TRANSLATIONAL-INVARIANT TOTAL ORDER: WHEN ORDER TAKES SHAPE

SHAOTIAN SUN

December 9, 2025

1. Introduction

A basic theme in mathematics is the study of orderings on sets, especially those that interact perfectly with the algebraic structure. Our focus on this project is *translation-invariant total orders* (TITOs) on the integers defined by Grant T. Barkley (and Colin Defant)[2, 1].

Definition 1.1. A *translation-invariant total order* (TITO) is a total order $\prec$ of the integers such that for all $a, b \in \mathbb{Z}$, we have $a \prec b$ if and only if $a + n \prec b + n$. The set of TITOs is denoted as $TTot_n$.

TITOs are interesting for several reasons. On the one hand, they provide a rich class of examples of total orders beyond the usual order on $\mathbb{Z}$. On the other hand, they are closely connected to the combinatorics of Coxeter groups and root systems, in particular of affine symmetric groups S_∞which is the affine Coxeter group of type $\tilde{A}_{n-1}$ [4]. In this context, TITOs capture information about weak order, torsion classes, and lattice structures that appear throughout algebraic combinatorics and representation theory.

From a computational perspective, TITOs are challenging to work with. For example, a TITO may have infinitely many inversions, making it non-trivial to compare two TITOs in weak order. Similarly, computing least upper bounds or automating visualizations of TITOs requires efficient algorithms and careful encoding.

The goal of this project is to explore TITOs both mathematically and computationally. Since TITO is a lattice as a weak order as proved in [3], we will implement lattices of TITOs in code, experiment with ways to represent them, and study operations such as checking weak order or computing joins. In addition, we will consider how to visualize TITOs, both for small values of n (e.g. $n = 2, 3$) and in general. Through this, we aim to better understand how these algebraic and combinatorial structures can be concretely realized.

2. Preliminaries

We need to define our context about the weak order on TITOs. Before that, we shall work on the notations about TITOs.

Date: December 9, 2025.

A TITO could be represented as a one-line notation and a window notation. To introduce these notations, blocks of a TITO should be defined first. Note that A subset $I \subseteq \mathbb{Z}$ is *order-convex* (with respect to $(\prec)$) if for any $a, c \in I$ and any $b \in \mathbb{Z}$ such that $a \prec b \prec c$, $b \in I$ holds.

Definition 2.1. Let $(\prec)$ be a TITO. A *block* of $(\prec)$ is an order-convex subset $I \subseteq \mathbb{Z}$ with the following properties:

(1) The ordering of I by $(\prec)$ has no minimal or maximal element.
(2) For any $a, c \in I$, the interval

$$\{\, b \in I \mid a \prec b \prec c \,\}$$

is finite.

The *size* of a block I is the number of residue classes modulo n appearing in I. We say that I is a *waxing* block if $x \prec x+n$ for all $x \in I$. We say that I is a *waning* block if $x+n \prec x$ for all $x \in I$.

One example using both notations is for $n = 2$, the window notation $[0][\underline{1}]$ is the same as $\cdots \prec -2 \prec 0 \prec 2 \prec \cdots \prec 3 \prec 1 \prec -1 \prec \cdots$.

3. Background

Back to defining weak order on TITOs with notations in our hands, we first recall the definition of a poset in combinatorics [5].

Definition 3.1 (Poset)**.** A *poset (or* partial order) is a relation $\prec$ on a set P that is reflective, antisymmetric, and transitive. That is, for all $a, b, c \in P$,

(1) (reflecitivity) $a \prec a$;
(2) (antisymmetry) If $a \prec b$ and $b \prec a$, then $a = b$;
(3) (transitivity) If $a \prec b$ and $b \prec c$, then $a \prec c$.

To define the weak order on TITOs, we need the equivalence relation on pairs (a, b) given by

$$(a, b) \sim (a+n, b+n).$$

The equivalence class containing (a, b) is denoted by (a, b). A *reflection index* is an equivalence class (a, b) with $a < b$.

