Journal of Machine Learning Research 24 (2023) 1-61 Submitted 7/18; Revised 11/21; Published 3/23

A Group-Theoretic Approach to Computational Abstraction: Symmetry-Driven Hierarchical Clustering

Haizi Yu HAIZIYU7@ILLINOIS.EDU
Coordinated Science Laboratory
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Igor Mineyev MINEYEV@ILLINOIS.EDU
Department of Mathematics
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Lav R. Varshney VARSHNEY@ILLINOIS.EDU
Coordinated Science Laboratory
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Editor: Risi Kondor

Abstract

Humans' abstraction ability plays a key role in concept learning and knowledge discovery. This theory paper presents the mathematical formulation for computationally emulating human-like abstractions—*computational abstraction*—and abstraction processes developed hierarchically from innate priors like symmetries. We study the nature of abstraction via a group-theoretic approach, formalizing and practically computing abstractions as symmetry-driven hierarchical clustering. Compared to data-driven clustering like k-means or agglomerative clustering (a chain), our abstraction model is *data-free*, *feature-free*, *similarity-free*, and *globally hierarchical (a lattice)*. This paper also serves as a theoretical generalization of several existing works. These include generalizing Shannon's information lattice, specialized algorithms for certain symmetry-induced clusterings, as well as formalizing knowledge discovery applications such as learning music theory from scores and chemistry laws from molecules. We consider computational abstraction as a first step towards a principled and cognitive way of achieving human-level concept learning and knowledge discovery.

Keywords: Computational abstraction, Partition/Clustering, Group, Symmetry, Lattice

1. Introduction

Abstraction refers to the process of generalizing high-level concepts from specific instances by *"ignoring the details"* (Weinberg, 1968; Giunchiglia and Walsh, 1992; Saitta and Zucker, 1998). This conceptual process is pervasive in humans, where more advanced concepts can be abstracted once a "conceptual base" is established (Mandler, 2000). However, it remains mysterious how concepts are abstracted originally, with the source of abstraction generally attributed to innate biology (Mandler, 2000; Gómez and Lakusta, 2004; Biederman, 1987).

Existing data clustering frameworks like k-means and agglomerative clustering touch on the nature of abstraction (Livingston, 1998), but have not achieved the flexibility and interpretability of human-level abstractions (Figure 1). Existing algorithms learning high-level abstractions/representations (Saitta and Zucker, 2013; LeCun et al., 2015; Bredeche et al., 2006; Yu et al., 2016) often require handcrafted priors, typically ad-hoc or hard to interpret (Marcus, 2018; Dietterich, 2018), e.g. rules in automatic reasoning, distributions in Bayesian inference, features in classifiers, or layered neural representations (Raina et al., 2006; Yu et al., 2007; Krupka and Tishby, 2007). Notably, extensive hard coding priors from domain knowledge may be considered "cheating" (Ram and Jones, 1994). This requires us to consider only *innate and universal priors*, i.e. basic and generic priors that can be developed further and become applicable across multiple subject domains.

In this paper, we present a general, mathematical formulation for modeling abstractions computationally, named *computational abstraction*. Via this formulation, we algorithmically emulate human abstraction processes—developed hierarchically from innate priors—to yield human-like and thus human-interpretable abstractions. In particular, we study the nature of abstraction via a group-theoretic approach, using *partitions of a domain* (i.e. clustering a data space rather than a data set) as the core formulation and using *symmetries* (groups in mathematics) as the priors. Our abstraction framework is universal in two senses: first, we consider the general question of conceptualizing a domain—a task-free preparation phase before specific problem solving (Zucker, 2003); second, symmetries in nature are universal priors from so-called Core Knowledge representing human innate cognition (Spelke and Kinzler, 2007). The ultimate goal is to learn domain knowledge by further coupling computational abstraction with statistical learning. This is closely related to but also differs from prior work at the intersection of group theory and machine/deep learning (Kondor, 2008; Romero and Lohit, 2021; Dehmamy et al., 2021; Basu et al., 2021).

We summarize our contributions and the scope of this paper as follows:

- a mathematical formulation for computational abstraction (Section 3);
- a top-down approach (Section 4) based on the identification theorem (Theorem 20);
- a bottom-up approach (Section 5) based on the duality theorems (Theorems 6, 9);
- an algorithmic generalization (Section 6) of an algorithm customized for special groups (Yu et al., 2021);
- a theoretical generalization (Section 7.1) of Shannon (1953)'s information lattice;
- a theoretical formalism (Section 7.2) of knowledge discovery methods and applications (Yu and Varshney, 2017; Yu et al., 2020).

