. In this section, a voxel is a unit cube centered at an integer lattice point, and connectivity means face connectivity. We construct a fixed replacement that forces every copy in a tiling to have one common orientation and one common grid, even when all 48 signed permutations are allowed.

7.1 The Six-Key Prototile

Let $Q = \{0, \dots, 39\}^3$. Index its faces by $-x, +x, -y, +y, -z, +z$. For face i, let n_i be its inward unit normal, and let u_i, v_i be the other two positive coordinate unit vectors, in coordinate order. The ***gate*** g_i has normal

coordinate 0 on a negative face and 39 on a positive face; both tangential coordinates are 12 on a negative face and 13 on a positive face. Choose dimensions

i	d_i	a_i	b_i
$-x$	3	8	15
$+x$	3	10	12
$-y$	4	6	15
$+y$	4	9	10
$-z$	5	6	12
$+z$	5	8	9

and remove the ***recess***

$$D_i = \{g_i + kn_i : 0 \le k \le 4\} \ \cup \ \{g_i + (5+k)n_i + uu_i + vv_i : 0 \le k < d_i,\ 0 \le u < a_i,\ 0 \le v < b_i\}.$$

Thus D_i is a five-voxel neck followed by a rectangular box. Every box has 360 voxels, so $|D_i| = 365$. Write $K_i = D_i \setminus \{g_i\}$ for the ***chamber***, with 364 voxels. The body and the replacement prototile are

$$B = Q \setminus \bigcup_i D_i, \qquad T = B \ \cup \ \bigcup_i (40n_i + D_i).$$

The translated recesses are the six protruding ***keys***. Each meets the body through its translated gate, by a single edge to $40n_i + g_i - n_i \in B$. The different tangential positions of opposite gates ensure that this attachment voxel remains in the body.

The recesses are mutually separated, as are the protruding keys. Moving each recess by a vector in $40\mathbb{Z}^3$ preserves every residue class, so T is a fundamental domain for $40\mathbb{Z}^3$ and $|T| = 40^3 = 64{,}000$. The definitions are in `Rotations/Geometry.lean`; the fundamental domain and exact cardinality are proved in `Rotations/FundamentalDomain.lean` and `Rotations/Volume.lean`.

7.2 Recognizing a Whole Key

Lemma 7.1. *The body B remains connected after deleting any one voxel. If a connected component of $T \setminus \{v\}$ has 364 voxels, then $v = 40n_i + g_i$ for some i, and that component is $40n_i + K_i$. Moreover, if $t + RK_i = K_j$ for a signed coordinate permutation R, then $i = j$, R is the identity, and $t = 0$.*

Proof. The six coordinate planes $x_j = 10, 29$, restricted to Q, avoid all recesses. Their union stays connected after one voxel deletion: each square plane stays connected, and intersecting planes share a full line of voxels. Every body voxel outside these planes has two paths to them with no common vertex except their starting point. If a coordinate lies strictly between 10 and 29, a whole clear coordinate segment through the voxel joins two of the planes. In a layer near just one face, the removed cross-section is one rectangle or one point; a retained point therefore has a clear tangential row or column. If all three coordinates are outside the central range, use two coordinate rays toward the nearest planes. At least one route survives any prescribed deletion. These path arguments are formalized in `Rotations/BodyPaths.lean`, `Rotations/PlanePaths.lean`, and `Rotations/BodyConnectivity.lean`.

A component of size 364 cannot meet the surviving body, which has at least 61,809 voxels. It is therefore contained in one protruding key. If the deleted voxel is outside that key, connectivity forces the component to contain all 365 key voxels. If the deleted voxel belongs to the key but is not its gate, the gate remains attached to the body, leaving at most 363 voxels in a detached component. The only remaining possibility is deletion of the gate, exposing the corresponding chamber. The general counting argument is `Rotations.KeyedBody.recognize` in `Rotations/KeyRecognition.lean`, specialized to the explicit gadget in `Rotations.SixKeys.recognize_component` in `Rotations/Gadget.lean`.