Definition 3.2. An *inversion* of a TITO $(\prec)$ is a reflection index (a, b) such that $b \prec a$. Let $N(\prec)$ denote the set of inversions of $(\prec)$. We define a partial order on TITOs by setting

$$(\prec_1) \leq (\prec_2) \quad \text{if and only if} \quad N(\prec_1) \subseteq N(\prec_2).$$

Therefore, we establish the context of our weak order on TITOs.

For example, for $n = 2$, $\prec_1 = [0, 1]$, $\prec_2 = \underline{[0, -1]}$, $\prec_3 = [0][1]$ will give their inversion sets as follows:

$$N(\prec_1) = \emptyset;$$
$$N(\prec_2) = \{(0,2), (0,4), \cdots\} \cup \{(1,3), (1,5), \cdots\} \cup$$
$$\{(0,1), (0,3), \cdots\} \cup \{(1,2), (1,4), \cdots\};$$

$$N(\prec_3) = \{(1,2),(1,4),\cdots\}.$$

Then, the weak order on TITOs will be:

$$(\prec_1) \leq (\prec_3) \leq (\prec_2)$$

because $N(\prec_1) \subseteq N(\prec_3) \subseteq N(\prec_2)$.

4. Problem

Since in general, the inversion set could be infinite, it won't be possible for an algorithm to check the weak order between two TITOs by checking inclusion of two (possibly infinite) inversion sets.

Our main focus is to design an algorithm to compare two given TITOs and implement the corresponding computer program. Specifically, given fixed n, we want to compare 2 TITOs', $(\prec_1)$ and $(\prec_2)$'s, order relation with possible outputs: $(\prec_1) \precneqq (\prec_2)$, $(\prec_2) \precneqq (\prec_1)$, $(\prec_2) = (\prec_1)$, and incomparable.

5. Normalization Algorithm (c.r. Owen Wu)

At first, we want to rule out the easiest comparison between two window notations, i.e. they represent the same TITO. Thus, a normalization as follows is proposed:

Algorithm. Fix $n \in \mathbb{Z}_{>0}$. Every TITO T decomposes uniquely into a finite ordered list of *blocks* $B_1, \ldots, B_m$, each block being either *waxing* or *waning*. If a block B has size k, then any consecutive k entries of B in the block order form a *window* of B. From any chosen window, repeated application of the slide operation produces exactly k possible windows of B (shifting by $\pm n$ depending on waxing or waning).

Definition 5.1. (Block normalization) For a block B, its *normalized window* is the lexicographically smallest among its k possible windows, which is denoted as $\mathrm{Norm}(B)$

Definition 5.2. (TITO normalization) For TITO T with block decomposition $B_1, \ldots, B_m$, the *normalized TITO* is

$$\mathrm{Norm}(T) := \big(\mathrm{Norm}(B_1), \ldots, \mathrm{Norm}(B_m)\big),$$

with the blocks in their original order.

Lemma 5.3. *(Well-defined block normalization) Any two windows of the same block B produce the same normalized window* $\mathrm{Norm}(B)$.

Proof. By construction, the set of k possible windows of B is closed under sliding, so any starting window generates the same finite set. The lexicographic minimum of this set is unique, hence $\mathrm{Norm}(B)$ does not depend on the choice of starting window. □

Remark 5.4. Although $\mathbb{Z}^k$ has no lexicographic minimum, each block admits only k possible windows under sliding. Since this is a finite set, the lexicographic minimum exists and is unique.

Lemma 5.5. *(Reconstruction from a normalized window) Given the direction (waxing/waning), n, and a normalized window w of a block B of size k, the entire block B is uniquely determined.*

Proof. See [3, Lemma 4.4]. □

Lemma 5.6. *(Block decomposition is intrinsic.) For a TITO T, the ordered list of blocks $B_1, \ldots, B_m$ is uniquely determined by the total order. Any valid window notation simply chooses one window from each block in this fixed order.*

Proof. Define a graph on $\mathbb{Z}$ by connecting x to y if y is the $\prec_T$-successor or predecessor of x at distance $\pm n$. In a TITO, each element has exactly one such neighbor forward and backward inside its block. The connected components of this graph are exactly the maximal $\pm n$-chains, i.e. the blocks. Hence the set of blocks is determined by $\prec_T$.

