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

Erik D. Demaine* Stefan Langerman†

Abstract

We prove co-RE-completeness, and thus undecidability, of the following problem: given a single (connected) polycube, decide whether it tiles 3D Euclidean space by translations. The same result holds if we also allow rotations and/or reflections. We reduce from Wang tiling using the decorated two-prime Sudoku construction of Greenfeld and Tao and a cyclic encoding adapted from OpenAI's 3D aperiodic tile, and apply a reduction of Kim to make the prototile connected (via faces). Dimension three is optimal: translational monotiling is known to be decidable in $\mathbb{Z}^2$ and for a single (possibly disconnected) polyomino in $\mathbb{R}^2$.

This paper gives a human-readable overview of the results, proof ideas, and formal theorem statements. The proof is verified in Lean. See Section 4 for details of our approach, and the repository [DL26b] for the complete Lean code and a longer LLM-written technical article.

1 Introduction

Given k finite subsets of $\mathbb{Z}^d$ (***prototiles***), can (infinitely many) copies of those subsets be *translated* to cover all of $\mathbb{Z}^d$ without overlaps? (Refer to Figure 1.) For which values of k and d is this problem decidable? In this paper, we prove undecidability for $k = 1$ and $d = 3$.

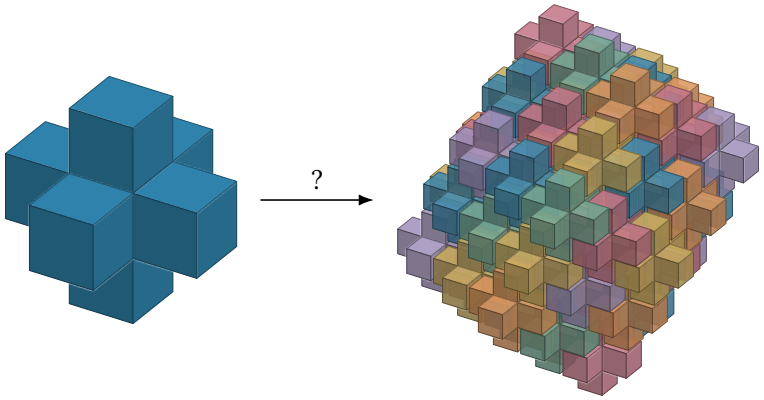


Figure 1: The tiling-by-translation decision problem: given k shapes, can translated copies tile dD space? Here, $k = 1$ and $d = 3$, the main version solved in this paper.

*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.

Translations only				
Prototiles	1D	2D	3D	dD ($d \geq 4$)
1	R (↓)	R [BN91, Bha20, GT21]	co-RE-c (Thm. 1.1)*	co-RE-c (←)*
2	R (↓)	OPEN	co-RE-c [Kim26] (↑)*	co-RE-c (↑)*
3	R (↓)	co-RE-c [MIT25]	co-RE-c [YZ25a, MIT25] (↑)*	co-RE-c (↑)*
4	R (↓)	co-RE-c [YZ25c] (↑)	co-RE-c (↑)*	co-RE-c (↑)*
5	R (↓)	co-RE-c [Kim25]*	co-RE-c (↑)*	co-RE-c (↑)*
6	R (↓)	co-RE-c (↑)*	co-RE-c [YZ25b] (↑)*	co-RE-c (↑)*
7	R (↓)	co-RE-c [YZ26a, YZ25d] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
8	R (↓)	co-RE-c [YZ26b] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
9	R (↓)	co-RE-c [Yan25] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
10	R (↓)	co-RE-c [Yan23] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
11	R (↓)	co-RE-c [Oll09] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
n	R [GT25]	co-RE-c [Ber66, Gol70] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*

Table 1: Complexity of tiling the whole integer lattice by translations with possibly disconnected polycubes, organized as in [MIT25, Table 3]. Rows bound the number of prototiles; n means an arbitrary finite input set. The last column applies to every fixed $d \geq 4$. "R" (green) denotes decidability, "co-RE-c" (red) denotes co-RE-completeness, and yellow marks open cases. Blue highlights our new one-tile result (Theorem 1.1) and its extension to higher dimensions by extrusion. Results marked "*" work even when every prototile is connected. (Unmarked results allow disconnected tiles.) Arrows point to adjacent entries that imply the stated result, including the connectedness restriction when marked.