To recognize the chamber shape, consider all eight-voxel $2 \times 2 \times 2$ blocks contained in it. Their union is exactly its rectangular box: no neck voxel belongs to such a block. The six boxes have different unordered dimension triples. Each box has three distinct side lengths, which fix the axes, and the neck's direction and corner attachment fix their signs. Thus the chamber determines its label and complete orientation. Lean verifies this conclusion using the translation-invariant sums

$$(\mu_k(K))_j = \sum_{x \in K} \left(|K| x_j - \sum_{y \in K} y_j \right)^k, \qquad k = 2, 3.$$

Their coordinates permute under a signed coordinate permutation; odd powers also acquire its coordinate signs. The computed values distinguish the six chambers and all their orientations. This is `Rotations.SixKeys.chamber_rigid` in `Rotations/KeyRigidity.lean`, using the kernel-checked calculations in `Rotations/KeyMoments.lean`. □

7.3 Forcing the Whole Grid

Theorem 7.2. *Every tiling of $\mathbb{Z}^3$ by grid-isometric copies of T has one common orientation and one common coset of the corresponding $40\mathbb{Z}^3$ grid.*

Proof. Normalize one placed copy to be T. Every face-neighbor of K_i outside K_i belongs to this copy, except for g_i. Any other copy meeting K_i must leave it, because it is connected and has $64{,}000 > 364$ voxels. Its path out must pass through g_i. Exactly one copy covers g_i, so one copy fills all of K_i. Deleting g_i from that copy makes K_i a whole connected component. This single-entrance argument is `Rotations.chamber_component` in `Rotations/Chambers.lean`.

By Lemma 7.1, this component is a protruding key of the same label and in the same orientation. Its translation is consequently $-40n_i$. All six neighboring grid copies are forced. Iteration forces every $40z + T$, $z \in \mathbb{Z}^3$. These copies already partition space, so every copy in the original tiling is one of them. No periodicity assumption on the original tiling is needed. The local and global results are `Rotations.SixKeys.chamber_forces_neighbor` in `Rotations/LocalForcing.lean` and `Rotations.SixKeys.rigidity` in `Rotations/GlobalRigidity.lean`. □

7.4 The Rotation-Allowed Reduction

For a finite nonempty $E \subset \mathbb{Z}^3$, define

$$P(E) = \bigcup_{e \in E} (40e + T).$$

The union is disjoint, and $|P(E)| = 64{,}000|E|$. If E is connected, so is $P(E)$: T is connected, and neighboring grid copies meet along faces at the intact cube edges.

Theorem 7.3. *Deciding whether one finite face-connected polycube tiles $\mathbb{Z}^3$ using integer translations and proper grid rotations is co-RE-hard, and hence undecidable. The same conclusion holds when all signed coordinate permutations, including reflections, are allowed.*

Proof. A translational tiling by E expands to a translational tiling by $P(E)$. Conversely, decompose any grid-isometric tiling by $P(E)$ into its constituent copies of T. By Theorem 7.2 these copies share one orientation and one grid. Undo that orientation and identify their grid centers with $\mathbb{Z}^3$. Their grouping into parent copies of $P(E)$ gives a translational tiling by E. Thus

$$E \text{ tiles by translations} \quad \Longleftrightarrow \quad P(E) \text{ tiles with proper grid rotations.}$$

The reverse direction even allows all 48 signed coordinate permutations; the forward direction uses only the identity. Apply this replacement to the connected output of Theorem 1.1. The map is computable: replace each input voxel by the same fixed list of 64,000 voxels, scaled and translated as above. The tiling equivalence and connectedness are `Rotations.SixKeys.assembly_allowed_iff` in `Rotations/Assembly.lean` and `Rotations.SixKeys.assembly_connected` in `Rotations/ConnectedAssembly.lean`. The computable reduction, co-RE-hardness, and undecidability are `Rotations.SixKeys.rotationCompile_correct` in `Rotations/Main.lean`, `Rotations.SixKeys.proper_rotation_coRE_hard`, and `Rotations.SixKeys.proper_rotation_undecidable` in the same file. □

8 Allowing Arbitrary Euclidean Rotations

The same six-key replacement also handles arbitrary angles and arbitrary real translations. For a finite lattice set S, write

$$\Omega(S) = \bigcup_{p \in S} \left(p + [-\tfrac{1}{2}, \tfrac{1}{2}]^3\right).$$

A ***Euclidean motion*** is a map $x \mapsto t + Rx$, where $t \in \mathbb{R}^3$ and R is an orthogonal matrix, meaning $R^{\mathsf{T}}R = I$. It is ***proper*** when $\det R = 1$; otherwise it reverses orientation. As in Section 6.6, tiling means coverage with multiplicity one almost everywhere, so shared boundary points are permitted. All interiors and boundaries in this section are taken in $\mathbb{R}^3$.