To order the blocks left-to-right: take the least element of each block in $\prec_T$ and order the blocks by these. This gives the same ordering as in the original decomposition. □

Theorem 5.7. *(Normalization is canonical and injective.) For TITOs T, T':*

(i) *(*Well-definedness*) Any two window notations of the same T produce the same* $\mathrm{Norm}(T)$.
(ii) *(*Injectivity*) If* $\mathrm{Norm}(T) = \mathrm{Norm}(T')$ *then* $T = T'$.

Proof. (i) Fix T. By Lemma 3, any two window notations select windows from the same block B_r in the same block position. By Lemma 1, both yield the same normalized window $\mathrm{Norm}(B_r)$. Concatenating in the fixed block order gives the same $\mathrm{Norm}(T)$.

(ii) Suppose $\mathrm{Norm}(T) = \mathrm{Norm}(T')$. Then block by block, the normalized windows agree and carry the direction tag. By Lemma 2, each normalized window reconstructs the entire block uniquely. Thus $B_r = B'_r$ for all r, and the block order is preserved. Hence $T = T'$. □

Corollary 5.8. *Two TITOs have the same normalized window notation if and only if they are the same TITO.*

6. Comparison Algorithm

Now that we are given two normalized TITOs, comparing them is our next goal.

To overcome the challenge of the infinity of inversion sets, we observe that the inversion set can be divided by pairs of residue classes. Furthermore, each such pair can be classified into finitely many cases (2 for same residue class pairs; 6 for different residue class pairs). To compare each pair, it suffices to compare those cases, which is manageable using finite steps. After comparing each pair, we combine the results using a logic rule.

The main idea is demonstrated in figure 1.

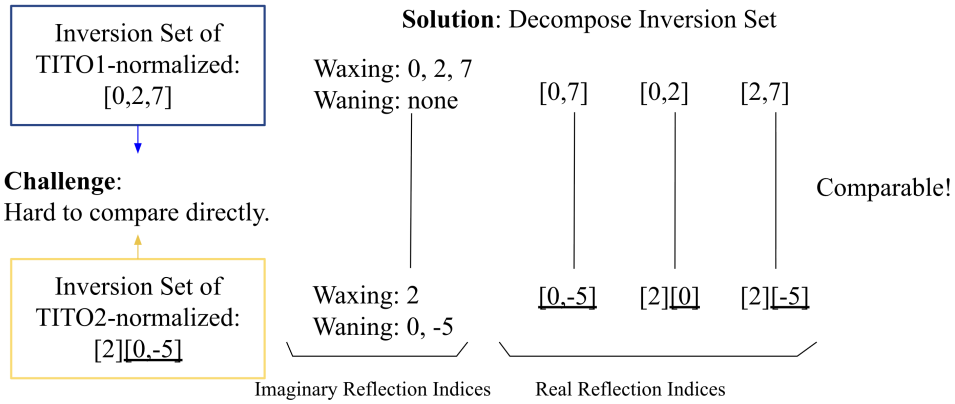


FIGURE 1. Main idea behind the comparison algorithm. Here $n = 3$. (c.r. Emma Lou)

The detailed proposed algorithm of time complexity $O(n^2)$ is as follows:

Algorithm.

Step 1: Inversion Set Decomposition.

Given two normalized TITOs ($\prec_1$ and $\prec_2$), they represent the same TITO if and only if they are identical, i.e. $(\prec_1) = (\prec_2)$.

If they don't fall into the trivial identical case, we consider the reflection indices between two residue classes moding n, which we call them *a residue class pair*. It suffices to consider that because the reflection indices between different residue class pairs are different, making the set order relation impossible to happen between them.

Therefore, we could successfully decompose the inversion set $N(\prec)$ into $\frac{n(n+1)}{2}$ residue class pairs' subsets. Of them, n comes from *same residue class pairs*, and $\frac{n(n-1)}{2}$ comes from *different residue class pairs*.