The decidability question has a long history, summarized in Table 1. The first result is Golomb's implementation [Gol70] of the undecidable Wang tiling problem [Ber66], which works in 2D but requires arbitrarily many prototiles. In 2D, Ollinger [Oll09] reduced the number of prototiles to a constant (11), which was subsequently improved in a series of papers; the current record is 3 by the MIT–ULB CompGeom Group [MIT25]. In 3D, the first undecidability result used 6 prototiles [YZ25b], and the current record is 2 [Kim26]; we improve to the minimum possible number of 3D prototiles, 1. 3D is also the minimum dimension for undecidability with 1 prototile, given known algorithms for 2D [BN91, Bha20, GT21].

Theorem 1.1. *For every fixed $d \geq 3$, tiling $\mathbb{Z}^d$ by translations of a single finite connected[1] subset of $\mathbb{Z}^d$ is co-RE-complete,[2] and hence undecidable.[3] Tiling $\mathbb{R}^d$ with a single finite connected[1] polycube, using any set of Euclidean motions containing all translations, is co-RE-hard,[4] and hence undecidable.[5] With translations and optionally rotations and/or reflections, the problem is co-RE-complete.[6]*

Co-RE is the complexity class of decision problems whose "no" instances can be recognized in finite time by an algorithm (Turing machine). A problem is ***co-RE-hard*** if every co-RE problem can be reduced to it via

[1] Connectivity of a set in $\mathbb{Z}^d$ is defined according to lattice adjacency in $\mathbb{Z}^d$. Connectivity of a polycube in $\mathbb{R}^d$ is thus defined according to facet adjacency of the constituent unit cubes. See Section 3.2.

[2] ∃∀ theorem `tiling_coRE_complete`. See Section 3.4.

[3] ∃∀ theorem `tiling_undecidable`. See Section 3.4.

[4] ∃∀ theorem `real_tiling_coRE_hard`. See Section 3.6.

[5] ∃∀ theorem `real_tiling_undecidable`. See Section 3.6.

[6] ∃∀ theorem `real_tiling_coRE_complete`. See Section 3.6.

Translations + rotations (+ reflections)				
Prototiles	1D	2D	3D	dD ($d \geq 4$)
1	R (↓)	OPEN	co-RE-c (Thm. 1.1)*	co-RE-c (Thm. 1.1)*
2	R (↓)	co-RE-c [MIT25]	co-RE-c [MIT25] (↑)*	co-RE-c (↑)*
3	R (↓)	co-RE-c [Sta25, Kim25]*	co-RE-c (↑)*	co-RE-c (↑)*
4	R (↓)	co-RE-c (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
5	R (↓)	co-RE-c [Oll09] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*
n	R [GT23]	co-RE-c [Ber66, Gol70] (↑)*	co-RE-c (↑)*	co-RE-c (↑)*

Table 2: Complexity of tiling Euclidean space by translations and rotations, with or without reflections, organized as in [MIT25, Table 2]. Notation follows Table 1.

a computable many-one reduction, and ***co-RE-complete*** if it is also in co-RE. It is known that some co-RE-complete problems (such as "does a given algorithm run forever?") are undecidable, so co-RE-completeness implies undecidability.

Our results also improve the state-of-the-art for tiling with translations and rotations (and optionally reflections), as summarized in Table 2. Previously, the best undecidability result for 3D required two prototiles [MIT25]. The 2D case with one prototile remains open.