We consider the computational abstraction in this paper to be a first step towards a principled, cognitive way of achieving human-level concept learning and knowledge discovery; this latter goal is beyond the scope here and is presented in separate papers.

1.1 Theoretical Formulation for Abstraction: A Motivating Example

Existing formulations of abstraction include *mathematical logic* with abstractions explicitly constructed from abstraction operators and formal languages (Saitta and Zucker, 2013; Zucker, 2003; Bundy et al., 1990) and *deep learning* with abstractions hinted at by the layered neural representations (LeCun et al., 2015; Bengio, 2009). The former is rule-based and interpretable, but requires crafting complicated logic from massive domain expertise; the latter replaces model crafting by data, but makes the model less transparent.

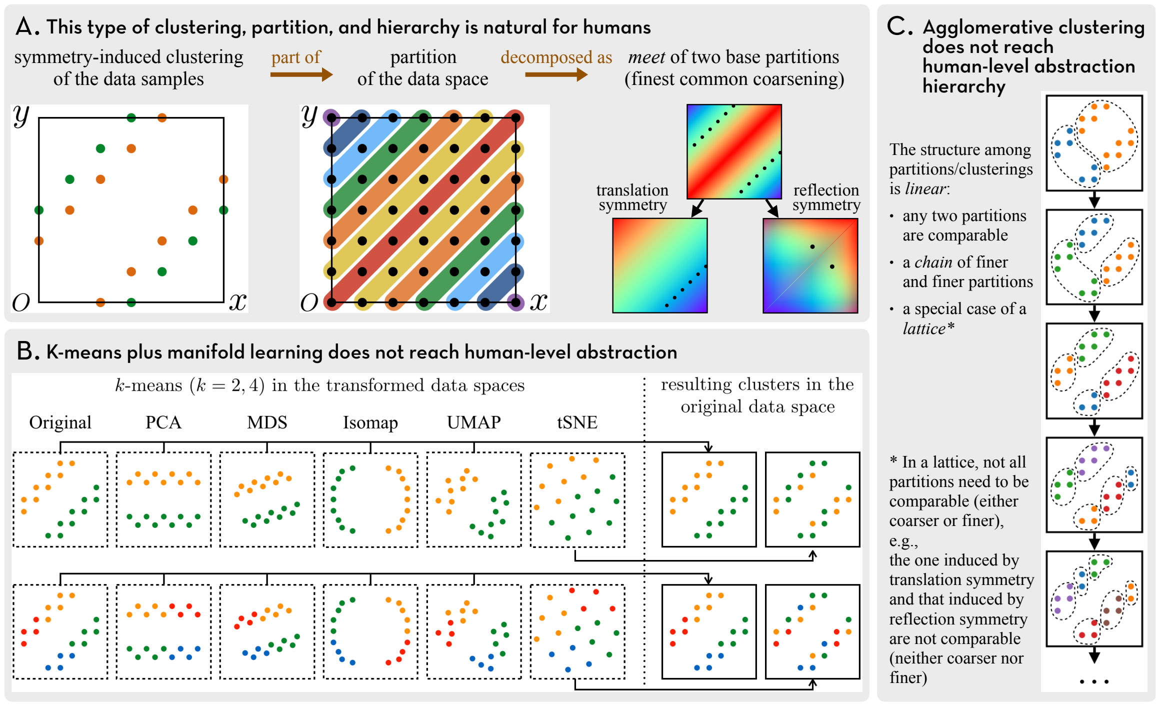

A. This type of clustering, partition, and hierarchy is natural for humans
symmetry-induced clustering of the data samples
part of
partition of the data space
decomposed as
meet of two base partitions (finest common coarsening)
translation symmetry
reflection symmetry
B. K-means plus manifold learning does not reach human-level abstraction
k-means ($k = 2, 4$) in the transformed data spaces
resulting clusters in the original data space
Original
PCA
MDS
Isomap
UMAP
tSNE
C. Agglomerative clustering does not reach human-level abstraction hierarchy
The structure among partitions/clusterings is *linear*:
any two partitions are comparable
a *chain* of finer and finer partitions
a special case of a *lattice**
* In a lattice, not all partitions need to be comparable (either coarser or finer), e.g., the one induced by translation symmetry and that induced by reflection symmetry are not comparable (neither coarser nor finer)

Figure 1: Clustering captures the nature of abstraction, whereas data clustering (e.g. k-means, hierarchical clustering) plus manifold learning (e.g. PCA, Isomap) lack the flexibility and interpretability of human-level abstractions.