Step 2: Same Residue Class Pairs Comparison.

When comparing same residue class pairs, the only possibilities for each such pair are all or none. For example, when $n = 3$, we consider $\{(0,3),(0,6),\cdots\}$; $\{(2,5),(2,8),\cdots\}$; or $\{(1,4),(1,7),\cdots\}$. For each such pairs, either all of the indices are reflection indices or none of them are. Thus, we could only consider waxing or waning (underlined or not) for these cases.

Step 3: Different Residue Class Pairs Comparison.

When comparing different residue class pairs, we consider $\{(i,j)|i < j, \text{ and } i < j \pmod n\}$. For example, when $n = 3$, all three residue classes are $\{0,1,2\}$, and all the different residue class pairs are

$$\begin{aligned}
&\{(i,j)|i<j, i\equiv 0 \pmod 3, \text{ and } j\equiv 1 \pmod 3\},\\
&\{(i,j)|i<j, i\equiv 0 \pmod 3, \text{ and } j\equiv 2 \pmod 3\},\\
&\{(i,j)|i<j, i\equiv 1 \pmod 3, \text{ and } j\equiv 2 \pmod 3\}.
\end{aligned}$$

Explicitly, they are

$$\{(0,1),(0,4),(0,7),\cdots\},$$
$$\{(0,2),(0,5),(0,8),\cdots\},$$
$$\{(1,2),(1,5),(1,8),\cdots\}.$$

In the following sub-steps, we will work on the general case with the example pair, i.e. residue class pairs of $i = 0$ and $j = 2$ when $n = 3$, alongside. Also, we will refer $i, j \in \{0, 1, \cdots, n-1\}$ such that $i < j$, and j', j'' some integers that are $j \equiv j' \equiv j'' \pmod{n}$.

Step 3.1: Different Block Cases.

When these two residue classes lie in different blocks, there will be overall two cases $[i][j]$, $[j][i]$ where $i < j \pmod{n}$ if we only consider those two residue classes in each TITO. (Note that the waxing/waning property of each block doesn't affect the resulted inversion set.)

For instance, given that $\prec_1 = [0,1][2]$, $\prec_2 = \underline{[1,5]}[0]$, $\prec_1$ falls into the $[0][2]$ where $0 < 2 \pmod 3$ case; and $\prec_2$ falls into the $[2][0]$ where $0 < 2 \pmod 3$ case.

To conclude, different block cases in different residue class pairs are as follows:

(1) Case 1: (Different block)
$[i][j]$, e.g. $[0][2]$;
(2) Case 2: (Different block)
$[j][i]$, e.g. $[2][0]$.

Step 3.2: Same Block Cases.

However, if these two residue classes lie in the same block, the waning/waxing property of that block will matter. The case classification is more complex in these cases through the following 3 parts:

Step 3.2.1: 2-Block Normalization.

If we only consider those two residue classes in the TITOs, we call it a *2-block*. For each 2-block B, we can apply the Block Normalization defined in 5.1 to it, and get $\mathrm{Norm}(B)$.

For instance, given $\prec_3 = [2,0][1]$, if we extract the residue classes 0 and 2 here, the resulted 2-block is $B = [2,0]$, then $\mathrm{Norm}(B) = [0,5]$. Say $\prec_4 = \underline{[0,1,2]}$, if we extract the residue classes 0 and 2 here, the resulted 2-block is $B = \underline{[0,2]}$, then $\mathrm{Norm}(B) = \underline{[0,2]}$.

As demonstrated in the examples, in general, when we extract residue classes i, j where $i < j \pmod{n}$, the resulted normalized 2-block is

$$\mathrm{Norm}(B) = [i, j'] \text{ or } \underline{[i, j'']}$$

where $j \equiv j' \equiv j'' \pmod{n}$.

Step 3.2.2: Special Cases.

There are two special cases that we have to separate and conclude the comparison result immediately.

