Tetris with Few Piece Types

MIT Hardness Group[*]   Erik D. Demaine[†]   Holden Hall[†]   Jeffery Li[†]

Abstract

We prove NP-hardness and #P-hardness of Tetris clearing (clearing an initial board using a given sequence of pieces) with the Super Rotation System (SRS), even when the pieces are limited to *any two* of the seven Tetris piece types. This result is the first advance on a question posed twenty years ago: which piece sets are easy vs. hard? All previous Tetris NP-hardness proofs used five of the seven piece types. We also prove ASP-completeness of Tetris clearing, using three piece types, as well as versions of 3-Partition and Numerical 3-Dimensional Matching where all input integers are distinct. Finally, we prove NP-hardness of Tetris survival and clearing under the "hard drops only" and "20G" modes, using two piece types, improving on a previous "hard drops only" result that used five piece types.

1   Introduction

Tetris is one of the oldest and most popular puzzle video games, originally created by Alexey Pajitnov in 1984. Tetris has reached mainstream media many times, most recently in the biopic *Tetris* [Tet23] and with the news of 13-year-old Willis Gibson being the first person to "beat" the NES version of Tetris by reaching a killscreen [Deb24].

The rules of Tetris are simple. In each round, a tetromino piece (one of ▭, ▦, ▼, ▟, ▜, ▙, ▞) spawns at the top of a grid and periodically moves down one unit, assuming the squares below the piece are empty. The player can repeatedly move this piece one unit left, one unit right, or one unit down, or rotate the piece by $\pm 90°$. When any part of the piece rests on top of a filled square for long enough that it triggers an automatic downward move, the piece "locks" in place, and stops moving. If a piece stops above a certain height or where the next piece would spawn, the player loses; otherwise, the next piece spawns at the top of the grid, and play continues. Completely filling a row causes the row to clear, and all squares above that row move downward by one unit. For more detailed rules, see [Wik].

To study Tetris from a computational complexity perspective, we generally assume that the player is given a sequence of pieces and an initial board state of filled cells, making the game perfect information (as introduced in [BDH+04]). The two main objectives we consider here are "clearing" and "survival" (as introduced in [DDE+17]). In **Tetris clearing**, we want to determine whether we can clear the entire board after placing all the given pieces. In **Tetris survival**, we want to determine whether the player can avoid losing before placing all the given pieces. Previous work shows that these problems are NP-complete, even to approximate various metrics within $n^{1-\varepsilon}$ [BDH+04], or with only 8 columns or 4 rows [ACD+20], or with additional constraints on drops [Tem15], or with k-ominoes for $k \geq 3$ clearing or $k \geq 4$ survival [DDE+17].

	I	O	T	S	Z	J	L
I	–	Prop. 4.1 (H)	Prop. 4.1 (H)	Prop. 4.1	Prop. 4.1	Prop. 4.1 (H, G)	Prop. 4.1 (H, G)
O	–	–	Prop. 4.4	Prop. 4.3	Prop. 4.3	Prop. 4.2	Prop. 4.2
T	–	–	–	Prop. 4.5	Prop. 4.5	Prop. 4.7	Prop. 4.7
S	–	–	–	–	Prop. 4.6	Prop. 4.8	Prop. 4.7
Z	–	–	–	–	–	Prop. 4.7	Prop. 4.8
J	–	–	–	–	–	–	Prop. 4.9 (G?)
L	–	–	–	–	–	–	–

Table 1: Our NP-hardness results for Tetris clearing assuming SRS. Each entry in a specific row and column corresponds to the proposition for the hardness of the two-element subset consisting of the row piece and column piece (for example, the entry "Prop. 4.1" in row I and column O indicates that Proposition 4.1 proves hardness for the subset { I, O }). Letters in parentheses denote additional models ("H" for "hard drop only", "G" for "20G"); question mark indicates a conjecture for hardness under that additional model.

1.1 Our Results

One of the open problems posed in the original paper proving Tetris NP-hard twenty years ago [BDH+04] is to determine which subsets of the seven Tetris piece types { I, O, T, S, Z, J, L } suffice for NP-hardness, and which admit a polynomial-time algorithm. All existing Tetris NP-hardness proofs [BDH+04, ACD+20, Tem15] use at least five of the seven piece types. In particular, [BDH+04, Section 6.2] mentions various sets of five piece types that suffice. What about fewer piece types?

Our main results are the first to make progress on this question: for any *size-2* subset $A \subseteq$ { I, O, T, S, Z, J, L }, Tetris clearing is NP-complete with pieces restricted to A. Most pairs of piece types require different constructions for their reductions; refer to Table 1. Our results require us to specify more details of the piece rotation model, specifically what happens when the player rotates a piece in a way that collides with a filled square. We assume the **Super Rotation System (SRS)** [Tetb], first introduced in the 2001 game *Tetris Worlds* and as part of the Tetris Company's Tetris Guideline for how all modern (2001+) Tetris games should behave [Tetc].

For every size-2 subset A of piece types, we also establish #P-hardness for the corresponding problem of counting the number of ways to clear the board. Here we distinguish solutions by the final placement of each piece, not the sequence of moves to make those placements (as long as the placement is valid). This definition lets us ignore e.g. the null effect of moving a piece repeatedly left and right.

For certain size-3 subsets of piece types, we further establish ASP-completeness for Tetris clearing. Recall that an NP search problem is **ASP-complete** [YS03] if there is a parsimonious reduction from all NP search problems (including a polynomial-time bijection between solutions). In particular, ASP-completeness implies NP-hardness of finding another solution given k solutions, for any $k \geq 0$, as well as #P-completeness. These results hold for piece types { I, T, L } and { I, T, J }.

We also study Tetris under two more restrictive models on piece moves:

- **Hard drops only**: In this model, pieces do not move downward on their own, and if the player moves a piece downward, the piece moves maximally downward before locking into place (a **hard drop** maneuver). The player is still free to rotate or move the piece left or right before hard-dropping the piece. This model is motivated by most Tetris games awarding higher scores for hard drops, and was posed in [BDH+04].

- **20G**: In this model, instead of periodically moving down one unit, all pieces move maximally downward *instantly and on their own*, and the player is not allowed to control how fast a piece moves downward. The player is still free to rotate or move the piece left or right before the piece locks.

This model is motivated by levels with the maximum possible gravity, as in Level 20+ of regular Tetris with 20 rows [Teta].

For certain size-2 subsets of piece types, we establish NP-hardness of both Tetris survival and clearing under either of these models. Table 1 labels which of our Tetris clearing results hold in which models.

Along the way, we prove new results about 3-Partition and Numerical 3-Dimensional Matching (3DM): both problems are strongly ASP-complete even when all integers are assumed distinct. These results are of independent interest for ASP-hardness reductions. Previously, these problems were known to be ASP-complete with multisets of integers [BDD+20], and strongly NP-complete with distinct integers [HWW08].

1.2 Outline

The structure of the rest of the paper is as follows. Section 2 details the Super Rotation System (SRS), an important aspect of modern Tetris and used in our constructions. Section 3 proves ASP-completeness of 3-Partition with Distinct Integers and Numerical 3-Dimensional Matching with Distinct Integers, two problems we reduce from. Section 4 discusses our hardness results for Tetris clearing with SRS with only two piece types. Section 5 discusses some Tetris survival results under the "hard drops only" and "20G" models. Section 6 proves ASP-completeness of Tetris clearing with SRS.

2 Super Rotation System (SRS)

Most previous Tetris results are not sensitive to exactly how Tetris pieces rotation: most reasonable rotation models work [BDH+04, Section 6.4]. By contrast, many of the results in this paper focus specifically on (and require) the **Super Rotation System (SRS)** [Tetb], defined as follows.

Each piece has a defined **rotation center**, as indicated by dots in Figure 1, except for ▦▦▦▦ and ▦, whose rotation centers are the centers of the 4 × 4 squares in Figure 1. When unobstructed, all non-▦ tetrominoes will rotate purely about the rotation center (note that ▦ pieces cannot rotate). The key feature about SRS is **kicking**: if a tetromino is obstructed when a rotation is attempted, the game will attempt to "kick" the tetromino into one of four alternate positions, each tested sequentially; if all four positions do not work, then the rotation will fail. See Figure 2 for an example of this kicking process. The full data for wall kicks can be found in Tables 2 and 3, and at [Tetb]. Of note is that SRS wall kicks are vertically symmetric for all pieces or pairs of pieces (i.e., ▦▦ ↔ ▦▦ and ▦▦ ↔ ▦▦) except for the ▦▦▦▦ piece, so all rotations can be mirrored.

	Test 1	Test 2	Test 3	Test 4	Test 5
$0 \rightarrow R$	$(0,0)$	$(-1,0)$	$(-1,+1)$	$(0,-2)$	$(-1,-2)$
$R \rightarrow 0$	$(0,0)$	$(+1,0)$	$(+1,-1)$	$(0,+2)$	$(+1,+2)$
$R \rightarrow 2$	$(0,0)$	$(+1,0)$	$(+1,-1)$	$(0,+2)$	$(+1,+2)$
$2 \rightarrow R$	$(0,0)$	$(-1,0)$	$(-1,+1)$	$(0,-2)$	$(-1,-2)$
$2 \rightarrow L$	$(0,0)$	$(+1,0)$	$(+1,+1)$	$(0,-2)$	$(+1,-2)$
$L \rightarrow 2$	$(0,0)$	$(-1,0)$	$(-1,-1)$	$(0,+2)$	$(-1,+2)$
$L \rightarrow 0$	$(0,0)$	$(-1,0)$	$(-1,-1)$	$(0,+2)$	$(-1,+2)$
$0 \rightarrow L$	$(0,0)$	$(+1,0)$	$(+1,+1)$	$(0,-2)$	$(+1,-2)$

Table 2: Kick data for ▦▦, ▦▦, ▦▦, ▦▦, and ▦▦ pieces. 0 indicates the default orientation, and R, 2, and L indicate the orientation reached from a 90°, 180°, and 270° rotation clockwise (respectively) from the default orientation. An ordered pair (a, b) denotes a translation of the center by a units in the x direction and b units in the y direction. Positive x direction is rightwards, and positive y direction is upward.

This system of kicking tetrominoes during rotations allows for moves which are often called **twists** or **spins**. All the spins that we utilize are detailed in Appendix A.
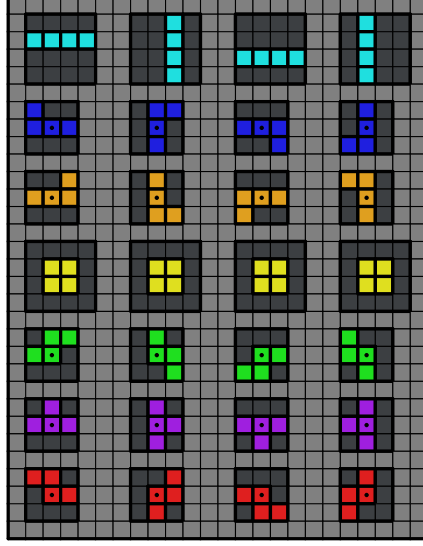
Figure 1: All tetromino pieces, in order from top to bottom: ▭▭▭▭, ▙▄, ▟▟, ▦, ▟▟, ▛▜, ▜▙. The first column is the default orientation of a piece upon spawning in; each column to the right indicates a 90° rotation clockwise about the rotation center of the piece.
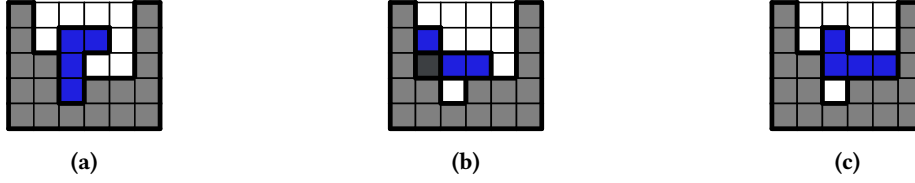

| (a) | (b) | (c) |

Figure 2: An example of the SRS kick system. Suppose the ▙▄ piece in (a) is being rotated 90° counter-clockwise. Test 1 (which is $(0, 0)$) would fail, due to the dark gray square shown in (b). Test 2 (which is $(+1, 0)$) would succeed, as shown in (c), and so the ▙▄ piece would rotate to the position in (c).

3 3-Partition and Numerical 3DM with Distinct Integers

Our reductions to Tetris are all from one of the following two problems, which are strengthenings of two standard strongly NP-complete problems:

Problem 3.1 (3-Partition with Distinct Integers). *Given a **set** $A = \{a_1, a_2, \ldots, a_n\}$ of n **distinct** positive integers such that $\frac{t}{4} < a_i < \frac{t}{2}$ for each i, where $t = \frac{3}{n} \sum_{i=1}^{n} a_i$, determine whether there is a partition of A into $\frac{n}{3}$ groups $D_1, \ldots, D_{n/3}$ (each necessarily of size 3) having the same sum $\sum_{x \in D_j} x = t$.*

Problem 3.2 (Numerical 3-Dimensional Matching (3DM) with Distinct Integers). *Given three **sets***

$$A = \{a_1, a_2, \ldots, a_n\}, B = \{b_1, b_2, \ldots, b_n\}, \text{ and } C = \{c_1, c_2, \ldots, c_n\}$$

*of n positive integers, where all 3n integers are **distinct**, and a target sum $t = \frac{1}{n} \sum_{i=1}^{n} (a_i + b_i + c_i)$, determine whether there is a partition of $A \cup B \cup C$ into n groups $D_1, \ldots, D_n$, each with exactly one element from each of A, B, and C, and $\sum_{x \in D_j} x = t$ for all j.*

Without the "distinct" and "set" conditions, both problems are well-known to be ***strongly NP-complete***, meaning that the problem is NP-hard even if the a_i integers are bounded by a polynomial in n. This property makes it feasible to represent each integer a_i (and t) in unary, which is the approach taken by all

	Test 1	Test 2	Test 3	Test 4	Test 5
$0 \to R$	$(0,0)$	$(-2,0)$	$(+1,0)$	$(-2,-1)$	$(+1,+2)$
$R \to 0$	$(0,0)$	$(+2,0)$	$(-1,0)$	$(+2,+1)$	$(-1,-2)$
$R \to 2$	$(0,0)$	$(-1,0)$	$(+2,0)$	$(-1,+2)$	$(+2,-1)$
$2 \to R$	$(0,0)$	$(+1,0)$	$(-2,0)$	$(+1,-2)$	$(-2,+1)$
$2 \to L$	$(0,0)$	$(+2,0)$	$(-1,0)$	$(+2,+1)$	$(-1,-2)$
$L \to 2$	$(0,0)$	$(-2,0)$	$(+1,0)$	$(-2,-1)$	$(+1,+2)$
$L \to 0$	$(0,0)$	$(+1,0)$	$(-2,0)$	$(+1,-2)$	$(-2,+1)$
$0 \to L$	$(0,0)$	$(-1,0)$	$(+2,0)$	$(-1,+2)$	$(+2,-1)$

Table 3: Kick data for ▭▭▭▭ pieces, with same notation as Table 2.

past Tetris NP-hardness proofs [BDH+04, DDE+17, ACD+20, Tem15], as then the total reduction size is still polynomial in n.

We want to ensure all integers are distinct in order to have more control over our reductions' blowup in the number of solutions, as needed for #P- and ASP-hardness. Bosboom et al. [BDD+20] proved that numerical 3DM is strongly ASP-complete when A is restricted to be a set, but allowed for B and C to be multisets as usual, and did not forbid repeated integers between A, B, C. Hulett, Will, and Woeginger [HWW08] proved that both 3-Partition and Numerical 3DM remain strongly NP-hard with distinct integers. We extend their proof to obtain ASP-completeness:

Theorem 3.3. *3-Partition with Distinct Integers, and Numerical 3-Dimensional Matching with Distinct Integers, are strongly ASP-complete.*

To prove this result, we use the following intermediate problems (which are thus also ASP-complete):

Problem 3.4 (Positive 1-in-3SAT). *Given a boolean formula in 3CNF (i.e., an AND of clauses consisting of 3 literals), where all literals are positive, does there exist an assignment of the variables to either true or false such that each clause has exactly one literal set to true?*

Problem 3.5 (Tripartite Edge-Disjoint Triangle Partition). *Given an undirected tripartite graph $G = (V, E)$, can we partition E into disjoint triangles?*

Proof of Theorem 3.3. We give a chain of parsimonious reductions from 3SAT, which is known to be ASP-complete [YS03]:

1. **3SAT → Positive 1-in-3SAT**: Hunt, Marathe, Radhakrishnan, and Stearns [HIMRS98, Theorem 3.8] gave such a parsimonious reduction.[1] See also [BDD+20, Lemma 2.1].