2 Proof Overview

We prove co-RE-hardness by reducing from the problem of Wang tiling, introduced by [Wan61] and proved co-RE-hard by [Ber66]. A ***Wang tile*** is a unit square with a color on each of its four edges. Given a set W of Wang tiles, a ***Wang tiling*** assigns a tile from W to each grid point in $\mathbb{Z}^2$ such that, when placing those tiles by translation, touching edges have the same color. Deciding whether a given finite set of Wang tiles admits such a tiling is co-RE-complete [Ber66].

Thus our main task for $\mathbb{Z}^3$ is to construct a computable map from a finite Wang tile set $\mathcal{T}$ to a finite connected set $F_{\mathcal{T}} \subset \mathbb{Z}^3$ such that $\mathcal{T}$ tiles $\mathbb{Z}^2$ if and only if $F_{\mathcal{T}}$ tiles $\mathbb{Z}^3$ by translations. To achieve this, we combine five constructions, as summarized in Figure 2:

1. Greenfeld and Tao [GT25, Section 5] reduce Wang tiling to finding a ***Sudoku array*** — a grid with infinitely many rows and a fixed number of columns (independent of the Wang tile set) such that

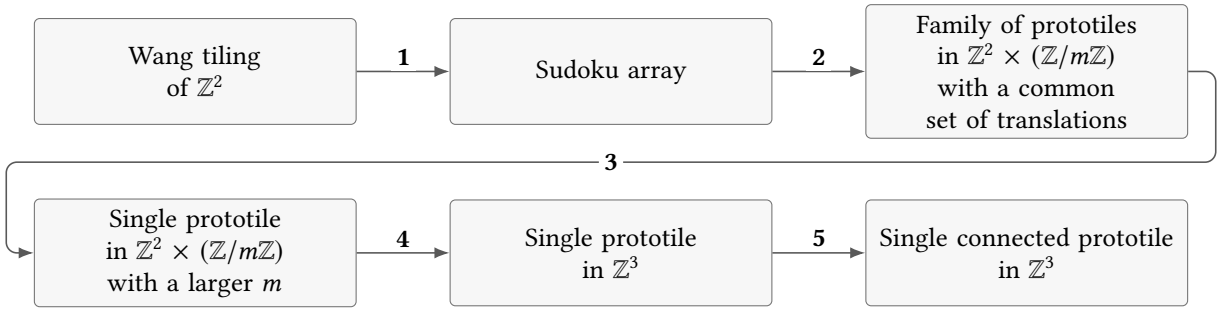


Figure 2: The five reductions in our construction for tiling $\mathbb{Z}^3$ by translations. Arrow numbers match the steps in the text; every reduction preserves existence of a solution in both directions.

every integer-slope line reads as one of a specified finite set of words. Specifically, each grid symbol specifies a Wang tile together with two integers; the Sudoku rule organizes these integers using powers of two fixed primes $p < q$, whose exponents encode the two coordinates of the simulated plane. The line rules compare neighboring positions in this simulated plane to enforce matching edge colors. We additionally require that neither integer component is constant down any column.

2. We adapt OpenAI's cyclic tiling encoding [Ope26, Sections 3–5] to the Sudoku array problem from the previous step. (OpenAI originally applied this encoding to a different construction of Greenfeld and Tao's [GT25, Section 4], which uses a one-prime Sudoku rule.) This adaptation is our main new work: it enforces the Wang tile constraints and both nonconstant-column conditions simultaneously. The set of translations encodes the Sudoku array, and the prototiles test the rules: forbidden combinations of symbols cause overlaps or gaps. The result is a finite family of prototiles (depending on the Wang tile set) that each tile a target space using the same infinite set of translations, if and only if the required Sudoku array exists. The target space being tiled is $\mathbb{Z}^2 \times (\mathbb{Z}/m\mathbb{Z})$, a 3D integer grid whose z coordinate wraps around modulo m (where m depends on the Wang tile set).