We aim for a middle ground between the two and take a new, clustering-based viewpoint. By means of *clustering*—which is a more general, philosophical concept than data clustering in machine learning like k-means—we ignore within-cluster variations and recognize between-cluster distinctions only (Bélai and Jaoua, 1998; Sheikhalishahi et al., 2016), viewing the nature of abstraction as equivalence relations (Livingston, 1998). We formalize abstraction as *symmetry-driven clustering*, which complements data clustering in four ways.

1. **Data-free.** Our clustering model partitions an input space rather than data samples. It is viewed more as conceptual clustering than data clustering like k-means (Michalski and Stepp, 1983; Fisher, 1987): clusters are formed in a *mechanism-driven* (not data-driven) fashion with our considered mechanisms being *symmetries*. The process is causal and the results are interpretable. One *clustering mechanism* (one symmetry) transfers to multiple domains; one *clustering result* transfers to various datasets.

2. **Feature-free.** Our clustering model involves no feature engineering, so no domain expertise. There is no feature filtering or normalization typically used for preprocessing raw inputs. There is no attribute functions used for representing ad-hoc concepts e.g. hand-designed abstraction operators (Zucker, 2003), arithmetic descriptors (Yu et al., 2016), or feature thresholding as in decision trees (Sheikhalishahi et al., 2016). There is also no meta-feature tuning e.g. pre-specifying the number of clusters.

3. **Similarity-free.** Our clustering model is not tied to a predefined similarity or metric. This differs from most clustering methods whose essential work is defining "closeness"

(Raman and Varshney, 2019; Rand, 1971). Instead of crafting similarities, we consider *equivalence relations*, induced from symmetries. Notably, defining certain symmetries (e.g. isometries) may require a metric space whose metric however, is not necessarily used as inverse similarity. Hence, points far apart in a metric space may be perceived together, yielding a "disconnected cluster" comprising disconnected regions in the input space. This is not likely to happen for algorithms such as k-means.

4. **Global hierarchy.** Like many hierarchical clusterings (Jain and Dubes, 1988; Rokach and Maimon, 2005), our clustering model outputs multiple partitions and their hierarchy. Yet here, we formalize *global* hierarchies by *lattices*, generalizing greedy hierarchical clusterings such as agglomerative/divisive clustering (Cormack, 1971; Kaufman and Rousseeuw, 2009) or topological clustering via persistent homology (Oudot, 2015). The greedy algorithms only produce a chain—a linear order and a special case of a lattice—since they produce partitions by local merges/splits made in a one-directional procedure, e.g. growing a dendrogram or a filtration. The chain structure, rather than the more general lattice structure (i.e. a directed acyclic graph), is oft-criticized since it is hard to recover from bad clusterings in early stages of construction (Oudot, 2015).

Being feature-free and similarity-free makes a clustering model *universal* (Raman and Varshney, 2019), which becomes more of a *science* than an *art* (Von Luxburg et al., 2012). In Figure 1, we pictorially exemplify why k-means or hierarchical clustering is far from human-level abstractions that are more flexible in clustering and more complex in hierarchy. So, in this paper, we generalize current clustering methods in a fundamental way.

1.2 Algorithmic Principles for Abstraction

To go from our mathematical formulation to computational methods for abstraction, we introduce two general principles—a top-down and a bottom-up approach—to algorithmically generate hierarchical abstractions from hierarchical symmetries enumerated systematically. The two principles leverage different theorems derived from our formulation and lead to practical algorithms that emulate human abstraction processes.

1. **A top-down approach.** We propose a principled pipeline to traverse from general symmetries to more specific ones. In particular, we present an identification theorem for an explicit and complete parameterization of the whole family of affine symmetries. This theorem allows the decomposition of a large symmetry-enumeration problem into smaller enumeration subproblems, which applies to other generic symmetry families. This approach from general symmetries to more specific ones corresponds to top-down paths in the symmetry hierarchy.
2. **A bottom-up approach.** Starting from a set of base symmetries (the seeds), we generate all symmetries that are seeded from the given set. Based on the duality theorems derived from our formulation, we introduce an induction algorithm for computing hierarchical abstractions without explicitly enumerating the corresponding symmetries. Our induction algorithm recursively computes higher-level abstractions from low-level ones and is more efficient than generating all abstractions from scratch (i.e. from symmetries). It quickly generates abstraction families from arbitrary and tentative seeds, and one can further fine tune the seeds as needed. This approach from base symmetries to higher-level ones corresponds to bottom-up paths in the symmetry hierarchy.