2. **Positive 1-in-3SAT → Tripartite Edge-Disjoint Triangle Partition**: We follow a simplification of a reduction from FCP 1-in-3SAT to FCP Tripartite Edge-Disjoint Triangle Partition [DMW+18, Theorem 12], which in turn is based on a reduction from 3SAT to Tripartite Edge-Disjoint Triangle Partition [Hol81]. By reducing from Positive 1-in-3SAT, we simplify the reduction of [DMW+18] by avoiding negative literals.

   We represent each variable by a sufficiently large triangular grid of vertices, with opposite sides of a parallelogram identified to form a flat torus, as shown in Figure 3a. This grid has exactly two solutions, corresponding to true (the triangles in Figure 3a) and false (the triangles in Figure 3b); note that the two solutions consist of exactly the same edges, and cover each exactly once. For each clause

---

[1] Their problem "1-Ex3MonoSat" is Positive 1-in-3SAT with the additional constraint that every clause has exactly three literals. Their reduction is also planarity preserving, so chaining with their parsimonious reduction from 3SAT to Planar 3SAT, we obtain that Planar 1-in-3SAT is also ASP-complete.
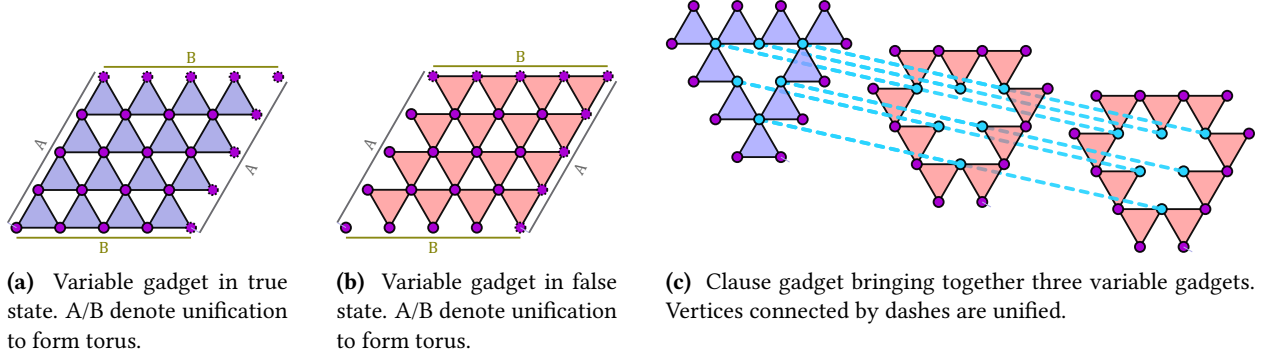
(a) Variable gadget in true state. A/B denote unification to form torus.

(b) Variable gadget in false state. A/B denote unification to form torus.

(c) Clause gadget bringing together three variable gadgets. Vertices connected by dashes are unified.

Figure 3: Reduction from Positive 1-in-3SAT to Tripartite Edge-Disjoint Triangle Partition.

(x, y, z), we pick one triangle of positive orientation, remove its edges, and unify the corresponding vertices of these triangles and of the three neighboring triangles of negative orientation, as shown in Figure 3c. Exactly one variable must choose the true state so as to cover the edges surrounding the unified hole exactly once. By choosing the variable gadgets large enough, we can ensure that the clause gadgets are disjoint from each other. Each gadget has a unique way to implement a given assignment, so this reduction is parsimonious.

3. **Tripartite Edge-Disjoint Triangle Partition → Numerical 3DM with Distinct Integers**: We combine a chain of reductions, from Triangle Edge-Disjoint Triangle Partition to Latin Square Completion [Col84], and from Latin Square Completion to Numerical 3DM with Distinct Integers [HWW08].

   If $U = \{u_1, u_2, \ldots\}, V = \{v_1, v_2, \ldots\}, W = \{w_1, w_2, \ldots\}$ is the vertex tripartition, then we do the following:

   - Let $q = 2\max\{|U|, |V|, |W|\}$, and let the target sum be $t = 19q^6$.
   - Map each edge (u_i, w_k) to $2q^6 + iq - k \in A$.
   - Map each edge (v_j, w_k) to $7q^6 + jq^2 + k \in B$.
   - Map each edge (u_i, v_j) to $t - (9q^6 + jq^2 + iq) = 10q^6 - jq^2 - iq \in C$.

   The lemmas in [HWW08] show that all the integers in A, B, and C are distinct (i.e., we have a valid instance of Numerical 3DM with Distinct Integers); and that any triple summing to t consists of one element each from A, B, and C, with the elements corresponding to a triangle in the graph. Thus we obtain a bijection between triangle partitions and Numerical 3DM solutions, i.e., the reduction is parsimonious.

4. **Numerical 3DM with Distinct Integers → 3-Partition with Distinct Integers**: We use standard techniques to relate these problems. Convert each integer a_i, b_i, and c_i in Numerical 3DM to integers $8a_i+1$, $8b_i+2$, and $8c_i-3$, respectively, in 3-Partition; and convert t to $8t$. In particular, all integers are still distinct, because we scale up by a factor of 8 and then shift values by less than 4. Furthermore, working modulo 8, every triple of integers summing to t must take exactly one a_i, one b_j, and one c_k. Therefore we have a parsimonious reduction.

Composing these reductions, we obtain that 3-Partition with Distinct Integers, and Numerical 3DM with Distinct Integers, are ASP-hard. Both problems are NP search problems, so they are ASP-complete.

□

4 Tetris with Two Piece Types

In this section, we will prove that for any size-2 subset $A \subseteq \{$, , , , , , $\}$, Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the sequence of pieces given to the player only contains the piece types in A. We will also show that some of the reductions work under the "hard drop only" model and the "20G" model. Refer to Table 1 for a table of all of our results.

All of our reductions are from 3-Partition with Distinct Integers and are in the same flavor as the reduction for clearing 3-tris with rotation as given in the Total Tetris paper [DDE+17], which we will use some terminology from. In particular, the reductions will involve a starting board involving $\frac{n}{3}$ structures, which we will call "**bottles**", of equal height of $\Theta(t \cdot \text{poly}(n))$, spaced sufficiently far apart so that bottles do not interact with each other except for line clears, and possibly along with an additional structure, which we will call a "**finisher**", to the right of the rightmost bottle.

Each bottle consists of a neck portion with n constant-sized "**top segments**", a body portion with t poly(n)-sized "**units**", and possibly $O(n)$ extra lines either above the neck portion, between the "top segments", between the neck portion and the body portion, and/or below the body portion that get cleaned up after the rest of the lines. To simplify our arguments, we make the size of each unit larger than the size of the neck portion.

The finisher will be a structure that specifically prevents the rows in the body portion from clearing before all of the top segments are cleared, and is located in same rows as the body portion of the bottles when required. We will use three types of finishers, a finisher, an finisher (which is a vertically symmetric version of the finisher), and a finisher, shown in Figure 4. Note that the finishers can be adapted to any number of rows larger than 4, and there is exactly one way to clear each type of finisher.
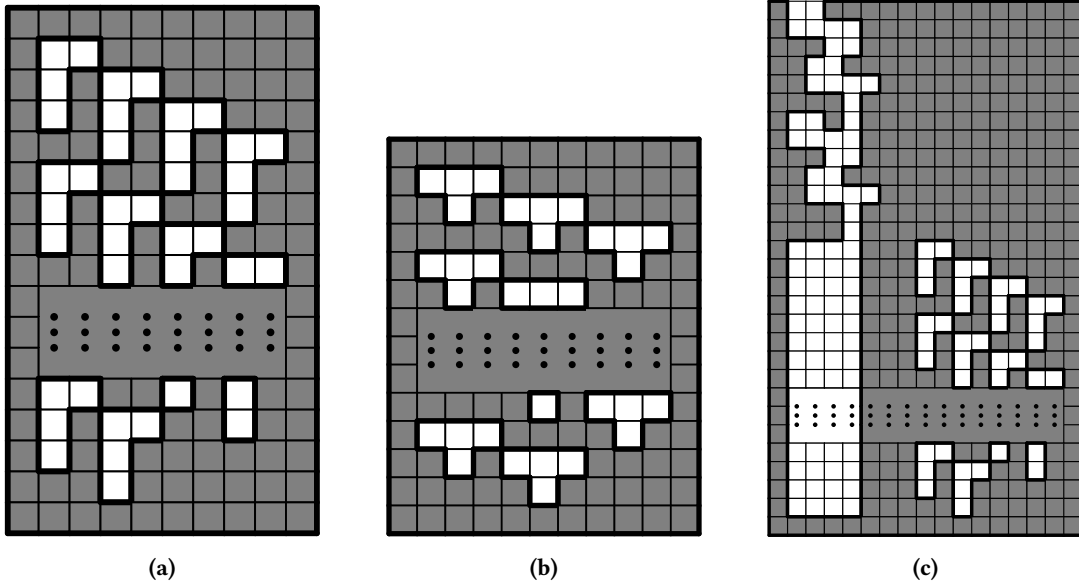

Figure 4: The and finishers (the finisher can be obtained by reflecting the finisher through a vertical line), and an example of the finisher next to a bottle (in the , setup).

For each element $a_i \in A$, we create a sequence of pieces S_i, which can be decomposed into three subsequences:

- **Priming sequence**: A piece sequence that, if used correctly, properly blocks all bottles but one in the same "top segment", and if used incorrectly, either directly "overflows" the bottle (i.e., puts blocks above the line under which all of our pieces must go) or "clogs" the bottle (i.e., improperly blocks

the bottle and prevents the player from being able to clear the lines necessary to re-open the bottle). For all of the bottle structures except for the one for { ⬛, ⬛ }, the pieces in the priming sequence cannot rotate or translate below the topmost "top segment" under SRS, and any piece placed into a "top segment" of a bottle prevents any piece in the filling sequence from reaching the body portion of that bottle.

- **Filling sequence**: A piece sequence of length $\Theta(a_i)$ that "fills" a_i units in the body portion of the unblocked bottle. If there are not enough units left to fill, then the pieces corresponding to one of the units will cause an overflow due to there not being enough empty space in the neck portion for all of the pieces (using the fact that the size of each unit is larger than the size of the neck portion).

- **Closing sequence**: A piece sequence that properly clears the lines corresponding to the "top segment" blocked by the priming sequence and resets the states of the neck portion of the bottles (albeit with one less "top segment").

We also have a **finale sequence** F, possibly the empty sequence, which helps clear any finishers on the board and the remaining lines on the board after the lines corresponding to the neck and body portions have been cleared.

In this section, when we write a sequence of pieces, we will use parentheses around sequences, commas between different piece types, and exponentiation to denote repeated pieces of the same piece type. For example, a sequence written as (⬛^{2}, ⬛^{3}, ⬛) consists of 2 ⬛s, 3 ⬛s, and an ⬛, in that order. The sequence of pieces given to the player will be of the form $(S_1, S_2, \ldots, S_n, F)$.

4.1 General Argument

We provide a very general argument for why these reductions work. If there exists a valid 3-partition $(D_1, \ldots, D_{n/3})$ for $\{a_1, a_2, \ldots, a_n\}$, then for each S_i, determine the corresponding j_i such that $a_i \in D_{j_i}$, then use the priming sequence to block all bottles properly except for the (j_i)th one, the filling sequence to fill a_i units in the body portion of the (j_i)th bottle, and the closing sequence to reset the states of the neck portion of the bottles. After all the S_i are used in this way, all lines corresponding to the "top segments" will be cleared as there are n such "top segments" with each S_i clearing exactly one of them, and each bottle will be filled to exactly t units. Thus, in the case where there are no pieces in the finale sequence, the lines corresponding to the body portions of the bottles will be cleared, meaning that no lines remain and we have cleared the board, and in the case where there are pieces in the finale sequence, the only lines that remain are those that can be cleared by the finale sequence. Thus, the sequence $(S_1, S_2, \ldots, S_n, F)$ can clear the board.

Conversely, if the sequence $(S_1, S_2, \ldots, S_n, F)$ can clear the board, then we claim that there is a corresponding 3-partition for $\{a_1, a_2, \ldots, a_n\}$. In particular, for each S_i, the priming sequence must properly block all but one bottle, say the (j_i)th bottle, forcing all the pieces in the filling sequence into the (j_i)th bottle. The filling sequence must then fill exactly a_i units in the (j_i)th bottle before the closing sequence, and it must do so without overfilling the body portion of the bottle, as otherwise there will be an overflow in that bottle. In particular, this means that, for each $1 \le j \le \frac{n}{3}$, the sum of the a_i corresponding to the S_j that filled some units in the jth bottle must be at most t. However, since $\sum a_i$ is exactly $t\left(\frac{n}{3}\right)$, the sum of the a_i corresponding to the S_j that filled some units in the jth bottle must actually be exactly t. In other words, there is a way to partition the a_i into $\frac{n}{3}$ subsets $D_1, \ldots, D_{n/3}$ such that the sum of the elements in each subset is t. Thus, there is a corresponding 3-partition for $\{a_1, a_2, \ldots, a_n\}$.

This general argument shows how YES instances of the two problems (3-Partition with Distinct Integers, Tetris clearing with SRS and restricted piece types) are equivalent, and hence that this reduction works. The rest of the subsections in this section show how to set up the bottles for each size-2 subset of piece types.

4.2 Subsets with ▭▭▭▭ Pieces

First we show how the reduction in the Total Tetris paper [DDE+17] can be easily adapted to any subset of pieces with ▭▭▭▭ pieces plus an additional piece type:

Proposition 4.1. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if:*

- *The type of pieces in the sequence given to the player is restricted to any of { ▭▭▭▭ , J }, { ▭▭▭▭ , L }, { ▭▭▭▭ , O }, { ▭▭▭▭ , S }, { ▭▭▭▭ , T }, or { ▭▭▭▭ , Z };*

- *The model being considered is "hard drops only" and type of pieces in the sequence given to the player is restricted to any of { ▭▭▭▭ , J }, { ▭▭▭▭ , L }, { ▭▭▭▭ , O }, or { ▭▭▭▭ , T }; or*

- *The model being considered is "20G" and the type of pieces in the sequence given to the player is restricted to either { ▭▭▭▭ , J } or { ▭▭▭▭ , L }.*

Proof. Refer to Figures 5 (which shows the bottle structure for { ▭▭▭▭ , L }) and 6 (which shows the bottle structure for the rest of the subsets). We focus on the reduction for { ▭▭▭▭ , L }; the rest of the reductions are similar. No finishers are required for any of the setups.
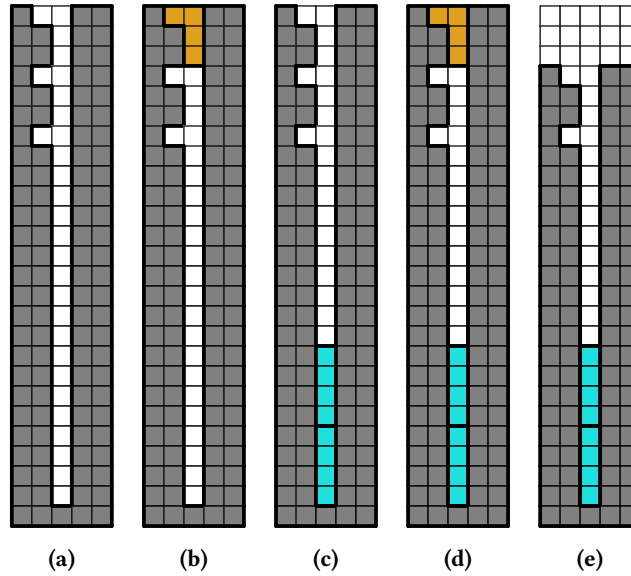
Figure 5: The bottle structure for { ▭▭▭▭ , L }. (b) shows how an L piece must block a bottle during the priming sequence. (c) shows the result of a filling sequence where $a_i = 2$. (d) shows where the L piece in the closing sequence must go. (e) shows the result of the closing sequence.

Here, the "top segments" are the L-shaped holes at the top of the bottle, and a "unit" is one $8n \times 1$ rectangle towards the bottom of the bottle (note that the size of the neck portion is $4n$, which is smaller than $8n$, the size of a "unit"); no extra lines are necessary. Each a_i is encoded by the sequence of pieces ($L^{n/3-1}$, ▭▭▭▭^{2na_i}, L). The priming sequence is ($L^{n/3-1}$), the filling sequence is (▭▭▭▭^{2na_i}), and the closing sequence is (L).

Our finale sequence is empty.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

The proofs for the other piece types are similar: replace with , , , , or , respectively, and use the corresponding bottle structure in Figure 6. Notice that the and reductions require the / spin in Figure 27 to rotate the piece into the uppermost "top segment".