3. We apply OpenAI's stacking construction [Ope26, Section 6] (based on an earlier construction of Greenfeld and Tao [GT24, Theorem 3.1 and Lemma 3.2]) to combine the prototiles from the previous step into a single prototile that tiles a (larger) target space if and only if such a common set of translations exists. This construction enlarges m in the target space $\mathbb{Z}^2 \times (\mathbb{Z}/m\mathbb{Z})$.

4. We apply the reduction of Meyerovitch, Sanadhya, and Solomon [MSS25, Theorem 1.1], which converts tiling problems in a quotient of $\mathbb{Z}^3$ into tiling problems in $\mathbb{Z}^3$, while preserving the number of prototiles.[7] Our target space $\mathbb{Z}^2 \times (\mathbb{Z}/m\mathbb{Z})$ is such a ***quotient***: it identifies each point (x, y, z) with $(x, y, z + m)$. Their construction uses matching bumps and dents to force alignment to a scaled grid and to force repetition every m grid steps in the z direction. Thus we convert the prototile from the previous step into a single finite 3D prototile that tiles $\mathbb{Z}^3$ if and only if the original prototile tiles the quotient space $\mathbb{Z}^2 \times (\mathbb{Z}/m\mathbb{Z})$.

5. We use a reduction of Kim [Kim26, Theorem 2.5] which makes the prototile connected while preserving dimension, the number of prototiles, and tileability. This construction scales up the prototile, giving space to connect the lattice points by thin corridors, with bumps and dents to force alignment.

For $d \geq 3$, appending $d - 3$ zero coordinates to each lattice point preserves tilability, by slicing $\mathbb{Z}^d$ into parallel copies of $\mathbb{Z}^3$. Replacing each lattice point p by the unit cube $p + [0, 1]^d$ then gives a polycube in $\mathbb{R}^d$ with the same translational tilability; the reverse implication rounds real translations to a generically shifted lattice. To preserve tilability under arbitrary Euclidean motions, we replace each unit cube by a constant-size polycube with distinct matching keys and recesses on its opposite facets, which force a common orientation and lattice in every Euclidean tiling. Co-RE membership, also for finitely many prototiles, follows by encoding tilings as a computable compact system of closed polynomial constraints and using finite-precision tests to recognize infeasibility.

For n Wang tiles, the translation-only reduction uses at most

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

lattice points / unit cubes in every fixed dimension $d \geq 3$, where $\lg = \log_2$. A logarithmic number of encoding channels and sparse tests give the quasipolynomial dependence, and further optimizations improve

[7] The proof of [MSS25, Lemma 2.1] has a gap in its local forcing argument: covering a neighboring box center without overlap does not force the claimed placement. Our Lean formalization gives a counterexample to this local claim and proves that their reduction is nonetheless correct in our setting.

the constants in the bound (see "Quantitative Size Bounds" in the technical overview). For $n = 1$, the same construction uses at most 2^{2112} unit cubes. Allowing rotations and/or reflections gives bounds of $64{,}000B(n)$ in 3D and $2^{10(d+1)^2}B(n)$ in dD.

3 Formal Verification in Lean

The proof of Theorem 1.1 is long and technical. To make it checkable, we formalized it in the Lean 4 proof assistant, building on Mathlib and on the formalization of Wang tiling undecidability (`lean-wang` [Dem26]). The complete Lean proof is available in the repository. The key point is that a reader does not need to read the proof, or even most of the definitions, to trust the result: it suffices to check that the handful of definitions and theorem statements in a single file, `Abstract.lean`, accurately express the claim. The Lean proof assistant then verifies the proofs of these statements, and that they rely on only Lean's three standard axioms (`propext`, `Classical.choice`, `Quot.sound`).