8.1 Aligning a Cube Inside a Neck

Lemma 8.1. *If an orthogonally rotated unit cube fits inside a coordinate-aligned square prism with cross-section 1×1, its orientation is a signed coordinate permutation. Its center lies on the prism's central axis.*

Proof. For a matrix row r, the cube's width in that coordinate is $\|r\|_1 = \sum_j |r_j|$, while orthogonality gives $\|r\|_2 = (\sum_j r_j^2)^{1/2} = 1$. The two transverse widths are at most 1. Equality between these two norms forces at most one nonzero entry in each transverse row, because

$$\left(\sum_j |r_j|\right)^2 - \sum_j r_j^2 = 2 \sum_{j<k} |r_j r_k|.$$

Each transverse row therefore has one entry equal to 1 or -1. Their orthogonality puts these entries in different columns, and forces the remaining row onto the remaining coordinate axis. The transverse widths are exactly 1, so fitting them inside the prism also fixes the two transverse center coordinates. The matrix argument and cube geometry are proved in `Rotations/Euclidean/MatrixAlignment.lean` and `Rotations.Euclidean.cube_alignment` in `Rotations/Euclidean/CubeAlignment.lean`. □

For a key on face i, use local coordinates (k, u, v) through the map $g_i + kn_i + uu_i + vv_i$. Its neck voxels have centers $(0,0,0), \ldots, (4,0,0)$. The neck's solid cross-section is $[-\tfrac{1}{2}, \tfrac{1}{2}]^2$. In the neighborhood

$$-\tfrac{1}{2} < k < \tfrac{9}{2}, \qquad |u|, |v| < \tfrac{5}{2},$$

every point outside this closed cross-section lies in the interior of the host body $\Omega(B)$. This includes points on seams between body voxels: all lattice voxels incident to such a point are present. These wall and seam claims are `Rotations.Euclidean.neck_wall` in `Rotations/Euclidean/NeckWalls.lean` and `Rotations.Euclidean.star_interior` in `Rotations/Euclidean/VoxelStars.lean`.

8.2 Filling a Chamber Without a Grid Assumption

Lemma 8.2. *In any Euclidean tiling by $\Omega(T)$ containing the identity copy, the copy $-40n_i + \Omega(T)$ is present for every face i. This conclusion allows arbitrary orthogonal orientations and real translations for every other copy.*

Proof. Distinct tiles have disjoint interiors. Each has a nonempty open interior, so the family of copies is countable. Consequently the union of all placed voxel boundaries has measure zero. We may choose a point p in the open middle neck

$$\tfrac{7}{4} < k < \tfrac{9}{4}, \qquad |u|, |v| < \tfrac{1}{2}$$

that lies in the interior of a voxel of its covering tile, say A. The host does not cover this point. Every point of that voxel is within distance $\sqrt{3} < 2$ of p, so the whole voxel lies in the neighborhood where the neck walls apply. Its closed cube cannot meet the host's interior. It therefore fits inside the neck prism. By Lemma 8.1, A has a signed grid orientation relative to the host, and this voxel's transverse center coordinates are zero. In particular, both transverse translation coordinates of A are integers. This is `Rotations.Euclidean.neck_copy_alignment` in `Rotations/Euclidean/NeckCopy.lean`.

Put $h = k(p)$ and let

$$U = \operatorname{int} \Omega(K_i) \cap \{k > h\}$$

be the chamber on the far side of the cut through p. Its boundary lies in the host solid together with the interior of the blocking voxel of A. Indeed, the original chamber has only its gate as an entrance; that gate is below the cut. The cut's open square lies inside the blocking voxel, and the square's edges lie in the host solid. The passage from a lattice chamber boundary to its solid boundary is `Rotations.Euclidean.frontier_single_entrance` in `Rotations/Euclidean/VoxelStars.lean`; the cut is handled in `Rotations/Euclidean/CappedChambers.lean`.

Consider another tile with an interior voxel point in U. That point can be joined to every voxel center through the tile's interior, following its face-adjacency paths. Such a path cannot cross ∂U, because another tile's interior avoids both the closed host solid and the interior of A. Hence all its voxel centers would lie in U. This is impossible: T contains the centers $(0, 0, 0)$ and $(39, 0, 0)$, at distance 39, whereas the chamber is contained in a box with all three side lengths at most 15, of diameter at most $\sqrt{675} < 39$. Thus only A can fill U almost everywhere. Since U is open and A is closed, $U \subset A$ and $\overline{U} \subset A$. The general separation argument is `Rotations.Euclidean.chamber_monopoly` in `Rotations/Euclidean/Monopoly.lean`.