These reductions are $((\frac{n}{3})!(\frac{n}{3}-1)!^n)$-monious:[2] for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, and for each a_i sequence, there are $(\frac{n}{3}-1)!$ ways to permute which piece in the priming sequence blocks which bottle. As such, we can also conclude #P-hardness.
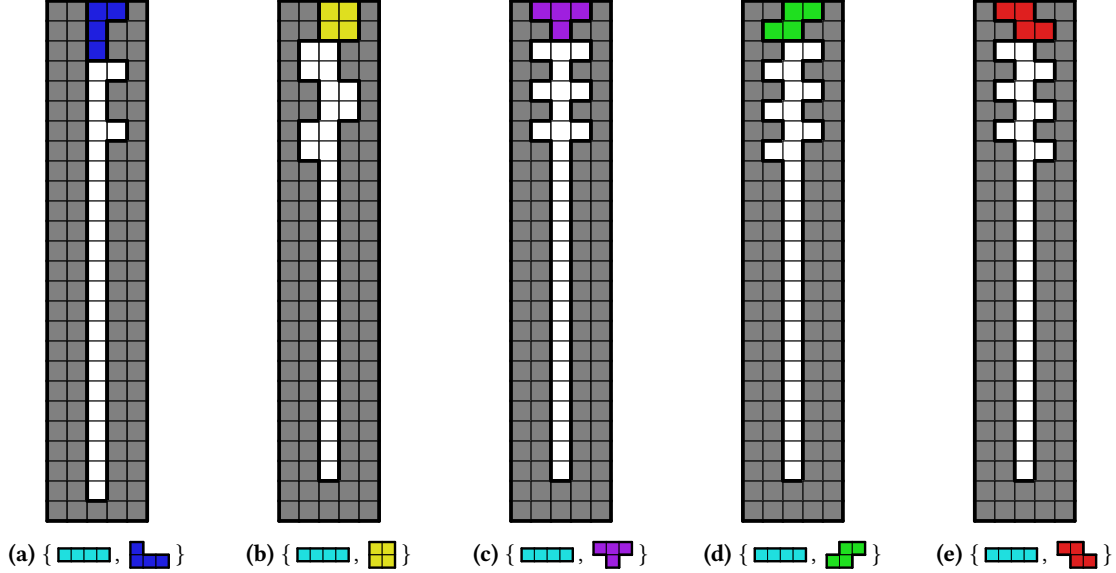

Figure 6: The bottle structures for the other subsets containing , including how the non- piece must block a bottle during the priming and closing sequence.

Notice that the { , }, { , }, { , }, and { , } set-ups do not require any last-second rotations (and in fact do not require the kicking system of SRS), and so all pieces can be hard-dropped from the top of the board for the { , }, { , }, { , }, and { , } set-ups.

In addition, because all buckets are two units wide in the { , } and { , } set-ups and the , , and pieces are at least three units wide when oriented horizontally, the { , } and { , } set-ups work even if pieces experience 20G gravity: when any piece drops, orient the piece horizontally, move it left or right to the desired bucket, and rotate accordingly. □

4.3 Other Subsets with Pieces

We now move on to all the remaining subsets which contain pieces.

Proposition 4.2. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either* { , } *or* { , }.

Proof. First we discuss the { , } case. Refer to Figure 7(a), which shows the bottle structure for { , }. We will also use a finisher in our setup to prevent rows in the body portion from clearing early.

---

[2]We call a reduction ***c-monious*** if it blows up the number of solutions by exactly a multiplicative factor of *c*.
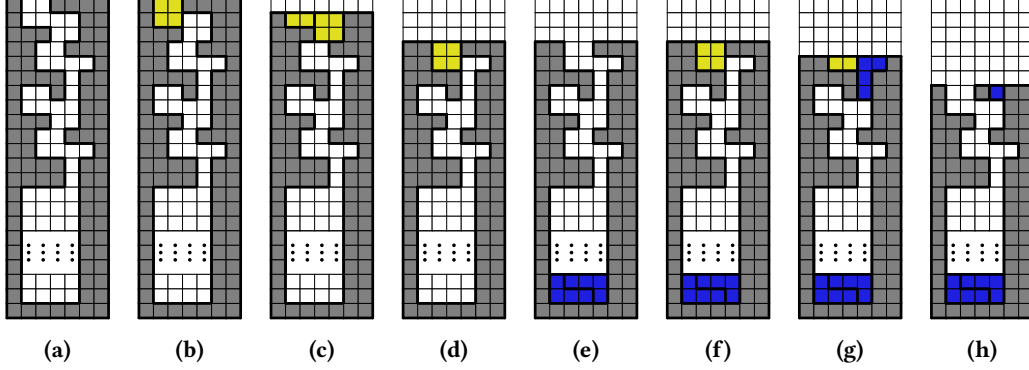
Figure 7: The bottle structure for { ▦, ▟ }. (b–c) show how the first two ▦ pieces per bottle must fit during the priming sequence. (d) shows how the ▦ pieces must block a bottle during the priming sequence. (e) shows the result of a filling sequence (requires the ▟ -spin sequence in Figure 25 to get ▟ pieces through the neck portion and the ▟ spin in Figure 20 to get ▟ pieces into the body portion). (f–g) show where the pieces in the closing sequence must go. (h) shows the result of the closing sequence.

Here, a "top segment" is a pattern of 16 squares repeating every 7 lines at the top of the bottle, and a "unit" is one $8n \times 4$ rectangle towards the bottom of the bottle (note that the size of the neck portion is $16n$, which is smaller than $32n$, the size of a "unit").

Each a_i is encoded by the sequence of pieces ($▦^{n-1}$, $▟^{8na_i}$, ▦, $▟^{n/3}$). The priming sequence is ($▦^{n-1}$), the filling sequence is ($▟^{8na_i}$), and the closing sequence is (▦, $▟^{n/3}$), with the ways to block, fill, and re-open a bottle properly shown in Figure 7. Discussion on improper piece placements can be found in Appendix B.1.

The finale sequence is ($▟^{8nt-2}$) to clear the ▟ finisher.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

Now, for our #P-hardness analysis, let $f_4(m)$ denote the number of ways to place m ▟ pieces in an rectangle of width 4 and height m to fill the rectangle exactly, using Tetris rules. This reduction is $\left((\frac{n}{3})! \left(((\frac{n}{3})!)^3 (\frac{n}{3}-1)!\right)^n (f_4(8nt))^{n/3}\right)$-monious, as for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, $((\frac{n}{3})!)^2(\frac{n}{3}-1)!$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence (the first $\frac{n}{3}$ pieces must be placed as in Figure 7(b), then the next $\frac{n}{3}$ pieces must be placed as in Figure 7(c), before we can reach Figures 7(d–f)), $(\frac{n}{3})!$ ways to permute how the ▟ pieces in the closing sequence get placed for each a_i sequence, and $f_4(8nt)$ ways to permute how the ▟ pieces get placed within the body portion of each bottle. As such, we can also conclude #P-hardness.

We get a similar argument for { ▦, ▄ } by vertical symmetry. □

A demo of the { ▦, ▟ } bottle structure can be found at https://jstris.jezevec10.com/map/80188.

Proposition 4.3. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either { ▦, ▟ } or { ▦, ▟ }.*

Proof. First we discuss the { ▦, ▟ } case. Refer to Figure 8(a), which shows the bottle structure for { ▦, ▟ }. We do not use a finisher in our setup.

Here, a "top segment" is the pattern that repeats every 3 lines at the top of the bottle, and a "unit" is a region of area $16n$ consisting of $4n$ ▟-shaped pieces stacked on top of each other. Three extra lines are necessary, corresponding to an extra "top segment". Note that the size of the neck portion plus the area between the neck and body portion is $8n + 6$, which is smaller than $16n$, the size of a "unit".
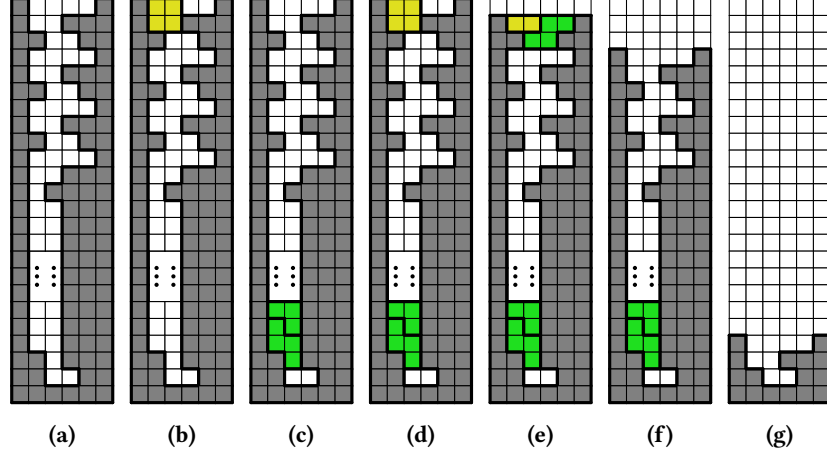
Figure 8: The bottle structure for { ⊞, ⬛ }. (b) shows how the ⊞ pieces must block a bottle during the priming sequence. (c) shows the result of a filling sequence (requires the ⬛-spin sequence in Figure 28 to get ⬛ pieces through the neck portion and the ⬛ spin in Figure 26 to get ⬛ pieces into the body portion). (d–e) show where the pieces in the closing sequence must go (requires the ⬛ spin in Figure 27). (f) shows the result of the closing sequence. (g) shows the state of the board immediately before the finale sequence.

Each a_i is encoded by the sequence of pieces ($⊞^{n/3-1}$, $⬛^{4na_i}$, ⊞, $⬛^{n/3}$). The priming sequence is ($⊞^{n/3-1}$), the filling sequence is ($⬛^{4na_i}$), and the closing sequence is (⊞, $⬛^{n/3}$), with the ways to block, fill, and re-open a bottle properly shown in Figure 8. Discussion on improper piece placements can be found in Appendix B.2.

The finale sequence is ($⊞^{n/3}$, $⬛^{n/3}$); the remaining lines collapse down to another copy of the "top segment" (see Figure 8(g)) and can be cleared in the same fashion as in the priming sequence and the closing sequence.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

This reduction is $\left(\left(\frac{n}{3}\right)! \left(\left(\frac{n}{3}-1\right)! \left(\frac{n}{3}\right)! \right)^n \left(\left(\frac{n}{3}\right)! \right)^2 \right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $\left(\frac{n}{3}\right)!$ ways to "permute" which subsets correspond to which bottles, $\left(\frac{n}{3}-1\right)!$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence, $\left(\frac{n}{3}\right)!$ ways to permute how the ⬛ pieces in the closing sequence get placed for each a_i sequence, and $\left(\left(\frac{n}{3}\right)! \right)^2$ ways to permute how the pieces in the finale sequence get placed. As such, we can also conclude #P-hardness.

We get a similar argument for { ⊞, ⬛ } by vertical symmetry. □

A demo of the { ⊞, ⬛ } bottle structure can be found at https://jstris.jezevec10.com/map/81818.

Proposition 4.4. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to* { ⊞, ⬛ }.

Proof. Refer to Figure 9(a), which shows the bottle structure for { ⊞, ⬛ }. We will also use a ⬛ finisher in our setup to prevent rows in the body portion from clearing early.

Here, a "top segment" is the pattern that repeats every 4 lines at the top of the bottle, and a "unit" is a region of area $16n$ consisting of the 8-line pattern right underneath the lowermost "top segment" in Figure 9(a) repeated n times (note that the size of the neck portion is $12n$, which is smaller than $16n$, the size of a "unit"). No extra lines are necessary.

Each a_i is encoded by the sequence of pieces ($⊞^{n/3-1}$, $⬛^{4na_i}$, ⊞, $⬛^{n/3}$, $⊞^{n/3}$). The priming sequence is ($⊞^{n/3-1}$), the filling sequence is ($⬛^{4na_i}$), and the closing sequence is (⊞, $⬛^{n/3}$, $⊞^{n/3}$),
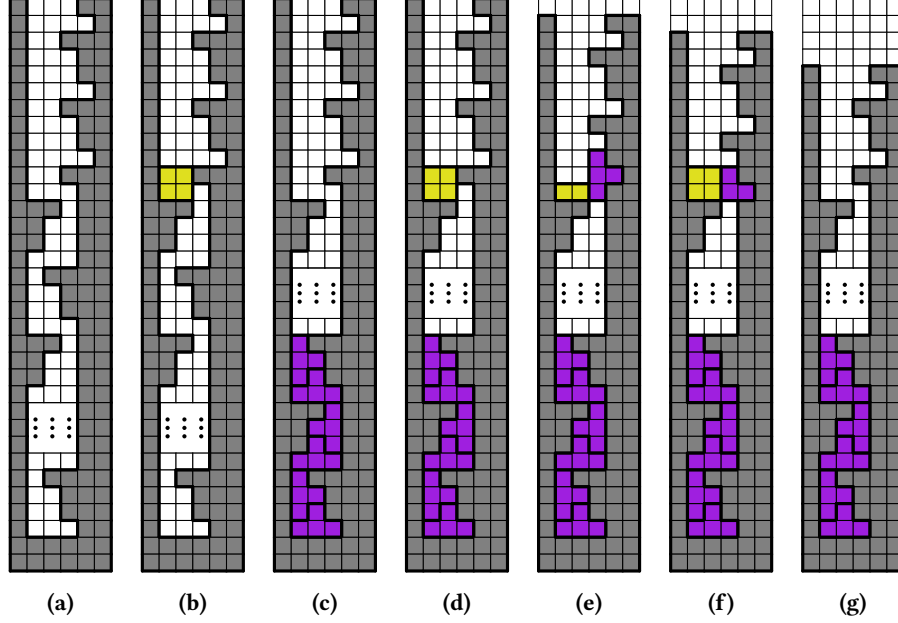
Figure 9: The bottle structure for { ⊞ , ⊤ }. (b) shows how the ⊞ piece must block a bottle during the priming sequence. (c) shows the result of a filling sequence (requires the ⊤-spin sequence in Figure 30 to get ⊤ pieces through the body portion). (d–f) show where the pieces in the closing sequence must go. (g) shows the result of the closing sequence.

with the ways to block, fill, and re-open a bottle properly shown in Figure 9. Discussion on improper piece placements can be found in Appendix B.3.

The finale sequence is (⊤ $^{8nt-2}$) to clear the ⊤ finisher.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

This reduction is $\left(\left(\frac{n}{3}\right)!\left(\left(\frac{n}{3}-1\right)!\left(\left(\frac{n}{3}\right)!\right)^2\right)^n\right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $\left(\frac{n}{3}\right)!$ ways to "permute" which subsets correspond to which bottles, $\left(\frac{n}{3}-1\right)!$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence, and $\left(\left(\frac{n}{3}\right)!\right)^2$ ways to permute how the ⊤ pieces and ⊞ pieces in the closing sequence get placed for each a_i sequence. As such, we can also conclude #P-hardness. □

A demo of the { ⊞ , ⊤ } bottle structure can be found at https://jstris.jezevec10.com/map/80169.

4.4 Two-Element Subsets of { ◰ , ⊤ , ◳ }

Proposition 4.5. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either { ◰ , ⊤ } or { ⊤ , ◳ }.*

Proof. First we discuss the { ◰ , ⊤ } case. Refer to Figure 10(a), which shows the bottle structure for { ◰ , ⊤ }. We do not use a finisher in our setup.

Here, a "top segment" is the pattern that repeats every 4 lines at the top of the bottle, and a "unit" is a region of area $16n$ consisting of $4n$ ◰-shaped pieces stacked on top of each other. Four extra lines are used: two between the neck portion and the body portion, and two under the body portion. Note that the size of the neck portion plus the area between the neck and body portion is $8n + 5$, which is smaller than $16n$, the size of a "unit".
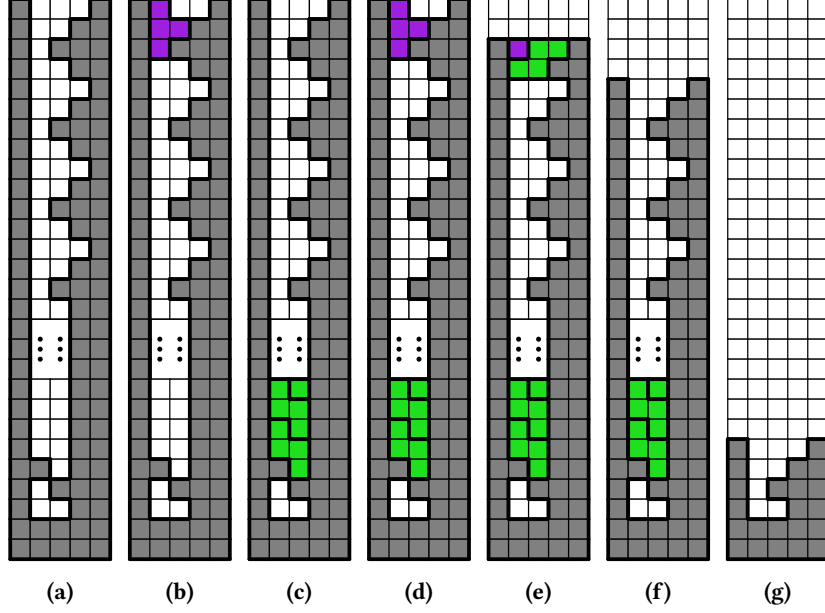
Figure 10: The bottle structure for { , }. (b) shows how the pieces must block a bottle during the priming sequence. (c) shows the result of a filling sequence (requires the -spin sequence in Figure 29 to get pieces through the neck portion and the spin in Figure 26 to get pieces into the body portion). (d–e) show where the pieces in the closing sequence must go (requires the spin in Figure 27). (f) shows the result of the closing sequence. (g) shows the state of the board immediately before the finale sequence.