In the rest of this section, we explain the statements from `Abstract.lean`. (The source file `lean/TranslationTiling/Abstract.lean` remains the authoritative version.) It uses Mathlib's definitions of computability, Euclidean space, affine isometries, and Lebesgue measure and `lean-wang`'s definitions of co-RE (`CoREPred`, `CoREHard`, and `CoREComplete`, defined as in the introduction) and Wang tiling (`TileSet` and `TilesPlane`), together with a short file `AbstractBridge.lean` whose job is to connect these statements to the definitions used in the rest of the formalization (but which need not be checked by a human).

```
import Mathlib.Computability.Halting
import Mathlib.MeasureTheory.Measure.Lebesgue.Basic
import Mathlib.Analysis.Normed.Affine.Isometry
import Mathlib.Analysis.InnerProductSpace.PiL2
import Mathlib.Geometry.Euclidean.Projection
import Mathlib.LinearAlgebra.Determinant
import LeanWang.CoRE
import LeanWang.Basic
import TranslationTiling.AbstractBridge
```

3.1 Prototiles

Fix a dimension $d \geq 3$. A ***prototile*** is a finite list P of points in $\mathbb{Z}^d$. It is a list (not a set) so that it can be given as input to an algorithm; repetitions are immaterial.

```
abbrev Point (d : ℕ) := Fin d → ℤ

abbrev Prototile (d : ℕ) := List (Point d)
```

3.2 Connectivity

Two points $a, b \in \mathbb{Z}^d$ are ***adjacent*** if $\|a - b\|_1 = 1$. A prototile P is ***connected*** if any two points in P are joined by a path of adjacent points inside P, i.e., they are related by the reflexive transitive closure of the relation "both points are in P and adjacent".

```
def Adjacent {d : ℕ} (a b : Point d) : Prop :=
  ∑ i, |a i - b i| = 1

def Connected {d : ℕ} (P : Prototile d) : Prop :=
  ∀ p ∈ P, ∀ q ∈ P, Relation.ReflTransGen (fun a b => a ∈ P ∧ b ∈ P ∧ Adjacent a b) p q
```

3.3 Tiling $\mathbb{Z}^d$

P ***tiles*** $\mathbb{Z}^d$ if there exists a set $T \subseteq \mathbb{Z}^d$ of translations such that every $x \in \mathbb{Z}^d$ lies in exactly one translate $t + P$ with $t \in T$, i.e., there is exactly one $t \in T$ with $x - t \in P$. (No periodicity or finiteness assumption is placed on T.)

```
def Tiles {d : ℕ} (P : Prototile d) : Prop :=
  ∃ T : Set (Point d), ∀ x : Point d, ∃! t ∈ T, x - t ∈ P
```

3.4 Main Theorem for $\mathbb{Z}^d$

Deciding whether a connected prototile tiles $\mathbb{Z}^d$ is co-RE-complete (`LeanWang.CoREComplete`, from `lean-wang`): the non-tilers are recursively enumerable, and every co-RE problem reduces to it by a computable many-one reduction. Consequently (and using only Mathlib's computable predicate) no algorithm decides it. In both statements, the proof is a single call to a theorem in `AbstractBridge.lean`.

```
theorem tiling_coRE_complete (d : ℕ) (hd : 3 ≤ d) :
  LeanWang.CoREComplete fun P : Prototile d => Connected P ∧ Tiles P := ...

theorem tiling_undecidable (d : ℕ) (hd : 3 ≤ d) :
  ¬ ComputablePred fun P : Prototile d => Connected P ∧ Tiles P := ...
```