The far face point $(d_i + \frac{9}{2}, 0, 0)$ belongs to $\overline{U}$, and hence to a voxel of A. Its transverse center coordinates are integers within $\frac{1}{2}$ of zero, so both are zero. Its axial center c satisfies $d_i + 4 \le c \le d_i + 5$. The host contains the wall voxel centered at $(d_i + 5, 0, 0)$. If $c > d_i + 4$, these two voxels overlap in their interiors. Therefore $c = d_i + 4$, and the axial translation coordinate of A is also integral. This pins all three translation coordinates; see `Rotations.Euclidean.phase_of_end_contact` in `Rotations/Euclidean/CapForcing.lean`.

We can now use lattice connectivity. The tile A contains every chamber voxel with $k \ge 3$, because their centers lie in U. Its original blocking voxel has center $(2, 0, 0)$, since that center is integral and its interior contains p. The only remaining chamber voxel is $(1, 0, 0)$. Every lattice path from the chamber suffix $k \ge 2$ to the outside, avoiding the host, passes through this voxel. The tile is connected and too large to stay in the suffix, so it contains $(1, 0, 0)$ as well. Thus A fills all of K_i. Deleting g_i makes this a whole cut component of A. Apply Lemma 7.1 to obtain the exact neighbor $A = -40n_i + \Omega(T)$. The complete local theorem is `Rotations.Euclidean.euclidean_chamber_forces_neighbor` in `Rotations/Euclidean/LocalForcing.lean`. □

The measure-theoretic details do not require face-to-face contacts, local finiteness as an extra assumption, or an initial common grid. Countability, avoidance of voxel boundaries, and disjointness of one tile's interior from another tile's closed solid are proved in `Rotations/Euclidean/Covers.lean`.

8.3 Euclidean Rigidity and Hardness

Theorem 8.3. *Every Euclidean tiling by $\Omega(T)$ has a common Euclidean frame: for some motion r, its placed copies are exactly*

$$\{r(40z + \Omega(T)) : z \in \mathbb{Z}^3\}.$$

Thus all copies have the same complete orientation, even when reflections are allowed.

Proof. Normalize any one placed copy to be the identity. Repeatedly apply Lemma 8.2 along lattice paths to force every $40z + \Omega(T)$. These closed solids cover all of $\mathbb{R}^3$, since T is a fundamental domain for $40\mathbb{Z}^3$. No additional copy is possible: an interior point of that copy lies in one of the forced closed solids, contradicting interior disjointness unless it is that same placed copy. Undo the normalization. This is `Rotations.Euclidean.rigidity` in `Rotations/Euclidean/GlobalRigidity.lean`. □

Theorem 8.4. *For every finite nonempty $E \subset \mathbb{Z}^3$,*

$$E \text{ tiles } \mathbb{Z}^3 \text{ by translations} \iff \Omega(P(E)) \text{ tiles } \mathbb{R}^3 \text{ by proper Euclidean motions.}$$

The equivalence also holds when all Euclidean motions are allowed. Consequently tilability of one face-connected polycube by arbitrary proper Euclidean motions is co-RE-hard and undecidable; the same holds when reflections are allowed.

Proof. The forward direction uses only translations. For the reverse direction, split each placed copy of $\Omega(P(E))$ into its constituent copies of $\Omega(T)$. They form an almost-everywhere tiling, because their constituent voxel sets are disjoint and the only overlaps introduced by thickening are boundaries. Apply Theorem 8.3. After undoing its common frame, every parent copy is translated by a vector in $40\mathbb{Z}^3$. Sampling integer voxel centers gives an exact lattice tiling by $P(E)$; collapsing the $40\mathbb{Z}^3$ grid gives a translational tiling by E, as in Theorem 7.3. Connectedness, computability, and the exact size multiplier 64,000 are unchanged. The equivalence is `Rotations.Euclidean.assembly_allowed_iff` in `Rotations/Euclidean/Assembly.lean`. The reduction and both hardness results are in `Rotations/Euclidean/Main.lean`, including `Rotations.Euclidean.arbitrary_proper_rotation_coRE_hard` and `Rotations.Euclidean.arbitrary_orthogonal_coRE_hard`. □

8.4 Public Reduction and Size Statements

The public abstract uses one Euclidean point type for both translations and motions. Its `Motion` type is Mathlib's affine isometry equivalence, and `f '' Solid P` is the image of the solid body under the motion. `Abstract.tiling_reduction` in `Abstract.lean` and `Abstract.real_tiling_reduction` in `Abstract.lean` state total computable Wang reductions for translations. The corresponding statement for any set of Euclidean motions containing all translations is `Abstract.rigid_tiling_reduction` in `Abstract.lean`.