1.3 A Theoretical Formalism of and Generalization to Existing Works

Computing abstractions as orbits under group actions is a classic problem in computational group theory (Holt et al., 2005). However, generic algorithms are restricted to finite groups or finite action spaces, whereas most symmetries (e.g. 2D translations) are infinite groups acting on infinite spaces. There are also efficient but specialized algorithms customized for special types of infinite groups, e.g. crystallographic groups (Eick and Souvignier, 2006) and atomically generated subgroups of semidirect products (Yu et al., 2021). Our computational abstraction framework yields a generic algorithm for any finitely generated subgroups (possibly infinite) and produces the hierarchy in the same pass of orbit computations.

Shannon's lattice theory of information was first brought about in communication problems where *information elements and lattices* were introduced to denote the nature and hierarchy of information, respectively (Shannon, 1953). This work remains relatively underexplored compared to Shannon's other works on information theory. Our theoretical work on computational abstraction allows us to reformulate and generalize Shannon's information lattice by coupling abstraction lattice with statistics, which further enables statistical learning in the lattice as already adopted in applications shown below.

Practical knowledge discovery methods have shown success in learning music theory from scores and chemistry laws from molecules (Yu and Varshney, 2017; Yu et al., 2020). By projecting and lifting data statistics along the downward and upward directions of partition lattices, one can recover 70% of a standard music theory curriculum from the sheet music of 370 Bach's chorales as well as compound composition and nomenclature laws from chemical formulas. This paper presents the theory behind these applications.

The main technical problem in this paper is: given an input space and a class of symmetries, compute symmetry-driven abstractions of the input space and their hierarchy. To motivate, to formalize, and to solve the problem, this paper is organized as follows. Section 2 describes intuitions on abstractions. Section 3 starts the mathematical formulation for computational abstraction, formalizing symmetry and abstraction and casting their hierarchies in a primal-dual viewpoint. Sections 4 and 5 introduce the top-down and bottom-up approaches, with the given class of symmetries taking two forms: all subgroups of a group (up to isomorphism) and subgroups generated by all subsets of a generating set. Section 6 gives implementation and a general setting where symmetries are arbitrary and abstractions are for arbitrary finite subspaces of a possibly infinite input space. Section 6.3 discusses how the general setting generalizes the algorithm specially designed for atomically generated subgroups. Section 7 discusses connections to Shannon's information lattice—a special case under our abstraction formalism—and connections to existing knowledge discovery applications that use computational abstraction as the backend theory in lattice learning.

2. Abstraction: Informal Description

We informally discuss *abstraction* to provide initial intuitions. To do so, we consider toy examples and examples from everyday life to illustrate abstraction in specific contexts. Nevertheless, the intuitions here cover everything about abstraction that we will generalize, formalize, and further operationalize (computationally) in the following sections. Therefore, this section, despite its informality, establishes an agenda for the remainder of the paper.

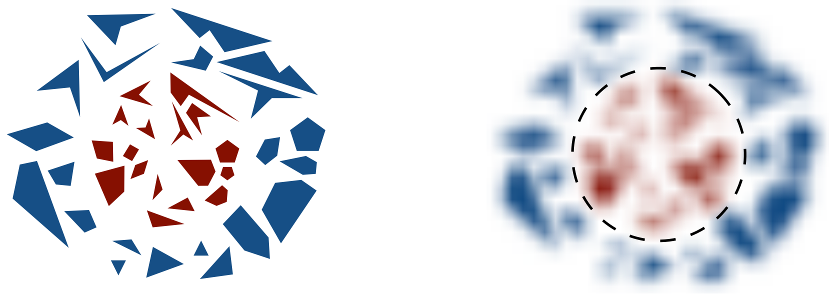


Figure 2: By clustering, an abstraction of polygons presents a coarse-grained view of the given polygons, where we ignore all shape-related information and only discern their colors. This results in two clusters: {red polygons, blue polygons}.

2.1 Nature of Abstraction

Abstraction is everywhere in our daily life (even though we may not realize it all the time). Examples of people making abstractions can be as simple as observing ourselves through social categories such as race or gender (Macrae and Bodenhausen, 2000); or as complicated as a systematic taxonomy of a subject domain, such as modern taxonomy of animals and Western classification of music chords. While the contexts might change, there are common intuitions for viewing the nature of abstraction.

Abstraction as partition (clustering). One way to view an abstraction is through a *partition*, or *clustering*, and then *ignoring* within-cluster variations. For instance, we cluster people into {men, women}, ignoring the difference between John and David, Mary and Rachel; we cluster vertebrates into {fish, amphibians, reptiles, birds, mammals}, ignoring the difference between penguins and eagles, dogs and bats; we cluster music triads into {major, minor, augmented, diminished, ...}, ignoring the difference between CEG and FAC, CE♭G and ACE.