Each a_i is encoded by the sequence of pieces ($^{n/3-1}$, ^{4na_i}, , $^{n/3}$). The priming sequence is ($^{n/3-1}$), the filling sequence is (^{4na_i}), and the closing sequence is (, $^{n/3}$), with the ways to block, fill, and re-open a bottle properly shown in Figure 10. Discussion on improper piece placements can be found in Appendix B.4.

The finale sequence is ($^{n/3}$, $^{n/3}$); the remaining lines collapse down to another copy of the "top segment" (see Figure 10(g)) and can be cleared in the same fashion as described in the priming sequence and the closing sequence.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

This reduction is $\left(\left(\frac{n}{3}\right)!\left(\left(\frac{n}{3}-1\right)!\right)^n\left(\left(\frac{n}{3}\right)!\right)^{n+2}\right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $\left(\frac{n}{3}\right)!$ ways to "permute" which subsets correspond to which bottles, $\left(\frac{n}{3}-1\right)!$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence, $\left(\frac{n}{3}\right)!$ ways to permute how the pieces in the closing sequence get placed for each a_i sequence, and $\left(\left(\frac{n}{3}\right)!\right)^2$ ways to permute how the pieces in the finale sequence get placed. As such, we can also conclude #P-hardness.

We get a similar argument for { , } by vertical symmetry. □

A demo of the { , } bottle structure can be found at https://jstris.jezevec10.com/map/80184.

Proposition 4.6. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to { , }.*

Proof. Refer to Figure 11(a), which shows the bottle structure for { , }. We do not use a finisher in our setup.

Here, a "top segment" is the -shaped region of area 4, and a "unit" is a region of area $16n$ consisting of $4n$ -shaped pieces stacked on top of each other. We use $O(n)$ extra rows: one row with a 1×3 hole
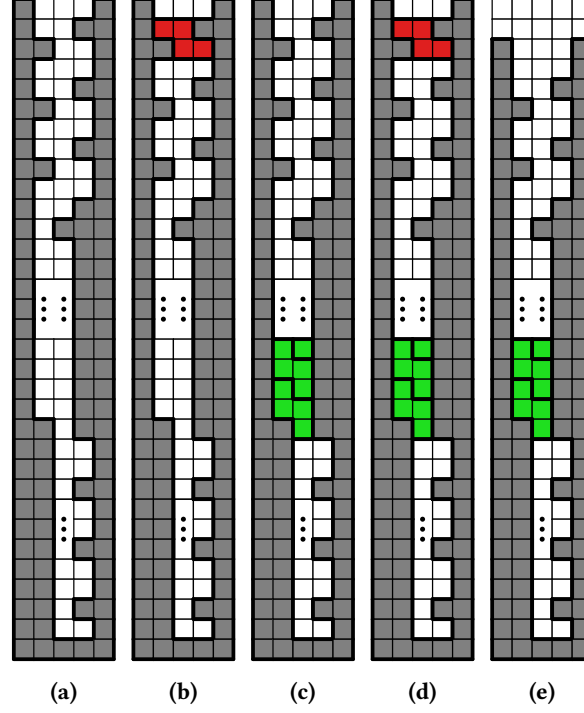
Figure 11: The bottle structure for { ⬛, ⬛ }. (b) shows how the ⬛ piece must block a bottle during the priming sequence (requires the ⬛-piece version of the ⬛ spin in Figure 27). (c) shows the result of a filling sequence (requires the ⬛-spin sequence in Figure 28 to get ⬛ pieces through the neck portion and the ⬛ spin in Figure 26 to get ⬛ pieces into the body portion). (d) shows where the piece in the closing sequence must go. (e) shows the result of the closing sequence.

above each top segment, two rows between the bottommost "top segment" and the body portion, and $3(n + 1) + 1$ other rows below the body portion. Note that the size of the region in the bottle above the body portion is $7n + 5$, which is smaller than $16n$, the size of a "unit".

Each a_i is encoded by the sequence of pieces (⬛ $^{n/3-1}$, ⬛ ^{a_i}, ⬛). The priming sequence is (⬛ $^{n/3-1}$), the filling sequence is (⬛ ^{a_i}), and the closing sequence is (⬛), with the ways to block, fill, and re-open a bottle properly shown in Figure 11. Discussion on improper piece placements can be found in Appendix B.5.

The finale sequence is (⬛ $^{n/3+2n(n+1)/3}$). The remaining lines will collapse down to another structure as indicated in Figure 12(a). The only way to clear the structure is to use the first $\frac{n}{3}$ pieces to clear two rows as indicated in Figure 12(b), and then repeatedly use $2n/3$ ⬛ pieces to clear four rows per iteration as indicated in Figure 12(c–e) until the board is cleared.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

This reduction is $\left((\frac{n}{3})!((\frac{n}{3}-1)!)^n((\frac{n}{3})!)^{2n+3}\right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, $(\frac{n}{3}-1)!$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence and $((\frac{n}{3})!)^{2n+3}$ ways to permute how the pieces in the finale sequence get placed (the only re-ordering allowed is within each group of $\frac{n}{3}$ ⬛ pieces). As such, we can also conclude #P-hardness. □

A demo of the { ⬛, ⬛ } bottle structure can be found at https://jstris.jezevec10.com/map/80198.

15
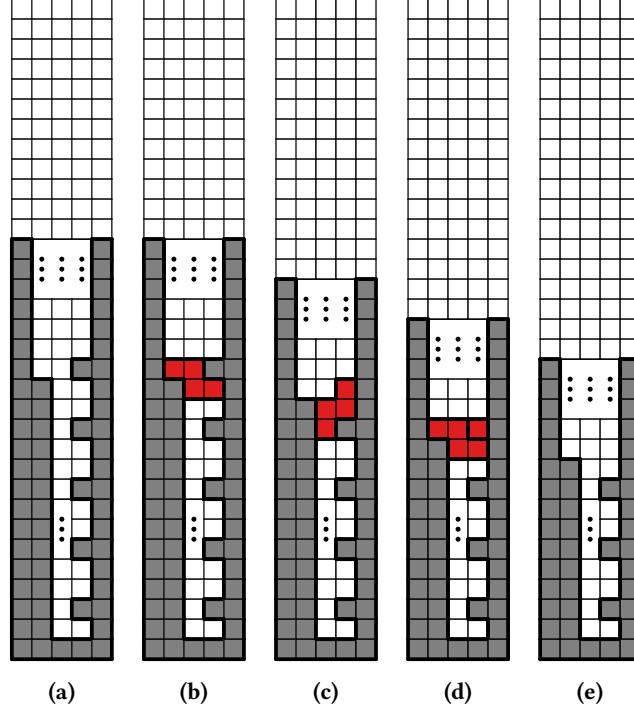
Figure 12: The finale structure for { ⬛, 🟥 }, including how to clear rows using 🟥 pieces (requires the 🟥-piece version of the 🟩 spin in Figure 27).

4.5 Remaining Subsets with More Complex Structures

Proposition 4.7. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to any of* { 🟦, 🟪 }, { 🟦, 🟥 }, { 🟧, 🟩 }, *or* { 🟧, 🟪 }.

Proof. First we discuss the { 🟦, 🟥 } and { 🟦, 🟪 } cases. Refer to Figure 13(a), which shows the bottle structure for { 🟦, 🟥 } and { 🟦, 🟪 }. We will also use a 🟦 finisher in our setup to prevent rows in the body portion from clearing early.

Here, a "top segment" is the pattern that repeats every 6 lines at the top of the bottle, and a "unit" is one $6n \times 4$ rectangle (note that the size of the neck portion is $16n$, which is smaller than $24n$, the size of a "unit"). No extra lines are necessary.

For { 🟦, 🟥 }, each a_i is encoded by the sequence of pieces (🟥 $^{2n/3-2}$, 🟦 ^{6na_i}, 🟥 ^{2}, 🟦 $^{2n/3}$). The priming sequence is (🟥 $^{2n/3-2}$), the filling sequence is (🟦 ^{6na_i}), and the closing sequence is (🟥 ^{2}, 🟦 $^{2n/3}$), with the ways to block, fill, and re-open a bottle properly shown in Figure 13. Discussion on improper piece placements can be found in Appendix B.6.

The finale sequence is (🟦 $^{6nt-2}$) to clear the 🟦 finisher.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

Now, for our #P-hardness analysis, let $f_4(m)$ denote the number of ways to place m 🟦 pieces in an rectangle of width 4 and height m to fill the rectangle exactly. This reduction is $\left((\frac{n}{3})! \left(\frac{(2n/3-2)!((n/3)!)^2}{2^{(n/3-1)}} \right)^n (f_4(6nt))^{n/3} \right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, $\frac{(2n/3-2)!}{2^{(n/3-1)}}$ ways to permute how the pieces in the priming sequence get placed for each a_i sequence (the only restriction is
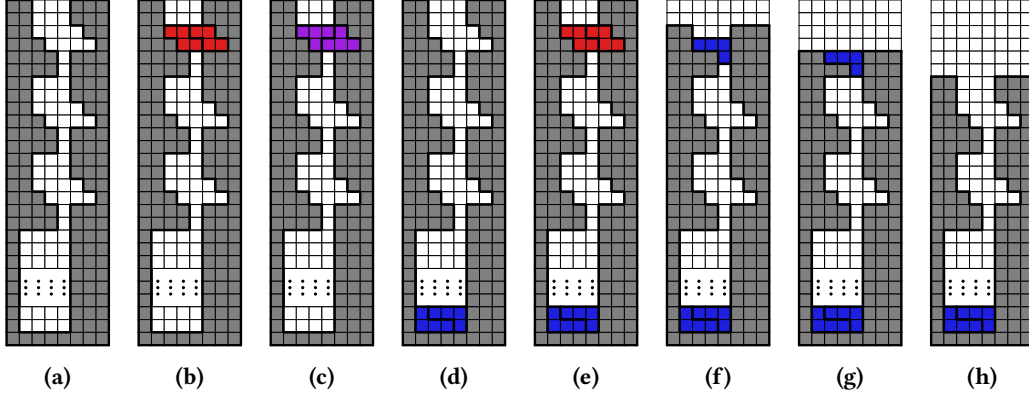
Figure 13: The bottle structure for { ▦ , ▦ } and { ▦ , ▦ }. (b) and (c) show how the ▦ pieces or ▦ pieces must block a bottle during the priming sequence (requires the ▦-piece version of the ▦ spin in Figure 27 for the second ▦ piece). (d) shows the result of a filling sequence (requires the ▦ spin in Figure 20 to get ▦ pieces through the neck portion and into the body portion). (e–g) show where the pieces in the closing sequence must go. (h) shows the result of the closing sequence.

on how the two ▦ pieces blocking the same bottle get placed), $((\frac{n}{3})!)^2$ ways to permute how the ▦ pieces in the closing sequence get placed for each a_i sequence, and $f_4(6nt)$ ways to permute how the ▦ pieces get placed within the body portion of each bottle. As such, we can also conclude #P-hardness.

For { ▦ , ▦ }, we use the exact same bottle structure, and have each a_i be encoded by the sequence of pieces (▦ $^{2n/3-2}$, ▦ ^{6na_i}, ▦ ^{2}, ▦ $^{2n/3}$). The way to block a bottle is slightly different and is shown in Figure 13(c) and the cases to check for the priming sequence are slightly different (see Appendix B.6), but the proof is overall the same.

We get a similar argument for { ▦ , ▦ } and { ▦ , ▦ } by vertical symmetry. □

A demo of the { ▦ , ▦ } bottle structure can be found at https://jstris.jezevec10.com/map/80205.

Proposition 4.8. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either { ▦ , ▦ } or { ▦ , ▦ }.*

Proof. First we discuss the { ▦ , ▦ } case. Refer to Figure 14(a), which shows the bottle structure for { ▦ , ▦ }. We will also use a ▦ finisher in our setup to prevent rows in the body portion from clearing early.

Here, a "top segment" is the pattern that repeats every 8 lines at the top of the bottle, and a "unit" is one $6n \times 4$ rectangle (note that the size of the neck portion is $20n$, which is smaller than $24n$, the size of a "unit"). No extra lines are necessary.

Each a_i is encoded by the sequence of pieces (▦ $^{n/3-1}$, ▦ ^{6na_i}, ▦ , ▦ $^{4n/3}$). The priming sequence is (▦ $^{n/3-1}$), the filling sequence is (▦ ^{6na_i}), and the closing sequence is (▦ , ▦ $^{4n/3}$), with the ways to block, fill, and re-open a bottle properly shown in Figure 14. Discussion on improper piece placements can be found in Appendix B.7.

The finale sequence is (▦ $^{6nt-2}$) to clear the ▦ finisher.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

Now, for our #P-hardness analysis, let $f_4(m)$ denote the number of ways to place m ▦ pieces in an rectangle of width 4 and height m to fill the rectangle exactly. This reduction is $\left((\frac{n}{3})!\left((\frac{n}{3}-1)!(\frac{n}{3})!^4\right)^n (f_4(6nt))^{n/3}\right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, $(\frac{n}{3}-1)!$ ways to
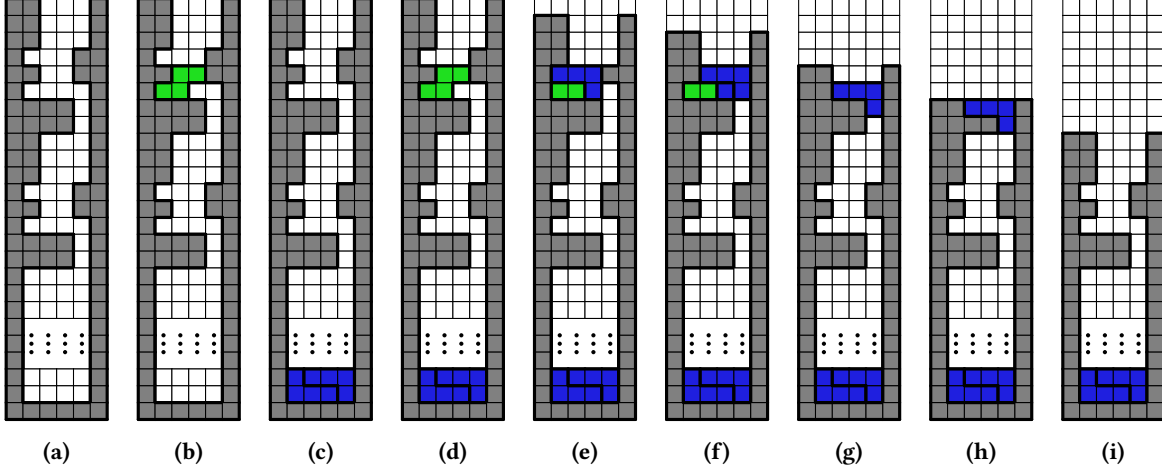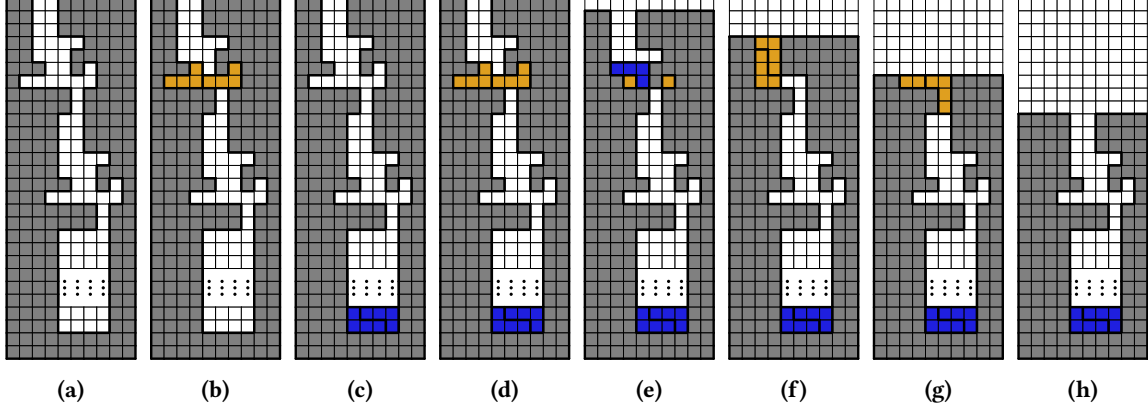
17

Figure 14: The bottle structure for { ⬛, 🟩 }. (b) shows how the 🟩 piece must block a bottle during the priming sequence. (c) shows the result of a filling sequence (requires the ⬛ spins in Figures 22 and 23 to get ⬛ pieces through the neck portion and into the body portion). (d–h) show where the pieces in the closing sequence must go (requires the ⬛ spin in Figure 21). (i) shows the result of the closing sequence.

permute how the 🟩 pieces in the priming sequence get placed for each a_i sequence, $((\frac{n}{3})!)^4$ ways to permute how the ⬛ pieces in the closing sequence get placed for each a_i sequence, and $f_4(6nt)$ ways to permute how the ⬛ pieces get placed within the body portion of each bottle. As such, we can also conclude #P-hardness.