A computable reduction converts a list of n Wang tiles into an equivalent nonempty connected prototile $P \subset \mathbb{Z}^d$ using at most $B(n) = \max\{(n+4)^{8.83302\left(\lg\lg(n+4)+\lg\lg\lg(n+4)\right)+570}, 2^{2392}\}$ unit cubes: the Wang tiles tile the plane if and only if P tiles $\mathbb{Z}^d$ by translations. Here `LeanWang.TileSet` is a list of Wang tiles, and `LeanWang.TilesPlane` asserts that they tile the whole plane. List lengths count the input tiles and output cubes, and `Real.logb 2` denotes the base-2 logarithm lg.

```
theorem tiling_reduction (d : ℕ) (hd : 3 ≤ d) :
  ∃ f : LeanWang.TileSet → Prototile d, Computable f ∧
    ∀ T, f T ≠ [] ∧ Connected (f T) ∧ (LeanWang.TilesPlane T ↔ Tiles (f T)) ∧
      ((f T).length : ℝ) ≤ max (((T.length : ℝ) + 4) ^
        ((441651 / 50000 : ℝ) * (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)) +
          Real.logb 2 (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)))) + 570))
        ((2 : ℝ)^2392) := ...
```

3.5 Tiling $\mathbb{R}^d$

We represent a ***point*** `x` of $\mathbb{R}^d$ using Mathlib's `EuclideanSpace`, which provides the usual Euclidean norm. A coordinate is written `x i` for `i : Fin d`, just like our `Point` definition for $\mathbb{Z}^d$. The ***solid body*** of P is the union of the closed unit cubes $p + [0,1]^d$ for $p \in P$.

```
abbrev RealPoint (d : ℕ) := EuclideanSpace ℝ (Fin d)

def Solid {d : ℕ} (P : Prototile d) : Set (RealPoint d) :=
  {x | ∃ p ∈ P, ∀ i, p i ≤ x i ∧ x i ≤ p i + 1}
```

A ***Euclidean motion*** is an affine bijection preserving distances. Mathlib's `AffineIsometryEquiv` represents such a motion. In the notation below, $\simeq$ denotes a bijection, the superscripts mean affine and isometric, and $[\mathbb{R}]$ specifies real scalars.

```
abbrev Motion (d : ℕ) := RealPoint d ≃ᵃⁱ[ℝ] RealPoint d
```

We define four sets of motions: all Euclidean motions; translations alone; translations and rotations; and translations and at most one arbitrary hyperplane reflection. A motion is a translation exactly when its linear part is the identity, and is a translation composed with a rotation exactly when its linear part has determinant 1. In Lean, `f.linear` extracts the linear part, `1` denotes the identity, and `.det` computes the determinant. Mathlib's `AffineSubspace.perpBisector p q` is the set of points equidistant from `p` and `q`, a hyperplane when they are distinct. The function `EuclideanGeometry.reflection` reflects in this affine subspace. The product applies the reflection first, then the translation. When `p = q`, the bisector is the whole space and its reflection is the identity, so translations alone are included too.

```
def allMotions {d : ℕ} : Set (Motion d) := Set.univ

def translations {d : ℕ} : Set (Motion d) := {f | f.linear = 1}

def translationsAndRotations {d : ℕ} : Set (Motion d) := {f | f.linear.det = 1}

def translationsAndReflections {d : ℕ} : Set (Motion d) :=
  {f | ∃ t ∈ translations, ∃ p q : RealPoint d,
    f = t * EuclideanGeometry.reflection (AffineSubspace.perpBisector p q)}
```

The predicate `RealTiles P motions` requires a set of permitted motions whose placed copies cover Lebesgue-almost every point exactly once. The notation `f '' S` means the image $\{f(x) : x \in S\}$. ("Almost every" so that cubes may share boundaries.)

```
def RealTiles {d : ℕ} (P : Prototile d) (motions : Set (Motion d)) : Prop :=
  ∃ T ⊆ motions, ∀ᵐ x ∂MeasureTheory.volume,
    ∃! f ∈ T, x ∈ f '' Solid P
```

3.6 Main Theorem for $\mathbb{R}^d$

Deciding whether a connected polycube tiles $\mathbb{R}^d$ is co-RE-hard (`LeanWang.CoREHard`, from `lean-wang`) and undecidable for any set of Euclidean motions containing all translations.

```
theorem real_tiling_coRE_hard (d : ℕ) (hd : 3 ≤ d)
  (motions : Set (Motion d)) (hm : translations ⊆ motions) :
  LeanWang.CoREHard fun P : Prototile d => Connected P ∧ RealTiles P motions := ...

theorem real_tiling_undecidable (d : ℕ) (hd : 3 ≤ d)
  (motions : Set (Motion d)) (hm : translations ⊆ motions) :
  ¬ ComputablePred fun P : Prototile d => Connected P ∧ RealTiles P motions := ...
```