Each reduction theorem includes its cube bound for the same computable function. In dimension three, the bounds are $B(n)$ for translations and $64{,}000B(n)$ for arbitrary Euclidean motions, with $B(n)$ from Proposition 6.10. Here n counts input Wang tiles. For the same rotation-resistant output P, `Abstract.rigid_tiling_reduction_three` in `Abstract.lean` gives the latter bound and states the equivalences

$$\begin{aligned} \mathcal{T} \text{ tiles the plane} &\iff P \text{ tiles } \mathbb{Z}^3 \text{ by translations} \\ &\iff P \text{ tiles } \mathbb{R}^3 \text{ by translations} \\ &\iff P \text{ tiles } \mathbb{R}^3 \text{ by Euclidean motions,} \end{aligned}$$

where the last condition may use any set of Euclidean motions containing all translations. The six-key correctness theorem already permits any prescribed set of allowed orientations containing the identity.

Restricting this set to the identity gives the translation-only equivalence; the lattice/real translation theorem then connects the two translation predicates. Thus no new construction or additional cubes are needed.

The total reduction in `FiniteCompiler/Effective/PackedTotalCompiler.lean` uses the guarded construction with checked prime tables through 3^{21} Wang tiles. For larger inputs, it generates primes and ternary digits, then builds the structural tests and one cycle test per forbidden partial pattern or excluded seed letter. The numerical tests are checked in `FiniteCompiler/Effective/GuardedNumericalCorrect.lean` and `FiniteCompiler/Effective/PackedPatternCorrect.lean`.

`FiniteCompiler/Effective/GuardedAssembly.lean` stacks these tests using the reserved partition and assembles the connected tile within the volume bound.

`FiniteCompiler.Effective.packedCompile_computable` proves computability of this total function. Its size bound is `FiniteCompiler.Effective.packedCompile_real_bound`. The six-key replacement is a primitive-recursive map and multiplies its output length by exactly 64,000, so the public Euclidean reduction combines computability, correctness, and size as well.

9 Co-RE Membership for Euclidean Motions

Fix $d \geq 1$ and a finite nonempty $P \subset \mathbb{Z}^d$, and let $S = \bigcup_{p \in P}(p + [-1/2, 1/2]^d)$. Choose $p_0 \in P$; its image under a placed copy is that copy's ***anchor***. Each copy contains the open radius-$1/2$ ball about its anchor. Thus anchors in a packing are at least distance 1 apart. Partition space into half-open coordinate boxes of side $1/(2(d+1))$. Each box contains at most one anchor because its diameter is less than 1. These boxes give countably many ***slots*** indexed by $z \in \mathbb{Z}^d$.

A slot has a presence variable $e_z \in \{0, 1\}$ and, when occupied, an anchor $a_z = (2z + u_z)/(4(d+1))$ with $u_z \in [-1, 1]^d$ and an orthogonal matrix R_z. Its placement is $x \mapsto a_z + R_z(x - p_0)$; all matrix entries lie in $[-1, 1]$. The allowed linear parts have closed polynomial descriptions: $R = I$ for translations, $R^{\mathsf{T}}R = I$ and $\det R = 1$ for proper motions, and $R^{\mathsf{T}}R = I$ for all Euclidean motions. For reflections without rotations, use $R = I$ or $R = I - 2vv^{\mathsf{T}}$ with $\sum_i v_i^2 = 1$. The unit normal $v \in [-1, 1]^d$ permits every mirror hyperplane. Impose these conditions only on occupied slots.

For every $y \in \mathbb{Q}^d$, require coverage by a closed voxel of an occupied slot. Only finitely many slots can contribute: their anchors lie within a computable diameter bound of S from y. For every pair of distinct occupied slots and every pair of their voxels, also forbid a rational point from lying in both open voxels. Inverse placement gives the tests

$$\left|\left(p_0 + R_z^{\mathsf{T}}(y - a_z)\right)_i - p_i\right| \leq \tfrac{1}{2}$$

for closed membership and strict inequalities for open membership. The negation of simultaneous open membership is a finite disjunction of weak polynomial inequalities. All requirements are therefore computably enumerable finite closed polynomial formulas in variables from $[-1, 1]$. Lean uses integer coefficients and additional bounded variables constrained to equal reciprocals of positive integers.