We get a similar argument for { 🟧, 🟥 } by vertical symmetry. $\qquad\square$

A demo of the { ⬛, 🟩 } bottle structure can be found at https://jstris.jezevec10.com/map/83069.

Proposition 4.9. *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to { ⬛, 🟧 }.*

Proof. Refer to Figure 15(a), which shows the bottle structure for { ⬛, 🟧 }. We will also use a ⬛ finisher in our setup to prevent rows in the body portion from clearing early.

Here, a "top segment" is the pattern that repeats every 9 lines at the top of the bottle, and a "unit" is one $8n \times 4$ rectangle (note that the size of the neck portion is $24n$, which is smaller than $32n$, the size of a "unit"). Notice that the "top segments" are horizontally offset from each other in order to align the bottom lines of a "top segment" to the top lines of the next "top segment". We will also use a ⬛ finisher in our setup to prevent rows in the body portion from clearing early.

Each a_i is encoded by the sequence of pieces (🟧$^{2n/3-2}$, ⬛^{8na_i}, 🟧^{2}, ⬛$^{n/3}$, 🟧^{n}). The priming sequence is (🟧$^{2n/3-2}$), the filling sequence is (⬛^{8na_i}), and the closing sequence is (🟧^{2}, ⬛$^{n/3}$, 🟧^{n}), with the ways to block, fill, and re-open a bottle properly shown in Figure 15. Discussion on improper piece placements can be found in Appendix B.8.

The finale sequence is (⬛$^{8nt-2}$) to clear the ⬛ finisher.

Since we have all of the required components, we can apply the general argument from Section 4.1 to conclude NP-hardness.

Now, for our #P-hardness analysis, let $f_4(m)$ denote the number of ways to place m ⬛ pieces in an rectangle of width 4 and height m to fill the rectangle exactly. This reduction is $\left((\frac{n}{3})! \left(\frac{(2n/3-2)!(2n/3)!((n/3)!)^2}{2^{(2n/3-1)}} \right)^n (f_4(8nt))^{n/3} \right)$-monious: for each solution to the 3-Partition with Distinct Integers instance, there are $(\frac{n}{3})!$ ways to "permute" which subsets correspond to which bottles, $\frac{(2n/3-2)!}{2^{(n/3-1)}}$

Figure 15: The bottle structure for { ⬛, 🟧 }. (b) shows how the 🟧 pieces must block a bottle during the priming sequence (requires the 🟧-piece versions of the ⬛ spins in Figures 23 and 24)). (c) shows the result of a filling sequence (requires the ⬛ spins in Figures 23 and 20 to get ⬛ pieces through the neck portion and into the body portion). (d–g) show where the pieces in the closing sequence must go. (h) shows the result of the closing sequence.

ways to permute how the pieces in the priming sequence get placed for each a_i sequence (the only restriction is on how the two 🟧 pieces blocking the same bottle get placed), $(\frac{n}{3})! \times \frac{(2n/3)!(n/3)!}{2^{n/3}}$ ways to permute how the ⬛ pieces and 🟧 pieces in the closing sequence get placed for each a_i sequence, and $f_4(8nt)$ ways to permute how the ⬛ pieces get placed within the body portion of each bottle. As such, we can also conclude #P-hardness. □

A demo of the { ⬛, 🟧 } bottle structure can be found at https://jstris.jezevec10.com/map/80195. We also note the following:

Conjecture 4.10. *The reduction in the proof of Proposition 4.9 works even if pieces experience 20G gravity.*

The reasoning is similar to the reasoning given for the proof of Proposition 4.1 (the holes at the top of the bottles are always at most 2 units wide); however, some more verification is required to make sure that the necessary rotations can be done in 20G gravity.

4.6 Putting It All Together

Combining all of these results, we get the following result:

Theorem 4.11. *For any size-2 subset $A \subseteq \{$ 🟦, 🟨, 🟪, 🟩, 🟥, ⬛, 🟧 $\}$, Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to the piece types in A.*

Proof. Propositions 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, and 4.9 cover all size-2 subsets of piece types, as shown in Table 1; combining all of the reductions, we obtain the desired result. □

Remark. All of our reductions involve a linear-factor blowup in the a_i for the filling sequences (i.e., we use $\Theta(na_i)$ pieces in the filling sequences); this makes it easier to argue about what happens when an overfill happens and makes the bottle analogy more fitting (since the neck portion is smaller while the body portion is much larger) but makes our reductions somewhat inefficient. Perhaps it is possible to reduce the blowup to a constant factor, though the argument may be a bit more complex.

19

5   Tetris Survival: Hard Drops Only and 20G

The previous section mentions NP-hardness of Tetris clearing under the "hard drops only" and "20G" Tetris models. Previous results about general Tetris [DDE⁺17, ACD⁺20] have also proven NP-hardness of Tetris survival, so in this section, we prove that, in both of these variants, Tetris survival is NP-hard using { ▭, ▦ } pieces. This improves upon a result by Temprano [Tem15] which proves hardness for "hard drops only" mode using the piece subset { ▰, ▰, ▭, ▰, ▰ }.

Theorem 5.1. *Tetris survival is NP-complete in the "hard drops only" and "20G" game modes, even if the type of pieces in the sequence given to the player is restricted to { ▭, ▦ }.*

Proof. We reduce from 3-Partition with Distinct Integers using a similar bottle structure to other proofs in this paper. From a 3-Partition with Distinct Integers instance, we create a setup consisting of $\frac{n}{3}$ width 1 buckets, each of height $4t$, separated by width 1 columns. In addition, we create a bucket of height $t - 1$ on the left which is blocked by a single square and has an open square on the upper left diagonal. We add one additional column on the left to obtain an even width board. We leave two rows at the top of the board empty. See Figure 16(a) for details. Each a_i is encoded by the sequence (▦ $^{n/3+1}$, ▭ ^{a_i}, ▦).
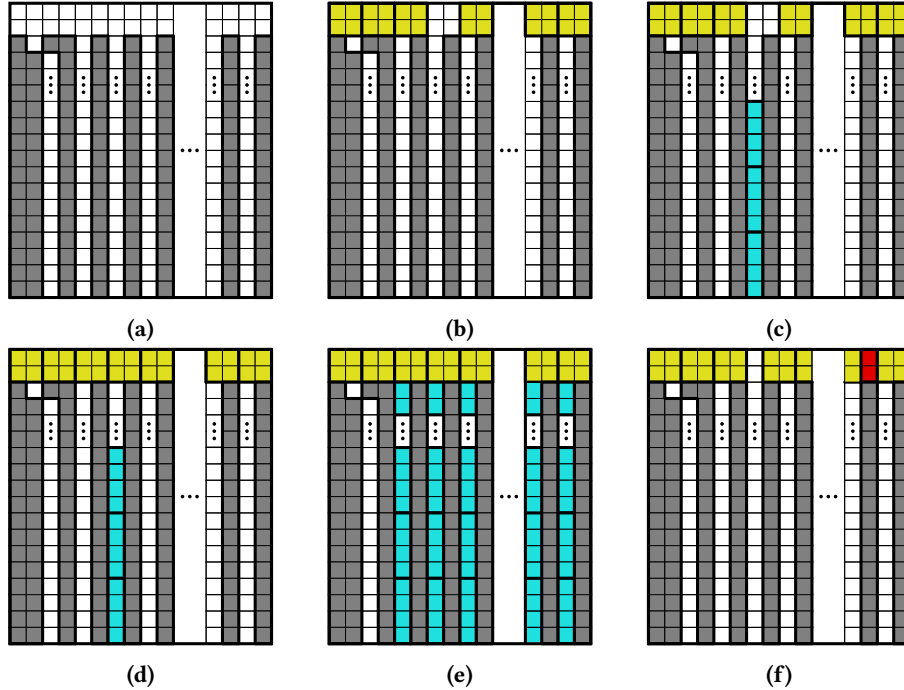
Figure 16: The bucket structure for "hard drops only" and "20G" Tetris modes. (b) shows how beginning sequence must block all but one bucket. (c) shows the result of a filling sequence. (d) shows how the closing sequence clears the top two rows. (e) shows a full board. (f) shows improper handling during the priming sequence.

The priming sequence is (▦ $^{n/3+1}$). Due to parity constraints, the player is forced to block all but one of the buckets using these pieces. They can choose to leave either one two-block wide gap or two one-block wide gaps in the top two rows. However, if they choose to leave one-block gaps, they will create spaces that can never be filled without overflowing the screen and causing a game loss (see the squares highlighted in red in Figure 16(f)), so we can assume that they will leave a 2×2 gap, as shown in Figure 16(b).

The filling sequence is (▭ ^{a_i}); the ▭ pieces must be placed in the pre-selected bucket, as shown in Figure 16(c). If pieces are not placed in accordance with a correct partition, then there will at some point

be an extra [I] piece which does not fit in the open bucket. This will cause the player to lose as an [I] piece has length 4 and cannot fit in the 2×2 gap in the top 2 rows (note that no [I] piece will stick partially out of a bucket since each bucket has a height that is a multiple of 4).

The closing sequence is ([O]). The [O] piece can only fit in the 2×2 square formed during the priming sequence (see Figure 16(e)), and it clears the top two rows, resetting the board to the initial state, except with a somewhat more filled bucket.

Once the player has received the entire sequence corresponding to each number in the 3-partition instance, including the final closing piece, the entire board will be full (see Figure 16(e)), with the exception of the extra inaccessible bucket. Because of this, the player must have filled each of the buckets to exactly the right height, solving the 3-partition instance, otherwise they would have lost.

For the entire duration of this sequence, every piece can be hard dropped into place as there are no overhangs in any of the buckets. Furthermore, each piece can be successfully maneuvered under 20G conditions. The [O] pieces cannot fall down any buckets, so they can be safely slid to any location in the top 2 rows, and the [I] pieces can move over the placed [O] pieces to the only open bucket. □

Corollary 5.2. *The above reduction can be extended to NP-hardness for board clearing under "hard drops only" or "20G" conditions if we also allow for [T] pieces.*

Remark. We have already proved that Tetris clearing under the "hard drops only" and "20G" models is hard with just two types of pieces. The primary reason for including hardness with a larger set of pieces is that this shows that, in both game modes, there is a board configuration and subset of pieces where the problems of surviving and clearing the board are both hard at the same time. In addition, our Tetris clearing results under the "20G" model does not include any proper subsets of $\{$ [I], [O], [T] $\}$, so this result is interesting in its own right.

Proof. We begin with the above proof that survivability with $\{$ [I], [O] $\}$ is hard. We extend the sequence with the finale sequence ([T], [I]^{t}, [O]$^{2n/3}$, [T]^{3}). Figure 17 shows the clearing process. We begin by using the first [T] piece to open the inaccessible bucket by clearing a row. We then use the [I] pieces to fill the previously inaccessible bucket, clearing all but the final two rows in the process. The final [I] piece protrudes one square from its bucket because of the row cleared by the first [T]. We use the [O] pieces to fill the $\frac{2n}{3} \times 4$ space on the right of the board. Finally, we place the remaining three [T] pieces to fill the remaining space and completely clear the board. All of these pieces can be placed in both special game modes. □
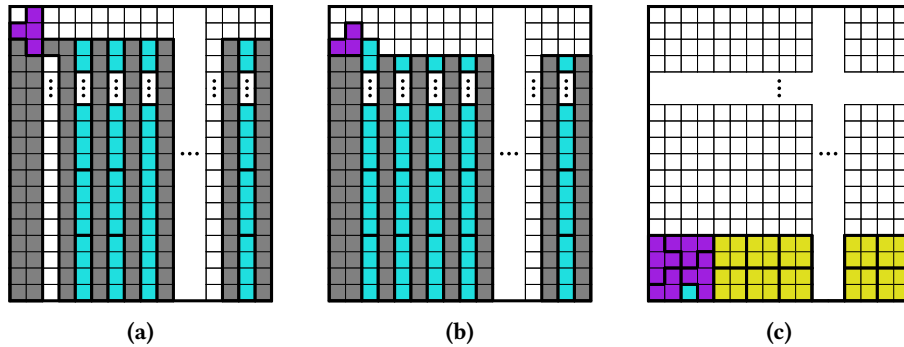
Figure 17: The clearing procedure for "hard drops only" and "20G" using $\{$ [I], [O], [T] $\}$

6 ASP-Completeness of Tetris

Even though the reductions in Section 4 are sufficient to prove #P-hardness, the reductions are not parsimonious, so they cannot be used to prove ASP-completeness. Indeed, the "blocking" bottles paradigm likely cannot be used to show ASP-completeness as there are many ways to permute the pieces that block all but one bottle. Thus, for ASP-completeness, we turn back to the "priming" buckets paradigm in [BDH+04]:

Theorem 6.1. *Tetris clearing with SRS is ASP-complete even if the type of pieces in the sequence given to the player is restricted to either* { ▬, ⊥, ⌐ } *or* { ▬, ⊥, ◣ }.

Proof. First we discuss the { ▬, ⊥, ⌐ } case. We give a parsimonious reduction from Numerical 3-Dimensional Matching with Distinct Integers; refer to Figure 18(a), which shows the bucket structure plus rightmost columns for { ▬, ⊥, ⌐ }.

(a)   (b)   (c)

(d)   (e)
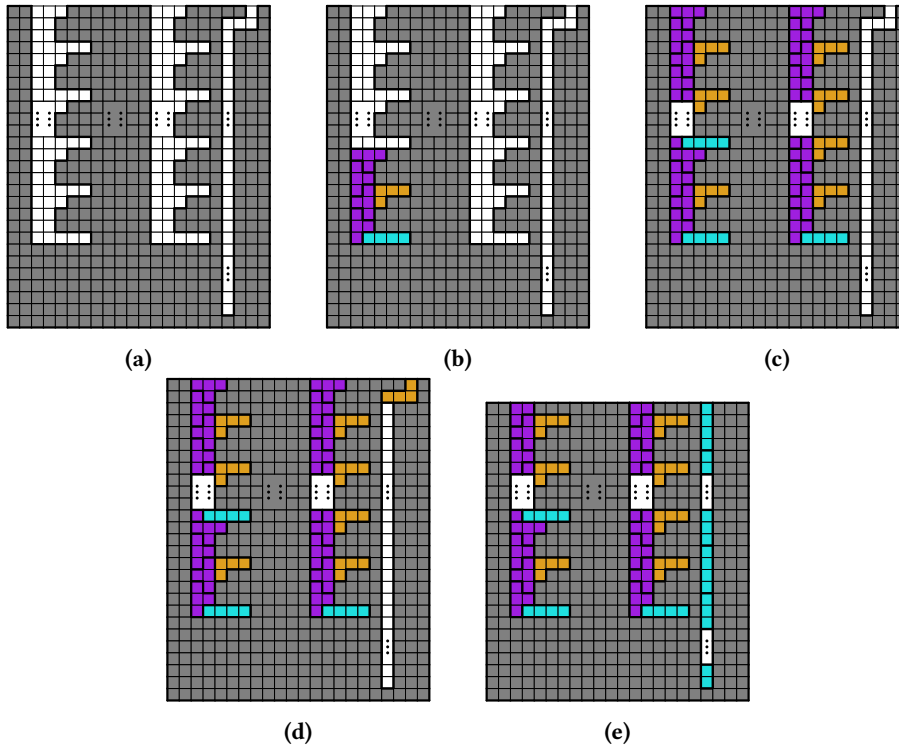
Figure 18: The bucket structure plus rightmost columns for { ▬, ⊥, ⌐ }. (b) shows how the ▬ piece must prime a bucket (requires the ▬ spin in Figure 19) and how the remainder of the pieces in a sequence must fit in the bucket ($m = 2$ here). (c) shows how the board looks like before the finale sequence. (d–e) show how the pieces in the finale sequence must be placed (requires the ⌐-piece version of the ◣ spin in Figure 23).