For each of the four motion sets defined above, tiling $\mathbb{R}^d$ with a single finite connected polycube is co-RE-complete.

```
theorem real_tiling_coRE_complete (d : ℕ) (hd : 3 ≤ d)
  (motions : Set (Motion d))
  (hm : motions ∈ ({allMotions, translations, translationsAndRotations,
                    translationsAndReflections} : Set _)) :
  LeanWang.CoREComplete fun P : Prototile d => Connected P ∧ RealTiles P motions := ...
```

Together, these Lean theorems establish Theorem 1.1.

The same computable reduction and cube bound from $\mathbb{Z}^d$ apply. The following are equivalent: the input Wang tiles tile the plane; the output prototile tiles $\mathbb{Z}^d$ by lattice translations; and its solid body tiles $\mathbb{R}^d$ by arbitrary real translations.

```
theorem real_tiling_reduction (d : ℕ) (hd : 3 ≤ d) :
  ∃ f : LeanWang.TileSet → Prototile d, Computable f ∧
    ∀ T, f T ≠ [] ∧ Connected (f T) ∧
      (LeanWang.TilesPlane T ↔ Tiles (f T)) ∧
      (Tiles (f T) ↔ RealTiles (f T) translations) ∧
      ((f T).length : ℝ) ≤ max (((T.length : ℝ) + 4) ^
        ((441651 / 50000 : ℝ) * (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)) +
          Real.logb 2 (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)))) + 570))
        ((2 : ℝ)^2392) := ...
```

A computable reduction produces a nonempty connected polycube that tiles $\mathbb{Z}^d$ by lattice translations if and only if the input Wang tiles tile the plane. For any set of Euclidean motions containing all translations, this is also equivalent to its solid body tiling $\mathbb{R}^d$ using those motions. Its cube bound is $2^{10(d+1)^2}B(n)$.

```
theorem rigid_tiling_reduction (d : ℕ) (hd : 3 ≤ d) :
  ∃ f : LeanWang.TileSet → Prototile d, Computable f ∧
    ∀ T, f T ≠ [] ∧ Connected (f T) ∧
      (LeanWang.TilesPlane T ↔ Tiles (f T)) ∧
      (∀ motions : Set (Motion d), translations ⊆ motions →
        (Tiles (f T) ↔ RealTiles (f T) motions)) ∧
      ((f T).length : ℝ) ≤ (2 : ℝ) ^ (10 * (d + 1) ^ 2) * (max (((T.length : ℝ) + 4) ^
        ((441651 / 50000 : ℝ) * (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)) +
          Real.logb 2 (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)))) + 570))
        ((2 : ℝ)^2392)) := ...
```

In 3D, the bound improves to $64{,}000B(n)$:

```
theorem rigid_tiling_reduction_three :
  ∃ f : LeanWang.TileSet → Prototile 3, Computable f ∧
    ∀ T, f T ≠ [] ∧ Connected (f T) ∧
      (LeanWang.TilesPlane T ↔ Tiles (f T)) ∧
      (∀ motions : Set (Motion 3), translations ⊆ motions →
        (Tiles (f T) ↔ RealTiles (f T) motions)) ∧
      ((f T).length : ℝ) ≤ 64000 * (max (((T.length : ℝ) + 4) ^
        ((441651 / 50000 : ℝ) * (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)) +
          Real.logb 2 (Real.logb 2 (Real.logb 2 ((T.length : ℝ) + 4)))) + 570))
        ((2 : ℝ)^2392)) := ...
```

4 Our Approach to the Proof

We made heavy use of AI to find, write, and formalize the proof, and we do not claim that this paper is a complete human-readable account of it. Instead, this paper contains only the part intended for humans: an explanation of the problem (Section 1), the main intuition behind the proof (Section 2), and the precise formalization of the statement (Section 3). The rigor of the proof comes from the Lean formalization rather than from the text.