(1) Special Case 1: (Usual Order)
$[i, j]$, e.g. $[0, 2]$;
(2) Special Case 2: (Reversing Order)
$[j, i]$, e.g. $[2, 0]$.

If one of the different residue pairs from two TITOs that we are comparing is in Special Case 1 (and the other is not in that case), then its corresponding inversion subset is empty, so it is (strictly) contained in the other. If one is in Special Case 2 (and the other is not), then its corresponding inversion subset is every potential reflection indices, so it (strictly) contains the other.

Step 3.2.3: General Same Block Cases.

Generally, there will be four cases (with special cases separated) considering that the normalized 2-block is waning or waxing, and that j' (or j'') in normalized 2-block is less than or greater than i. They are as follows:

(1) Case 3: (Same block)
$[i, j']$ where $i < j'$, e.g. $[0, 5]$;
(2) Case 4: (Same block)
$[i, j']$ where $i > j'$, e.g. $[0, -1]$.
(3) Case 5: (Same block)
$[i, j'']$ where $i < j''$, e.g. $[0, 2]$;
(4) Case 6: (Same block)
$[i, j'']$ where $i > j''$, e.g. $[0, -4]$.

Step 3.3: Case Comparison.

By the following section, we could directly compare cases here. The special cases are already handled in *Step 3.2.2: Special Cases.* For other cases, it follows from the lemma 6.1 if the different residue class pairs fall into different cases, and from the lemma 6.2 if the different residue class pairs fall into a same case.

Lemma 6.1. *(Different cases comparison rule) Group 1 = union of case 1,4,5. Group 2 = union of case 2,3,6. Cases in each group are comparable, while any cases between two groups are incomparable. Inside each group, the inversion set inclusion are*

$$\textit{Group 1: case } 3 \subset \textit{ case } 2 \subset \textit{ case } 6;$$
$$\textit{Group 2: case } 4 \subset \textit{ case } 1 \subset \textit{ case } 5.$$

Lemma 6.2. *(Same case comparison rule) If two different residue class pairs falls into a same case, we can conclude the set inclusion with respect to which cases that they belong to as below:*

(1) case 1, 2: they are equal;
(2) case 3, 6: larger j' (or j'') is the superset (after checking equality);
(3) case 4, 5: larger j' (or j'') is the subset (after checking equality).

Step 4: Combination Logic Rule.

With results of each residue class pairs (4 possible results: subset($\subset$), superset($\supset$), equal($=$), or incomparable($\times$)), the combination rule is:

The rule of the result is as follows:

(1) If there are any incomparable($\times$) results, the combined result is incomparable;
(2) If there are both ($\subset$) and ($\supset$), the combined result is incomparable;
(3) If there are only ($\subset$)) and ($=$), the combined result is $(\prec_1) \precneqq (\prec_2)$;
(4) If there are only ($\supset$)) and ($=$), the combined result is $(\prec_2) \precneqq (\prec_1)$;
(5) This should conclude all possibilities because the equality condition are already mentioned at the very beginning of the algorithm.

For instance, $\prec_\alpha = [0, 2, 7]$ and $\prec_\beta = [2]\underline{[0, -5]}$. The results of each residue class pairs are shown in the table 1.

$n = 3$	0	1	2
0	$\subset$	$\subset$	$\subset$
1		$\subset$	$\times$
2			$=$

TABLE 1. Results of each residue class pair for $\prec_\alpha$ and $\prec_\beta$.

Based on the combination logic rule, the final result for $\prec_\alpha$ and $\prec_\beta$ is incomparable.

7. Justification of Two Comparison Rules

The non-trivial part for our algorithm will be the proof of lemma 6.1 and 6.2. Both proofs follows directly from the explicit expression of the inversion sets for each cases.