Here, a "unit" is the pattern that repeats every four rows. A positive integer m is encoded by the sequence of pieces (▬, (⊥, ⌐, ⊥)$^{m-1}$, ⊥, ⊥). The finale sequence is (⌐, ▬^{N}), where $N = \text{poly}(nt)$ is much larger than the height of the buckets, and is used to clear the rightmost columns after the buckets have been filled.

In this case, the ▬ piece serves as the "primer", and must be placed as indicated in Figure 18(b) (the placement is possible due to ▬ spins). The ⊥ pieces cannot be placed in a non-primed bucket without blocking off certain holes, particularly the squares in which an ▬ piece or an ⌐ piece must be placed, and misplacing an ⌐ piece (i.e., putting it in a different, non-primed bucket, putting it where the ⌐ piece goes during the finale sequence, or putting it too high in the bucket) causes the next two

 pieces to block off squares in which an piece or an piece must be placed. Thus, once the piece is placed in a bucket, the placements of the rest of the pieces encoding the positive integer m are forced. Further discussion on and figures for improper piece placements can be found in Appendix B.9.

Lastly, to make the reduction parsimonious, from the instance

$$A = \{a_1, a_2, \ldots, a_n\}, B = \{b_1, b_2, \ldots, b_n\}, \text{ and } C = \{c_1, c_2, \ldots, c_n\}$$

of Numerical 3-Dimensional Matching with Distinct Integers, we scale the a_i, b_i, and c_i and add/subtract constants such that each group must consist of exactly one a_i, one b_j, and one c_k (i.e., no group with at least two elements from any of A, B, or C can sum to the target sum). In addition, we "pre-fill" all of the buckets so that the ith bucket from the left is already filled up to a_i units, and the sequence of pieces given to the player consist only of the pieces in the b_i sequences, the pieces in the c_i sequences, and the pieces in the finale sequence. This ensures that the subsets have a fixed ordering and must consist of exactly one a_i, one b_j, and one c_k. Combined with the fact that all the piece placements after each piece in a sequence encoding a positive integer are forced, this means that each solution to the Numerical 3-Dimensional Matching instance corresponds to exactly one way to clear the Tetris board, and vice versa.

Therefore, we have a parsimonious reduction from Numerical 3-Dimensional Matching with Distinct Integers to Tetris clearing with { , , }, meaning that Tetris clearing with SRS is ASP-complete even if the type of pieces in the sequence given to the player is restricted to { , , }.

We get a similar argument for { , , } by vertical symmetry; the spin in Figure 19 still works when mirrored even though the kick tests for are not vertically symmetric.   □

Demos of the bucket structure can be found at https://jstris.jezevec10.com/map/80170 for { , , } and at https://jstris.jezevec10.com/map/83325 for { , , }.

7 Open Problems

One big open problem that still remains is the computational complexity of Tetris clearing with SRS if the player is only given one piece type (for example, if the sequence consists of entirely pieces). In this case, the "blocking" bottles paradigm no longer works, because the same piece type cannot be used both to block and to fill bottles without the reduction breaking, so a proof of hardness would involve an entirely different setup. It is also possible that Tetris clearing with SRS and with only one piece type is in P. For example, [BDH$^+$04] conjectures polynomial time for the piece type.

Similarly, it is open whether or not Tetris clearing with SRS is ASP-complete for subsets of pieces that are not supersets of { , , } or { , , }. One could likely construct similar structures for other 3-piece subsets, but arguing whether Tetris clearing with SRS is ASP-complete for 1- or 2-piece subsets may require different ideas.

Some open questions arise regarding whether our results can be extended if we consider different objectives or add additional features. For example, can we establish results for 2-piece subsets, similar to those in Section 4, for Tetris survival? If we add a "holding" function, where the player can put one piece aside for later use, can get similar results?

Modern variants of Tetris also use different random generators to ensure that the player does not receive the same piece arbitrarily many times in a row. One of the simplest random generators is called a **7-bag randomizer**. For this randomizer, the sequence of pieces is divided into groups (or "bags") of 7, each group containing one of each tetromino in one of 7! = 5,040 possible orderings. Can we show NP-hardness even if the sequence of pieces has to be able to be generated from this randomizer?

Acknowledgments

References

[ACD⁺20] Sualeh Asif, Michael Coulombe, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Jayson Lynch, and Mihir Singhal. Tetris is NP-hard even with $O(1)$ rows or columns. *Journal of Information Processing*, 28:942–958, 2020.

[BDD⁺20] Jeffrey Bosboom, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Roderick Kimball, and Justin Kopinsky. Path puzzles: Discrete tomography with a path constraint is hard. *Graphs and Combinatorics*, 36:251–267, 2020.

[BDH⁺04] Ron Breukelaar, Erik D. Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A. Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2):41–68, 2004.

[Col84] Charles J. Colbourn. The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984.

[DDE⁺17] Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Adam Hesterberg, Andrea Lincoln, Jayson Lynch, and Y. William Yu. Total Tetris: Tetris with monominoes, dominoes, trominoes, pentominoes, ... *Journal of Information Processing*, 25:515–527, 2017.

[Deb24] Sopan Deb. Boy, 13, is believed to be the first to 'beat' tetris. *The New York Times*, 3 January 2024. https://www.nytimes.com/2024/01/03/arts/tetris-beat-blue-scuti.html.

[DMW⁺18] Erik D. Demaine, Fermi Ma, Erik Waingarten, Ariel Schvartzman, and Scott Aaronson. The fewest clues problem. *Theoretical Computer Science*, 748:28–39, November 2018.

[HIMRS98] Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard E. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167, 1998.

[Hol81] Ian Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[HWW08] Heather Hulett, Todd G. Will, and Gerhard J. Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596, 2008.

[Tem15] Oscar Temprano. Complexity of a Tetris variant. arXiv:1506.07204, 2015.

[Teta] TetrisWiki. 20G. https://tetris.wiki/20G. Accessed: 2023-09-23.

[Tetb] TetrisWiki. Super Rotation System. https://tetris.wiki/Super_Rotation_System. Accessed: 2023-09-23.

[Tetc] TetrisWiki. Tetris guideline. https://tetris.wiki/Tetris_Guideline. Accessed: 2023-09-23.

[Tet23]   Tetris, 2023. Apple TV+ movie directed by Jon S. Baird and written by Noah Pink.

[Wik]   Wikipedia. Tetris. https://en.wikipedia.org/wiki/Tetris. Accessed: 2023-09-23.

[YS03]   Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, E86-A(5):1052–1060, 2003. Also IPSJ SIG Notes 2002-AL-87-2, 2002.

A Spins

In this appendix, we note the important spins used in our constructions. If the diagram has dark gray blocks indicated, then the dark gray blocks indicate the blocks required to perform the spin (as those blocks cause all the kick tests before the successful test to fail).

A.1 Spins involving ▦ Pieces

(a)　　　　　　　　　　(b)　　　　　　　　　　(c)
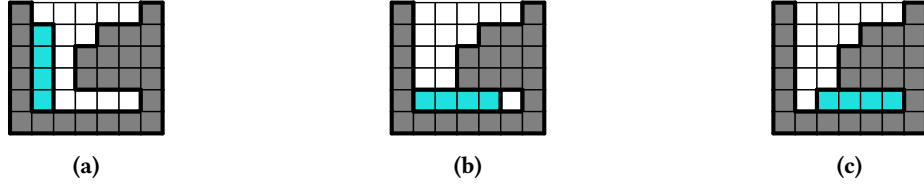
Figure 19: An ▦ spin used in the setup for ASP-completeness for the subset { ▦, ▦, ▦ }. Depending on the direction of rotation and the location of the rotation center (i.e. if the center is on the left side of the ▦ piece or the right side of the ▦ piece), the rotation will send the piece to either (b) or (c); however, no rotation will send the center of the piece upwards.

A.2 Spins involving ▦ / ▦ Pieces

(a)　　　　　　　　　　(b)

Figure 20: A ▦ spin used in the setups for { ▦, ▦ }, { ▦, ▦ }, and { ▦, ▦ }

(a)　　　　　　　　　　(b)

Figure 21: A ▦ spin used in the setup for { ▦, ▦ }

(a)　　　　　　　　　　(b)

Figure 22: A ▦ spin used in the setup for { ▦, ▦ }

(a)

(b)

Figure 23: A spin used in the setups for { , }, { , }, and the ASP-completeness for { , , }

(a)

(b)

Figure 24: A spin used in the setup for { , } (although the version of the spin used is the -spin version)

(a)

(b)

(c)

(d)

(e)
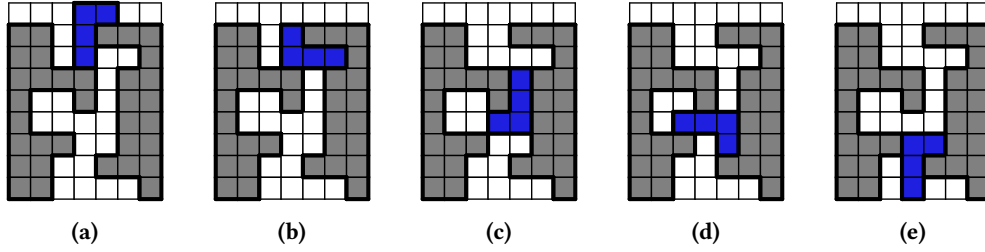
Figure 25: A sequence of spins used in the setup for { , }

A.3 Spins involving / Pieces

For these spins, the rotation center is important, so the rotation center is marked with a dot.

(a)

(b)

Figure 26: An spin used in the setups for { , }, { , }, and { , }, used for getting pieces into the body portions of the bottles

(a)   **(b)**   **(c)**
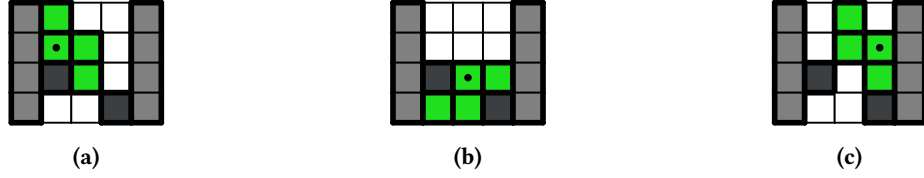
Figure 27: An ▦ spin ((a) to (b)) used in the setups for { ▦, ▦ }, { ▦, ▦ }, { ▦, ▦ }, { ▦, ▦ }, and { ▦, ▦ } (and similar), used either for clogging a bottle during the priming sequence or for re-opening a bottle during the closing sequence (or both). (c) indicates what happens if one more rotation is applied when the ▦ piece is tucked in as in (b).

(a)   **(b)**   **(c)**   **(d)**   **(e)**   **(f)**

Figure 28: A sequence of ▦ spins used in the setups for { ▦, ▦ } and { ▦, ▦ }

(a)   **(b)**   **(c)**   **(d)**   **(e)**   **(f)**

Figure 29: A sequence of ▦ spins used in the setup for { ▦, ▦ }

A.4 Spins involving ▦ Pieces

(a)   **(b)**   **(c)**
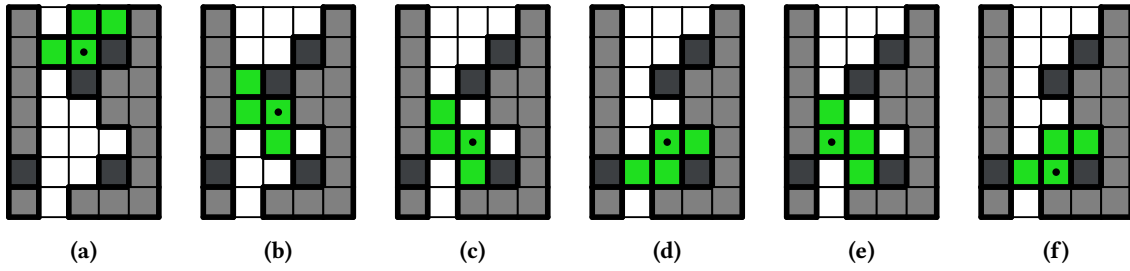
Figure 30: A ▦-spin sequence used in the setup for { ▦, ▦ }

B Clogs

In this appendix, we illustrate improper and impossible piece placements that do not directly overflow the bottle and show how the improper piece placements break the bottle setup (or indirectly cause overflow). In all of our analyses (except for { ⊞ , ⬛ } and the last clog for { ⊞ , ⬛ }), we assume that all top segments above where the clog happens, except for possibly the top segment directly above where the clog happens, get cleared properly (as we can always consider where the "highest" clog happens). Some of the piece placements that are not stated as being impossible may still not be possible, but we include them so that our analysis is thorough.

Note that the bottle structures in Figures 5 and 6 do not have nontrivial improper piece placements (i.e. piece placements that do not cause overflow), so we focus on other setups.

B.1 Clogs for { ⊞ , ⬛ }

There are no improper ways to place the pieces in the priming and closing sequences that do not directly overflow the bottle, so we only look at the improper ways to place ⬛ pieces in the filling sequences into the neck portion of a bottle, as in Figure 31.
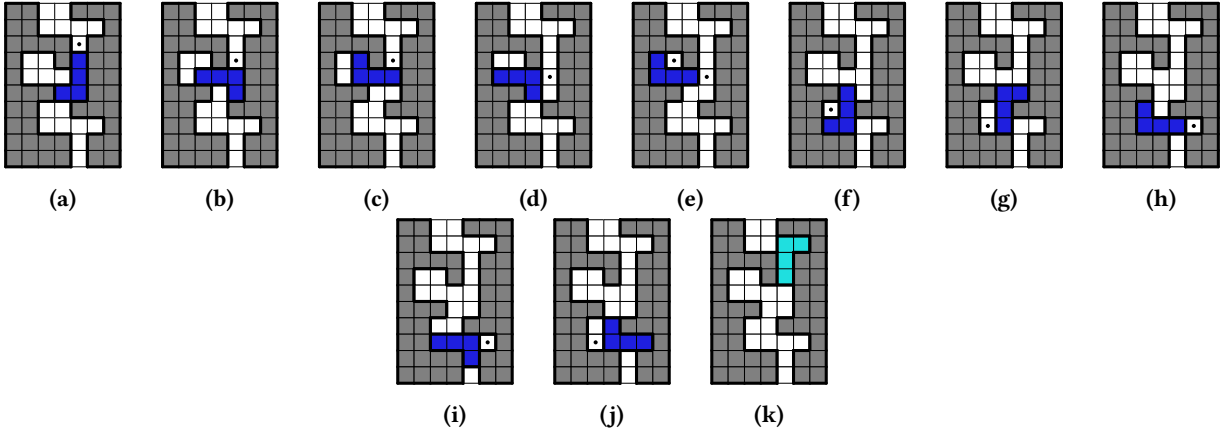
Figure 31: Clogs or impossible piece placements (in the case of (k)) involving ⬛ pieces in the { ⊞ , ⬛ } setup

The first five improper piece placements ((a) through (e)) prevent additional ⬛ pieces from rotating into the top segment the ⬛ piece is placed in, with the blue squares next to empty squares with dots causing issues and preventing the necessary spins from working, and attempting to fix the top segment via line clearing leads to overflow. The next five improper piece placements ((f) through (j)) block off one to two empty squares (indicated with dots) from being accessible, and attempting to clear lines and fit a piece covering those empty squares leads to overflow.

It is also not possible to rotate a ⬛ piece into the required placement in the top segment during a filling sequence (as in Figure 31(k)) because all of the rotations into that orientation change the y-coordinate of the center of the ⬛ piece by 0, +1, or −2, and of the possible y-coordinates the center of the ⬛ piece can start at, there are no valid ways to place a ⬛ piece in one of the two possible orientations pre-rotation without colliding with one of the gray squares.

B.2 Clogs for { ⊞ , ▟ }

There are no improper ways to place the pieces in the priming and closing sequences that do not directly overflow the bottle, so we only look at the improper ways to place ▟ pieces in the filling sequences into

the neck portion of a bottle, as in Figure 32 ((d)-(e) indicates a sequence of placements right above the body portion).
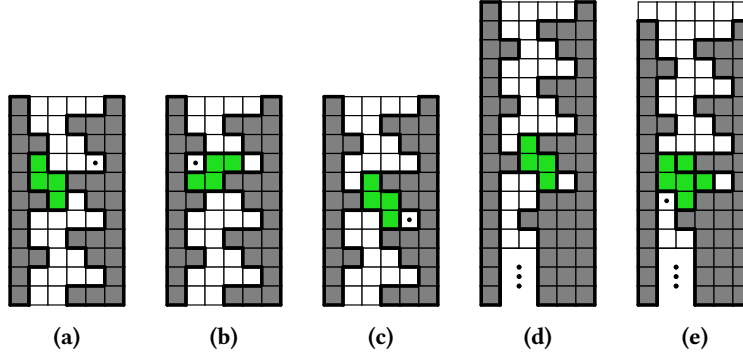
Figure 32: Clogs involving ⬛ pieces in the { ⬛, ⬛ } setup

For all of the improper piece placements in Figure 32, the empty square indicated with a dot no longer becomes fill-able without overflow.