These constraints characterize tilings. An actual tiling supplies its slot data. Conversely, open overlaps would contain rational points, contradicting packing. The anchors are thus separated, so only finitely many copies meet any bounded set. Their union is closed and contains the dense set $\mathbb{Q}^d$, hence equals $\mathbb{R}^d$. The countably many transformed voxel boundaries have measure zero, giving almost-everywhere unique coverage. Closed slot boxes can share boundary points; packing still forces occupied anchors to be separated.

At stage k, take the first $k+1$ constraints and relax each atomic inequality $f \leq 0$ to $f \leq 1/(k+1)$. Only finitely many variables occur. For $f = \sum_m c_m m$, the computable Lipschitz bound $\sum_m |c_m| \deg(m)$ on the unit box, in the maximum norm, gives a rational grid fine enough that rounding preserves every exact solution with this tolerance. Enumerate the finite grid and halt if no point passes. Clear denominators to

perform the tests with integers. If every stage passes, compactness of $[-1, 1]^{\mathbb{N}}$ and the vanishing tolerances give an exact solution to every constraint. The procedure therefore halts exactly when no tiling exists.

The proof files are:

- `EuclideanMembership/EffectiveCompactness.lean`: effective compactness.
- `EuclideanMembership/SourceModel.lean`: the geometric equivalence.
- `EuclideanMembership/EffectiveSystem.lean`: co-RE membership.

The last file proves membership for all four choices of the rotation and reflection flags, both with and without requiring connectedness. Combined with hardness, all four connected tiling problems are co-RE-complete for every fixed $d \geq 3$.

References

[Ber66] Robert Berger. *The Undecidability of the Domino Problem*. Number 66 in Memoirs of the American Mathematical Society. American Mathematical Society, 1966.

[Bha20] Siddhartha Bhattacharya. Periodicity and decidability of tilings of $\mathbb{Z}^2$. *American Journal of Mathematics*, 142(1):255–266, 2020. arXiv:1602.05738v1.

[BN91] Danièle Beauquier and Maurice Nivat. On translating one polyomino to tile the plane. *Discrete & Computational Geometry*, 6:575–592, 1991.

[dDPGGM26] Jaume de Dios Pont, Jan Grebík, Rachel Greenfeld, and José Madrid. Periodicity and decidability of translational tilings by rational polygonal sets. *Expositiones Mathematicae*, 44(1):125620, 2026. Numbered references use arXiv:2408.02151v2.

[DL26] Erik D. Demaine and Stefan Langerman. translation-tiling-paper: Tiling 3D space by (translates of) a single polycube is undecidable. GitHub repository, 2026. https://github.com/edemaine/translation-tiling-paper.

[GT21] Rachel Greenfeld and Terence Tao. The structure of translational tilings in $\mathbb{Z}^d$. *Discrete Analysis*, (16):1–28, 2021. arXiv:2010.03254v3.

[GT23] Rachel Greenfeld and Terence Tao. Undecidable translational tilings with only two tiles, or one nonabelian tile. *Discrete & Computational Geometry*, 70(4):1652–1706, 2023. Numbered references use arXiv:2108.07902v3.

[GT24] Rachel Greenfeld and Terence Tao. A counterexample to the periodic tiling conjecture. *Annals of Mathematics*, 200(1):301–363, 2024. Numbered references use arXiv:2211.15847v3.

[GT25] Rachel Greenfeld and Terence Tao. Undecidability of translational monotilings. *Journal of the European Mathematical Society*, 2025. Online first. Numbered references use arXiv:2309.09504v2.

[Kim26] Yoonhu Kim. Undecidability of translational tiling with 2 polycubes, 2026. Version 2, August 10, 2026. Numbered references use arXiv:2508.11725v2.

[MSS25] Tom Meyerovitch, Shrey Sanadhya, and Yaar Solomon. A note on reduction of tiling problems. *Israel Journal of Mathematics*, 267:421–435, 2025. Numbered references use arXiv:2211.07140v1.

[Ope26] OpenAI. A translational tile with no fully periodic tiling in dimension three. Preprint, September 23, 2026. Versioned preprint on GitHub, accessed October 7, 2026.