Theorem 7.1. *(Explicit inversion sets for each case)*

Type	*Block Normalized Format*	*Inversion Set*	*Inversion Size*
Case 1	$[i][j]$, $i < j$	$\{(j - kn, i) \mid k \in \mathbb{N}\}$	*Infinite*
Case 2	$[i][j]$, $i > j$	$\{(i, j + kn) \mid k \in \mathbb{N} \cup \{0\}\}$	*Infinite*
Case 3	$[i, j']$, $i < j'$	$\{(i, j + kn) \mid 0 \le k \le \frac{j'-j}{n}, k \in \mathbb{Z}\}$	*Finite*
Case 4	$[i, j']$, $i > j'$	$\{(j - kn, i) \mid 1 \le k \le \frac{j-j'}{n}, k \in \mathbb{Z}\}$	*Finite*
Case 5	$\underline{[i, j'']}$, $i < j''$	*complement of* $[j'', i]$*'s inversion set*	*Cofinite*
Case 6	$\underline{[i, j'']}$, $i > j''$	*complement of* $[j'', i]$*'s inversion set*	*Cofinite*

Take the residue class 0 and 1 when $n = 3$ as an example. The six cases' inversion sets (colored numbers in residue class 1 in the number line paired with 0) are shown in figure 2.

Based on Theorem 7.1, two comparison rules, i.e. Lemma 6.1 and 6.2, are simply corollaries.

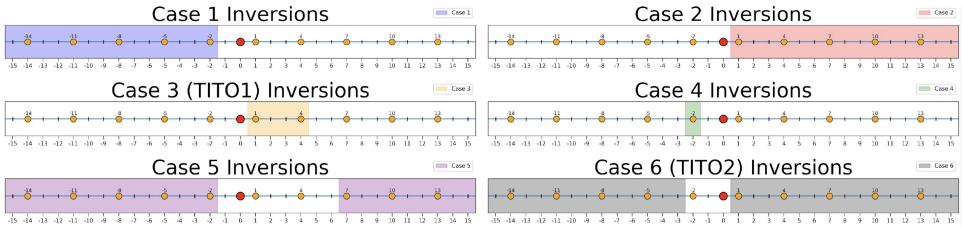
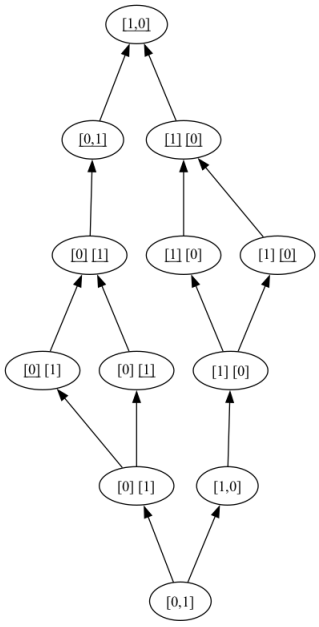


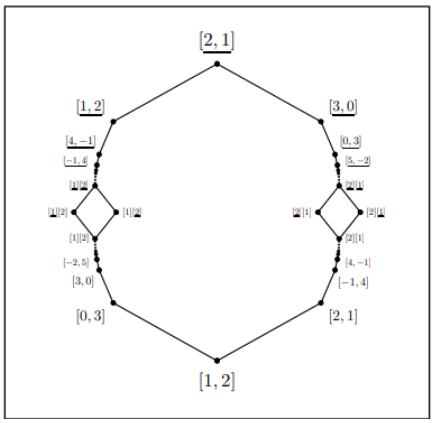
FIGURE 2. Inversions generated by two residue classes.

8. HASSE DIAGRAM

We implement both normalization algorithm and comparison algorithm using Python, and uploaded them here. Moreover, leveraging the comparison algorithm, we obtained the local graph of Hasse Diagram for *small order-ideals* of TITOs when $n = 2$ shown as Figure 3a and $n = 3$ shown as Figure 4.



(A) Diagram generated by our program



(B) Diagram from Barkley [2, Figure 8]

FIGURE 3. Hasse diagrams for $n = 2$ in two representations.

Here, for fixed n, the *small order-ideals* refer to the TITOs whose window notations can be expressed using elements in $\{0, 1, \cdots, n-1\}$.