A much longer LLM-generated technical article, which is not meant for human consumption, gives the full details of the construction and its proofs. It and the complete Lean code are available in the repository [DL26b] (files `slop.texlish` and `lean/`, respectively). We intend to expand this paper with further human-readable explanations of the proof.

Acknowledgments

We thank the MIT CompGeom Group for helpful discussions and for providing an inspiring atmosphere. This paper was typeset with Texlish [DL26a]. We used the following LLM models: GPT 6.1 Sol, GPT 6 Astra, Claude Sonnet 5.5, Claude Opus 5.5, and Claude Fable 5.1.

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.

[Dem26] Erik D. Demaine. lean-wang: Lean 4 formalization of Wang tiling and the domino problem. GitHub repository, 2026. https://github.com/edemaine/lean-wang.

[DL26a] Erik D. Demaine and Stefan Langerman. Texlish: Better LaTeX. Software, version 0.2.0, 2026. https://texlish.org/.

[DL26b] 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.

[Gol70] Solomon W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9(1):60–71, 1970.

[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.

[Kim25] Yoonhu Kim. Undecidability of tiling the plane with a set of 5 polyominoes, 2025. arXiv:2508.10067v1.

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

[MIT25] MIT–ULB CompGeom Group, Zachary Abel, Hugo Akitaya, Lily Chung, Erik D. Demaine, Jenny Diomidova, Della Hendrickson, Stefan Langerman, and Jayson Lynch. Undecidability of tiling with a tromino, 2025. arXiv:2509.07906v1.

[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.

[Oll09] Nicolas Ollinger. Tiling the plane with a fixed number of polyominoes. In *Language and Automata Theory and Applications (LATA 2009)*, volume 5457 of *Lecture Notes in Computer Science*, pages 638–647. Springer, 2009.

[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.

[Sta25] Jack Stade. Two tiling is undecidable, 2025. arXiv:2506.11628v1.

[Wan61] Hao Wang. Proving theorems by pattern recognition—II. *Bell System Technical Journal*, 40(1):1–41, 1961. Bell Labs archive.

[Yan23] Chao Yang. Tiling the plane with a set of ten polyominoes. *International Journal of Computational Geometry & Applications*, 33(03n04):55–64, 2023.

[Yan25] Chao Yang. On the undecidability of tiling the plane with a set of nine polyominoes (in Chinese). *SCIENCE CHINA Mathematics (Chinese Edition) [SCIENTIA SINICA Mathematica]*, 55(6):1113–1122, 2025.

[YZ25a] Chao Yang and Zhujun Zhang. On the undecidability of tiling the 3-dimensional space with a set of 3 polycubes, 2025. In Chinese. arXiv:2508.00192v1.

[YZ25b] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the 3-dimensional space with a set of 6 polycubes. *Proceedings of the American Mathematical Society*, 153(8):3541–3554,

2025. arXiv:2408.02196.

[YZ25c] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the plane with four tiles. arXiv:2506.19295, 2025. https://arxiv.org/abs/2506.19295.

[YZ25d] Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the plane with orthogonally convex polyominoes. arXiv:2506.12726, 2025. https://arxiv.org/abs/2506.12726.

[YZ26a] Chao Yang and Zhujun Zhang. Translational aperiodic sets of 7 polyominoes. *Science China Mathematics*, 2026. To appear. Originally arXiv:2412.17382.

[YZ26b] Chao Yang and Zhujun Zhang. Translational tiling with 8 polyominoes is undecidable. *Discrete & Computational Geometry*, 75(4):1048–1059, 2026. Originally arXiv:2403.13472.