B.3   Clogs for { ⬛ , ⬛ }

First we discuss improper piece placements involving ⬛ pieces in the priming sequence, which can be broken into two main cases: one indicated in Figure 33(a) and one indicated in Figure 33(b–c). The former shows that the ⬛ pieces cannot be placed outside of the first two columns without causing a empty square to become inaccessible, and the latter shows that we cannot place a second ⬛ piece from a priming or closing sequence into a bottle before a ⬛ piece, as no ⬛ piece can rotate in and cover the three empty squares with dots in Figure 33(c). Thus, all of the ⬛ pieces in the priming sequence must be placed in different bottles.
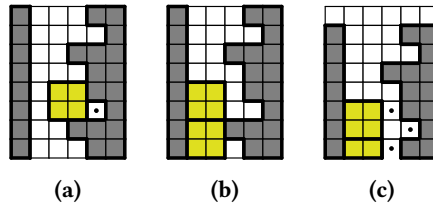
Figure 33: Clogs involving ⬛ pieces in the neck portion of a bottle in the { ⬛, ⬛ } setup

Next, we discuss improper piece placements involving ⬛ pieces in the filling sequence. Here, we first note the improper piece placements within the body portion of the bottle in Figure 34, where (c) is considered improper if there are any empty squares below the ⬛ piece (as the presence of the ⬛-finisher means that the rest of the bottle becomes inaccessible until the finale sequence, thus causing an overflow). In particular, this means that no ⬛ pieces can be placed in the body portions of the bottles, meaning that all ⬛ pieces must be placed in the neck portions. Furthermore, it is impossible to place an ⬛ piece with its rotation center between the rightmost two columns of the bottle, even with line clearing with ⬛s and ⬛s, and placing an ⬛ piece with its rotation center in between the middle two columns of the bottle, like in Figure 33(a) and after Figure 35(i), causes at least one empty square (indicated with a dot) to be inaccessible without causing an overflow. This means that all ⬛ pieces must be placed within the leftmost two columns of the bottle, and by a counting argument, the first two columns must be covered by only ⬛ pieces.

30

Therefore, all the ⬛-piece placements in Figure 35 are improper and cause clogs, as they take up space required for ⬛ pieces, in addition to potentially causing certain squares (marked with dots) to become inaccessible without causing an overflow.
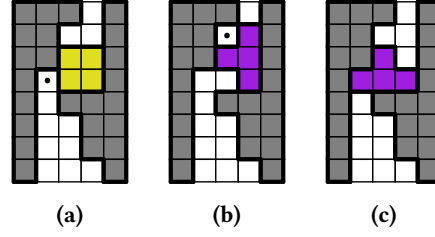
Figure 34: Clogs in the body portion of a bottle in the { ⬛, ⬛ } setup
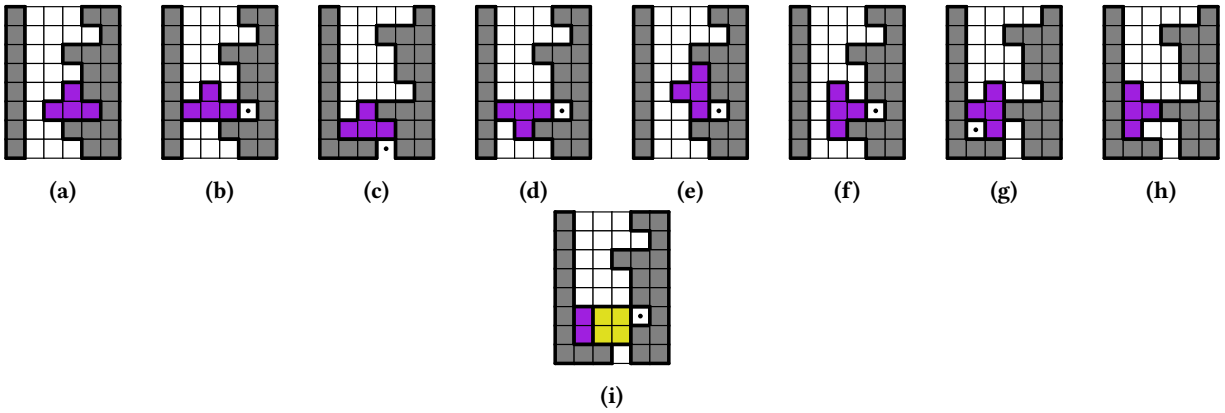
Figure 35: Clogs involving ⬛ pieces in the neck portion of a bottle in the { ⬛, ⬛ } setup

Improper piece placements involving pieces in the closing sequence are similar to the above; for ⬛ pieces, any other placement takes up space required for ⬛ pieces, and for ⬛ pieces, the only other possible placements are those similar to Figure 33(a).

B.4 Clogs for { ⬛, ⬛ }

For the ⬛ pieces in the priming sequence, the only improper piece placement that does not directly overflow the bottle is the one in Figure 36(a); however, after the top line clears, no ⬛ or ⬛ piece can rotate into the rest of the bottle, and the empty square indicated with a dot cannot be filled, without overflowing the bottle.

For the ⬛ pieces in the filling sequence, there are three different types of improper piece placements, as shown in Figure 36(b–d); all of them cause specific empty squares (indicated with dots) to become inaccessible or unable to be filled without overflowing the bottle.

There are no improper piece placements among the pieces in the closing sequence that do not overflow the bottle (except for the first ⬛ piece, which is covered by the discussion on ⬛ pieces in the priming sequence).

B.5 Clogs for { ⬛, ⬛ }

The improper piece placements for ⬛ pieces in the priming and closing sequences are shown in Figure 37; all of them cause at least one empty square (indicated with dots) to become inaccessible or unable to
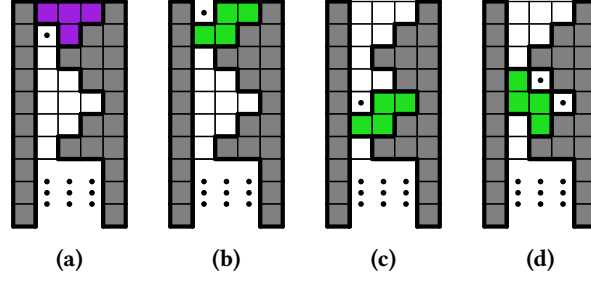
Figure 36: Clogs in the { , } setup

be filled without overflowing the bottle.

Improper piece placements for pieces in the finale sequence cause similar issues with certain empty squares becoming inaccessible or unable to be filled without overflow.
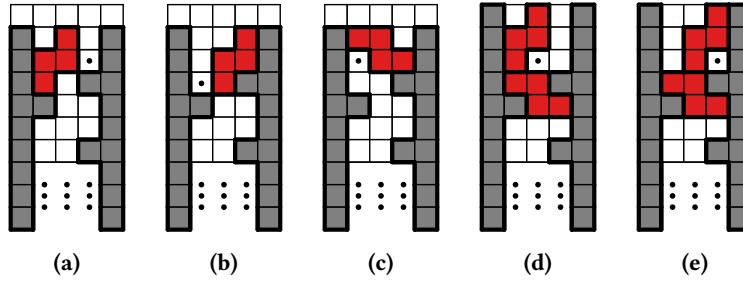
Figure 37: Clogs involving pieces in the { , } setup

The improper piece placements for pieces in the filling sequences are shown in Figure 38. The piece placements in (a) and (b) force a piece to cover the empty square indicated with a dot, which causes at least one square in the rightmost column to become unable to be filled without overflow. The piece placement in (c) causes similar issues involving the empty square indicated with a dot. The piece placement in (d) leads to the other piece placements in (e)-(i), where it becomes impossible to reach the empty squares with dots without overflowing the bottle.

B.6 Clogs for { , }, { , }

The improper piece placements for pieces and pieces in the priming and closing sequences for { , } and { , }, respectively, are shown in Figures 39 and 40; all of them cause at least one empty square (indicated with dots) to become inaccessible or unable to be filled without overflowing the bottle, or cause the rest of the bottle to become inaccessible without overflow due to pieces not being able to move or rotate into the bottle. In particular, putting too many or pieces into a top segment during a priming sequence causes clogs similar to Figure 39(g) and 40(h−i).

The improper piece placements for pieces in the filling and closing sequences are shown in Figure 41. The piece placements in (a)-(e) and (j) cause at least one empty square (indicated with a dot) to become inaccessible without overflowing the bottle. The piece placements in (f)-(i) either prevent more pieces from rotating into the top segment the piece is placed in or allow only one more piece to rotate in and be stuck at the top of the top segment (in the case of (g) and (i)). Due to the structure of a top segment, the pieces in the closing sequence must be placed as in Figure 13; (k)-(l) shows the result of the only improper piece placement involving a piece in the closing sequence that does not directly overflow the bottle, but causes the empty square indicated with a dot to become inaccessible without overflowing

(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)**

(i)
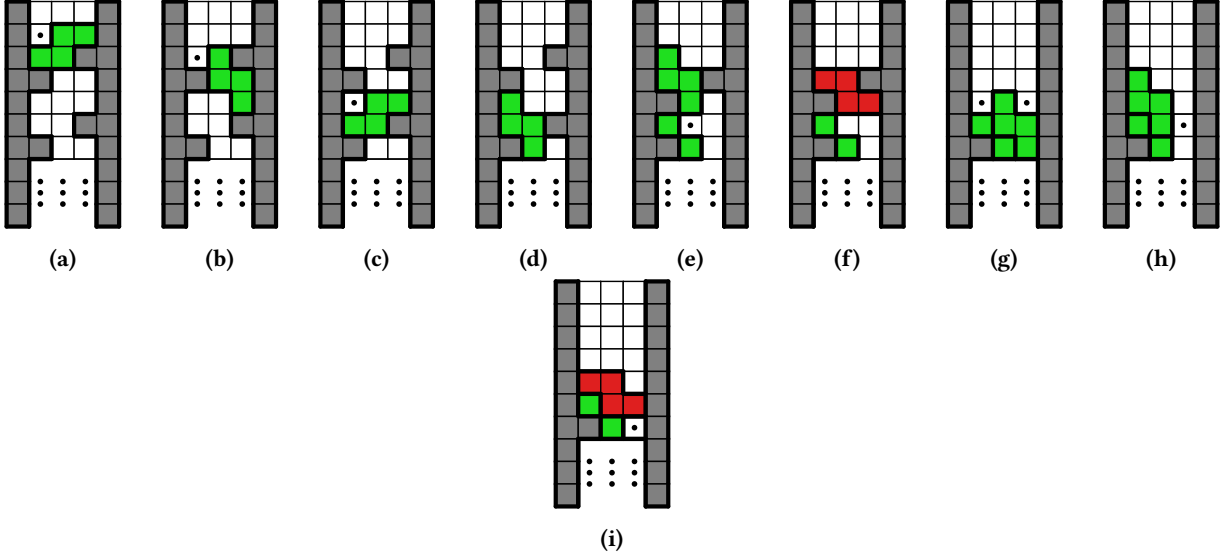
Figure 38: Clogs involving pieces in the { , } setup

(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)**
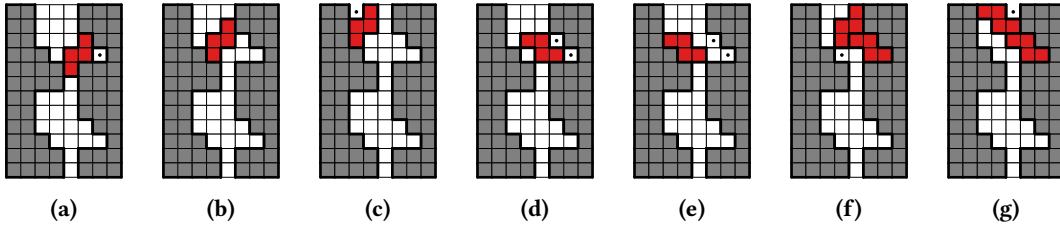
Figure 39: Clogs involving pieces in the { , } setup
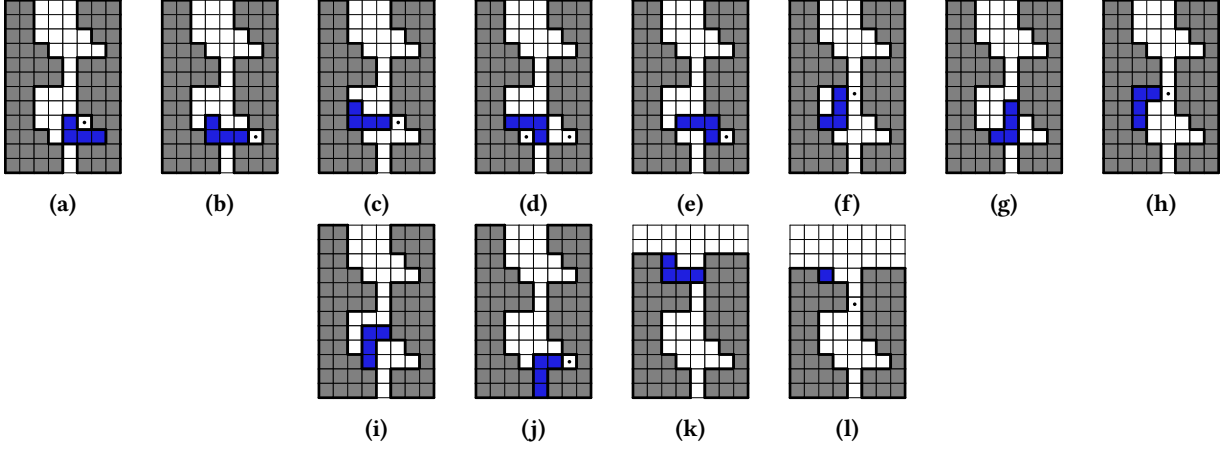

(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)**

(i)
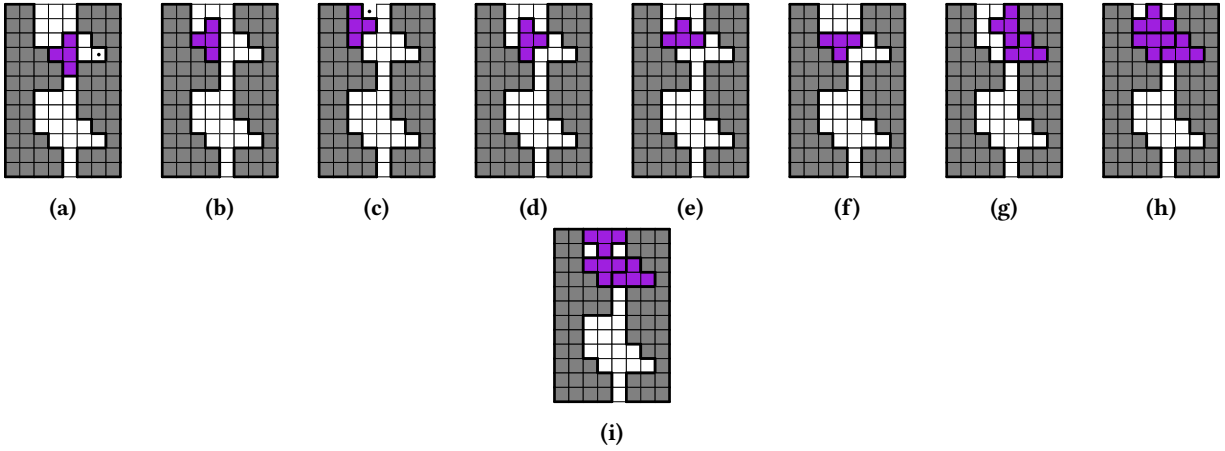
Figure 40: Clogs involving pieces in the { , } setup

the bottle.