When $n = 2$, the 2 *Boolean lattices of* $n = 2$, which are of diamond shapes, are clearly presented in both figures in Figure 3. In Figure 4, we highlighted

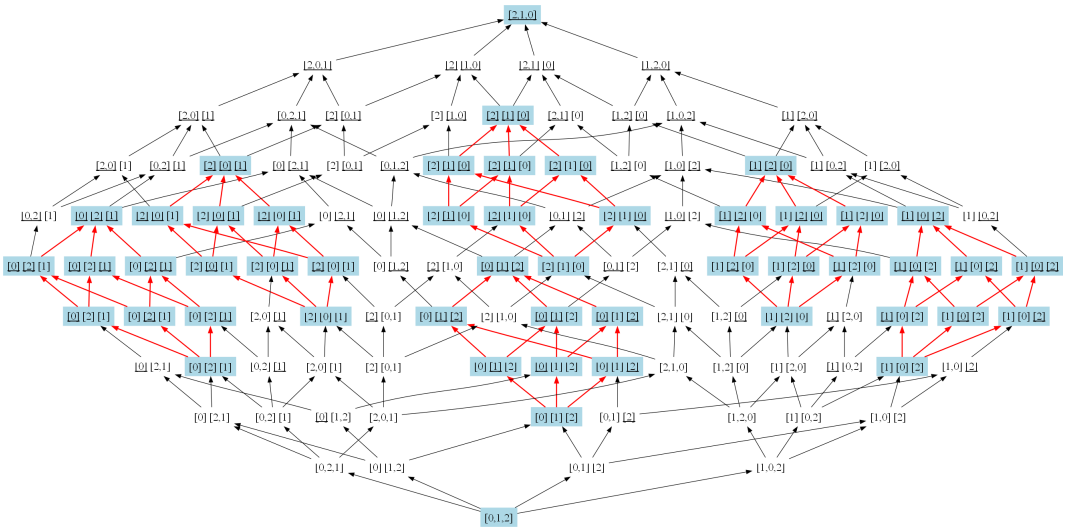


FIGURE 4. Hasse diagram for $n = 3$ with elements $0, 1, 2$.

the 6 *Boolean lattices of* $n = 3$, which are parallel (incomparable) to each other. Those lattices come from the TITOs composed of n blocks.

9. Future Work

In the future, we could complete the algorithm to compute the least upper bound (join) of two TITOs.

Moreover, to explore more thoroughly about the Hasse diagram, we could extend the Hasse diagram to include TITOs beyond *small order-ideals*.

10. Conclusion

To conclude, we have solved the problem and drawn the local Hasse Diagram when n is small. In detail, we have accomplished the following results about weak order on TITOs:

(1) Implementing a complete TITO comparison algorithm;
(2) Correctly identifying all four outputs: $(\prec_1) \precneqq (\prec_2)$, $(\prec_2) \precneqq (\prec_1)$, $(\prec_2) = (\prec_1)$, and incomparable.;
(3) Verifying consistency with theoretical properties;
(4) Generating Hasse diagrams for *small order-ideals* of TITOs.

11. Acknowledgment

I would like to express my sincere gratitude to my teammates, Chi Dinh, Yifan Jing, Emma Lou, and Owen (Yuxuan) Wu, for their thoughtful collaboration and dedication throughout the Fall 2025 semester. I am also deeply appreciative of our graduate student mentor, Mia Smith, and our faculty advisor, Grant Barkley, whose guidance, insight, and encouragement were

invaluable to our work. I am grateful as well to our course instructor, Alejandro Bravo-Doddoli, for his instruction on mathematical research and his helpful advice during the development of our projects.

References

[1] Grant Barkley and Colin Defant. The affine tamari lattice, 2025.
[2] Grant T. Barkley. Extended weak order for the affine symmetric group, 2025.
[3] Grant T. Barkley and David E Speyer. Affine extended weak order is a lattice, 2024.
[4] Grant T. Barkley and David E Speyer. Combinatorial descriptions of biclosed sets in affine type. *Combinatorial Theory*, 4(2), September 2024.
[5] Richard P. Stanley and Sergey Fomin. *Enumerative Combinatorics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.