B.7 Clogs for { , }

The improper piece placements for pieces in the priming and closing sequences are shown in Figure 42; all of them cause at least one empty square (indicated with dots) to become inaccessible or unable

Figure 41: Clogs involving ▬ pieces in the { ▬, ▬ } and { ▬, ▬ } setups

to be filled without overflowing the bottle, or cause the rest of the bottle to become inaccessible without overflow due to pieces not being able to move or rotate into the bottle, or are impossible under SRS (as in (f)). In particular, putting more than one ▬ piece into a top segment during a priming sequence causes one of the ▬ pieces to be placed in one of the five improper ways shown in Figure 42.
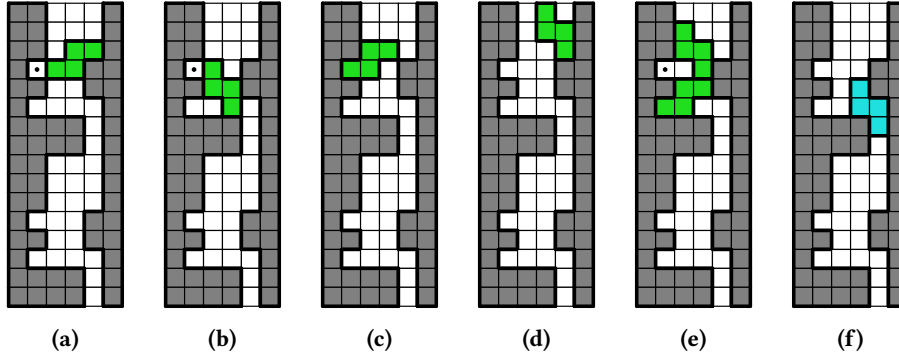
Figure 42: Clogs or impossible piece placements (in the case of (f)) involving ▬ pieces in the { ▬, ▬ } setup

The improper piece placements for ▬ pieces in the filling sequences are shown in Figure 43. The piece placements in (a), (b), (d)-(g), and (i) cause at least one empty square (indicated with a dot) to become inaccessible without overflowing the bottle, either due to being blocked off or due to pieces not being able to rotate in to fill the square. The piece placements in (c), (h), and (j) cause issues with ▬ pieces attempting to rotate into the top segment the ▬ piece is placed in; (k)-(m) show possible piece placements after (j) (with (m) being after the lines above the top segment have been cleared), with certain empty squares (indicated with dots) becoming inaccessible without overflowing the bottle.

Due to the structure of a top segment, the ▬ pieces in the closing sequence must be placed as in Figure 14; improper piece placements for ▬ pieces in the closing sequence here follow a similar analysis as improper piece placements for ▬ pieces in the closing sequence for the { ▬, ▬ } setup or for ▬ pieces in the filling sequence for this setup.

B.8 Clogs for { ▬, ▬ }

The improper piece placements for ▬ pieces in the priming sequences and the first two ▬ pieces in the closing sequences are shown in Figure 44; all of them cause at least one empty square (indicated with
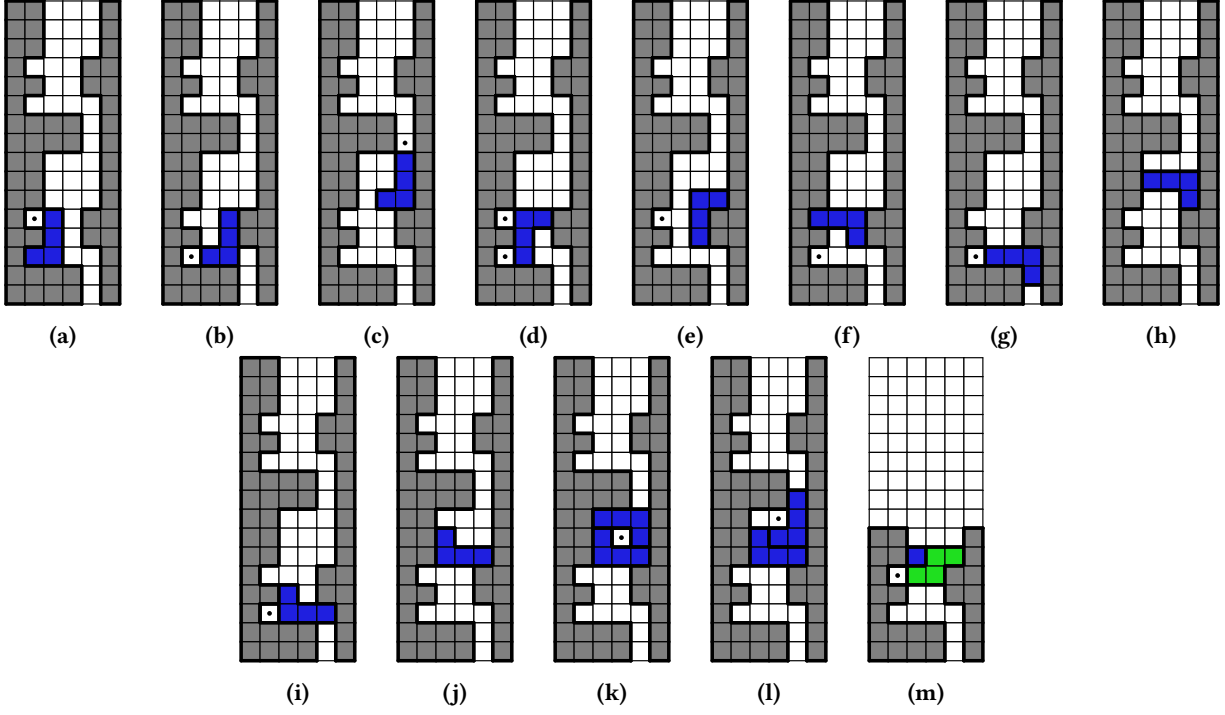
Figure 43: Clogs involving ▦ pieces in the { ▦ , ▦ } setup

dots) to become inaccessible or unable to be filled without overflowing the bottle, or cause the rest of the bottle to become inaccessible without overflow due to pieces not being able to move or rotate into the bottle. In particular, putting more than two ▦ pieces into a top segment during a priming sequence, or having more than two ▦ pieces in a top segment before a ▦ piece gets placed in a closing sequence, causes one of the ▦ pieces to be placed in one of the improper ways shown in Figure 44, particularly in (n) and (o).
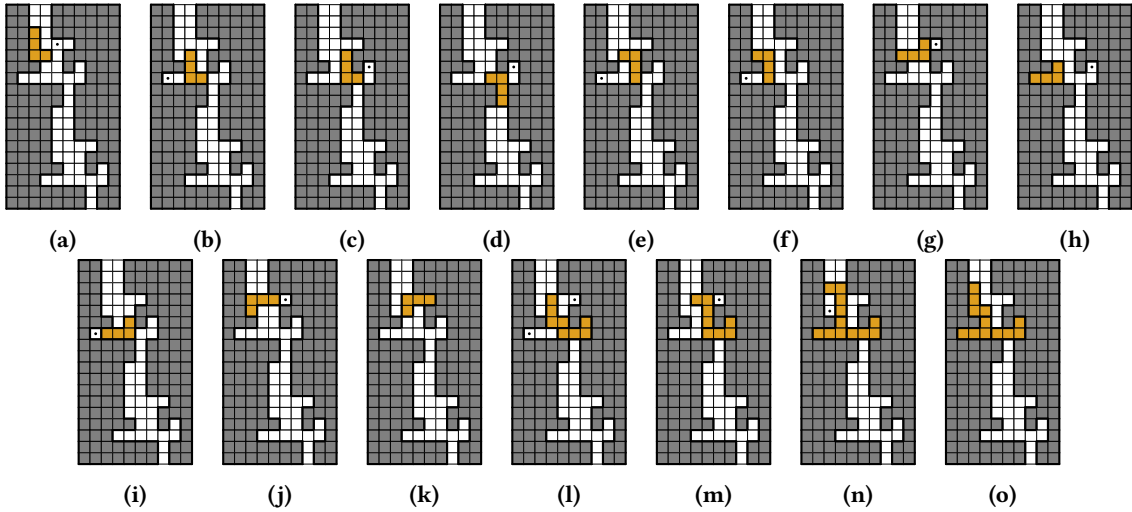
Figure 44: Clogs involving ▦ pieces from priming sequences in the { ▦ , ▦ } setup

The improper piece placements for ▦ pieces in the filling sequences are shown in Figure 45. Most of the piece placements cause at least one empty square (indicated with a dot) to become inaccessible

without overflowing the bottle, either due to being blocked off or due to pieces not being able to rotate in to fill the square or to reach the rest of the bottle (like in (c), (e), and (g)-(i)). The piece placement in (l) prevents any piece from rotating in to cover the empty square with a dot without overflow, as ▨ pieces cannot normally rotate in to cover that empty square and ▨ pieces require one of the squares covered by the ▨ piece to be empty in order to rotate in to cover that square (in addition, it is not possible to clear the lines above the ▨ piece without overflow as the number of empty squares is 2 (mod 4) and all tetrominoes fill 4 squares).
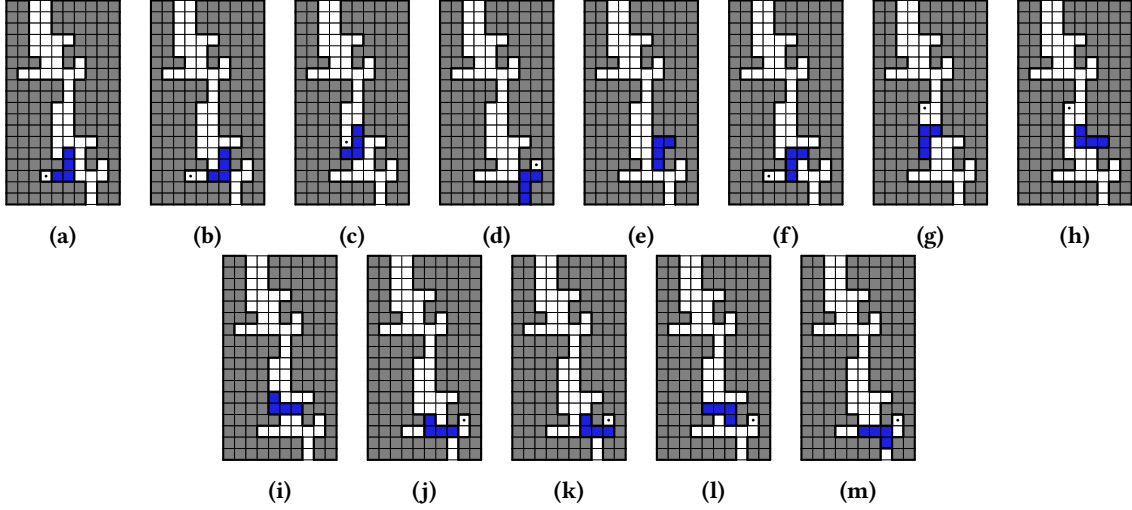
Figure 45: Clogs involving ▨ pieces from filling sequences in the { ▨ , ▨ } setup

The improper piece placements for ▨ pieces and ▨ pieces in the closing sequences are shown in Figure 46; all of them yield scenarios in which it is impossible to re-open the top of the bottle without overflow.
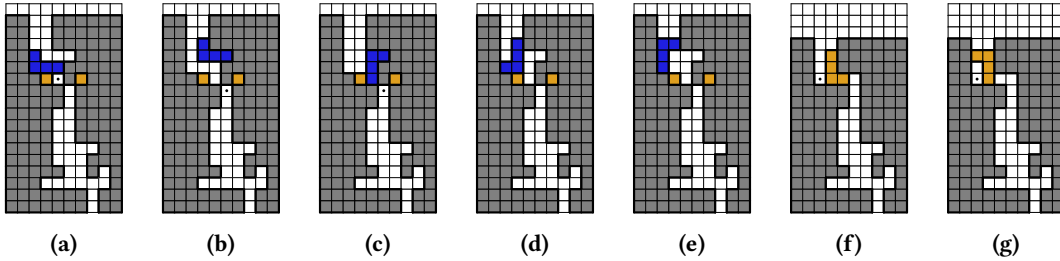

Figure 46: Clogs involving pieces from closing sequences in the { ▨ , ▨ } setup

B.9 Clogs for the ASP-Completeness Setup

The improper piece placements for ▭ pieces are shown in Figure 47. The placements in (a) and (b) block the empty square indicated with a dot. The placement in (c) forces the empty square indicated with a dot to be filled with an ▭ piece, which blocks off other empty squares. The placement in (d) causes the empty square indicated with a dot to become inaccessible, as only ▭ pieces can drop below the hole of size 1 next to the ▭ piece, and ▭ pieces cannot rotate in to fill that empty square.
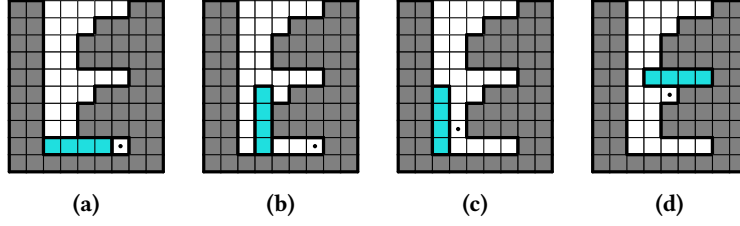
Figure 47: Clogs involving ▭▭▭ pieces in the setup for ASP-completeness of { ▭▭▭, ⬒, ⬛ }

The improper piece placements for ⬒ pieces and ⬛ pieces are shown in Figures 48 and 49; all of them cause an empty square (indicated with a dot) to become inaccessible.
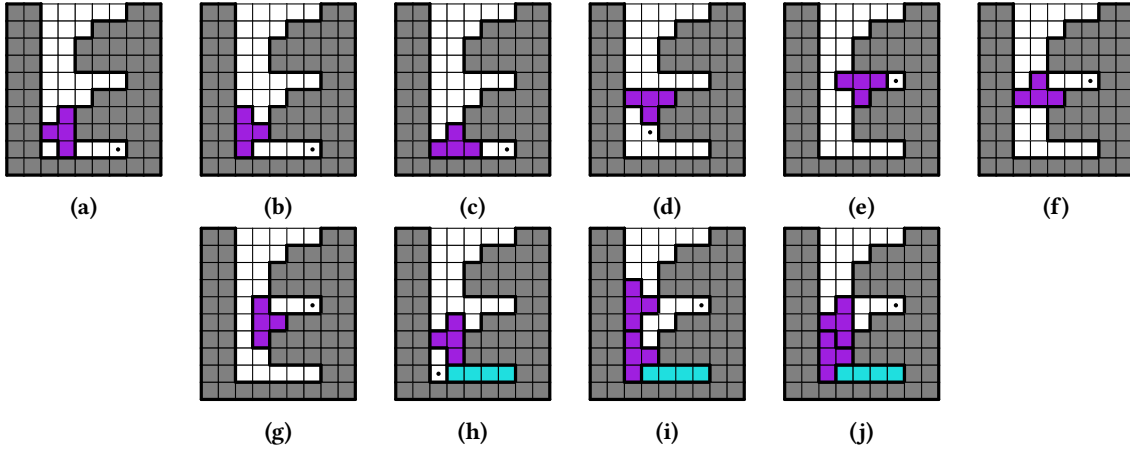

Figure 48: Clogs involving ⬒ pieces in the setup for ASP-completeness of { ▭▭▭, ⬒, ⬛ }
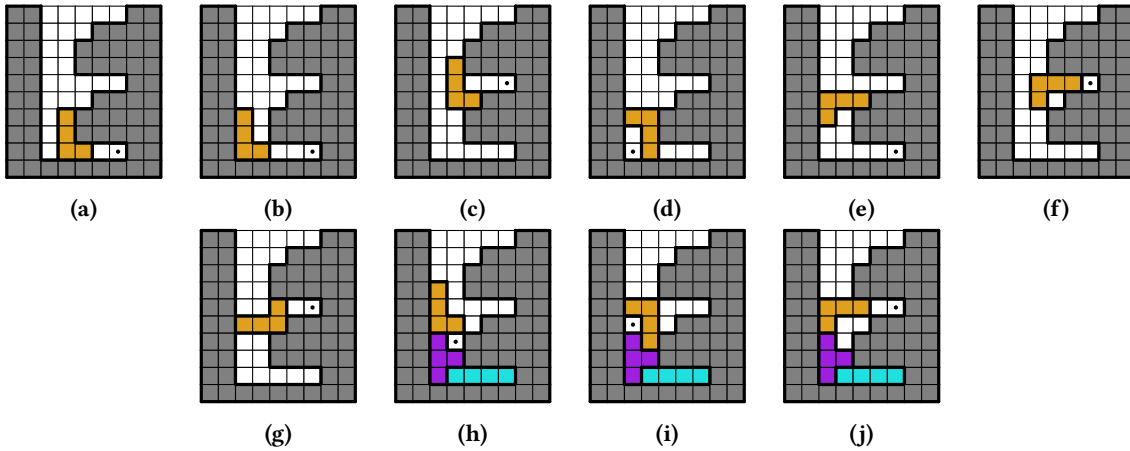
Figure 49: Clogs involving ⬛ pieces in the setup for ASP-completeness of { ▭▭▭, ⬒, ⬛ }