Abstraction as equivalence relation. Another (equivalent) way to view an abstraction is through an *equivalence relation*, where the earlier idea of "clustering and ignoring" is reinterpreted as identifying things (in the same cluster) indistinguishably. For instance, identifying John and David (as men); identifying dogs and bats (as mammals); identifying CEG and FAC (as major triads). This view is consistent with mathematics, since a partition of a set is basically the same thing as an equivalence relation on that set. This view is also consistent with psychology, stating the nature of abstraction is to treat instances as if they were qualitatively identical, although in fact they are not (Livingston, 1998).

Classification is more than needed. This idea of clustering is common in many definitions of abstraction, but more often termed as classification (or categorization, taxonomy). While clustering and classification (likewise clusters and classes) may be synonyms in everyday life, they are different in machine learning. The former falls under unsupervised learning, whereas the latter falls under supervised learning. The difference lies in whether or not there is a label for each cluster. Labels are important in supervised learning, since a perfect binary classifier with a 100% accuracy is clearly different from a bad one with a 0% accuracy. Yet in light of clustering, the two classifiers are identical: the "bad" one,

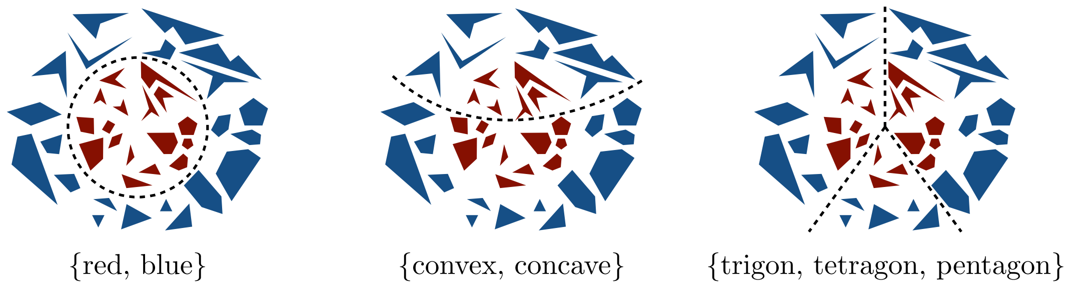


Figure 3: Different and incomparable abstractions of the same set of polygons.

for instance, may call black as white and call white as black, but still accurately captures the concept of binary colors. So, to make this abstraction, all we need are two clusters of colored objects; yet, labeling the two colors by black and white is not essential.

In summary, treating an abstraction as clustering (or a partition, an equivalence relation) presents a *coarse-grained view* (Figure 2) of the observed instances, where we deliberately ignore within-cluster variations by collapsing equivalent instances into one cluster and discerning only between-cluster variations. The intuitions here are formalized in Section 3.1.

2.2 Abstraction Hierarchy

More than one abstraction exists for the same set of instances, simply because there are many different ways of partitioning the set and there are many different ways of identifying things to be equivalent. For instance, considering {bats, dogs, eagles}, we can identify bats and dogs indistinguishably as mammals, but distinguish them from eagles which are birds; we can also identify bats and eagles indistinguishably as flying animals, but now distinguish them from dogs since dogs cannot fly. Intuitively, how different abstractions compare with one another induces the notion of a hierarchy.

Linear hierarchy: total order. Perhaps the simplest is a linear hierarchy, where "later" abstractions are made from their immediate "precursors" by continuously merging clusters. For instance, we cluster animals into {fish, birds, mammals, annelids, mollusks, ...}, and further cluster these abstracted terms into {vertebrates, invertebrates}. A linear hierarchy is essentially a totally (a.k.a. linearly) ordered set; it has a clear notion of *abstraction level*, e.g. in animal taxonomy: species $\rightarrow$ genus $\rightarrow$ family $\rightarrow$ order $\rightarrow$ class $\rightarrow$ phylum $\rightarrow$ kingdom. Notably, the famous agglomerative hierarchical clustering is an example of linear hierarchy, producing a linear sequence of coarser and coarser partitions.

More than linear: partial order. Nevertheless, an abstraction hierarchy can be more complicated than simply linear due to various clustering possibilities that are incomparable. For instance, take the same set of polygons from Figure 2. Besides color, consider two other ways of partitioning the same set (Figure 3). These three abstractions are made from three incomparable criteria (color, convexity, number of sides), and the resulting partitions are also incomparable (none of them is made from merging clusters from either of the others). Hence, instead of a linear hierarchy, a family of abstractions forms a partially ordered set

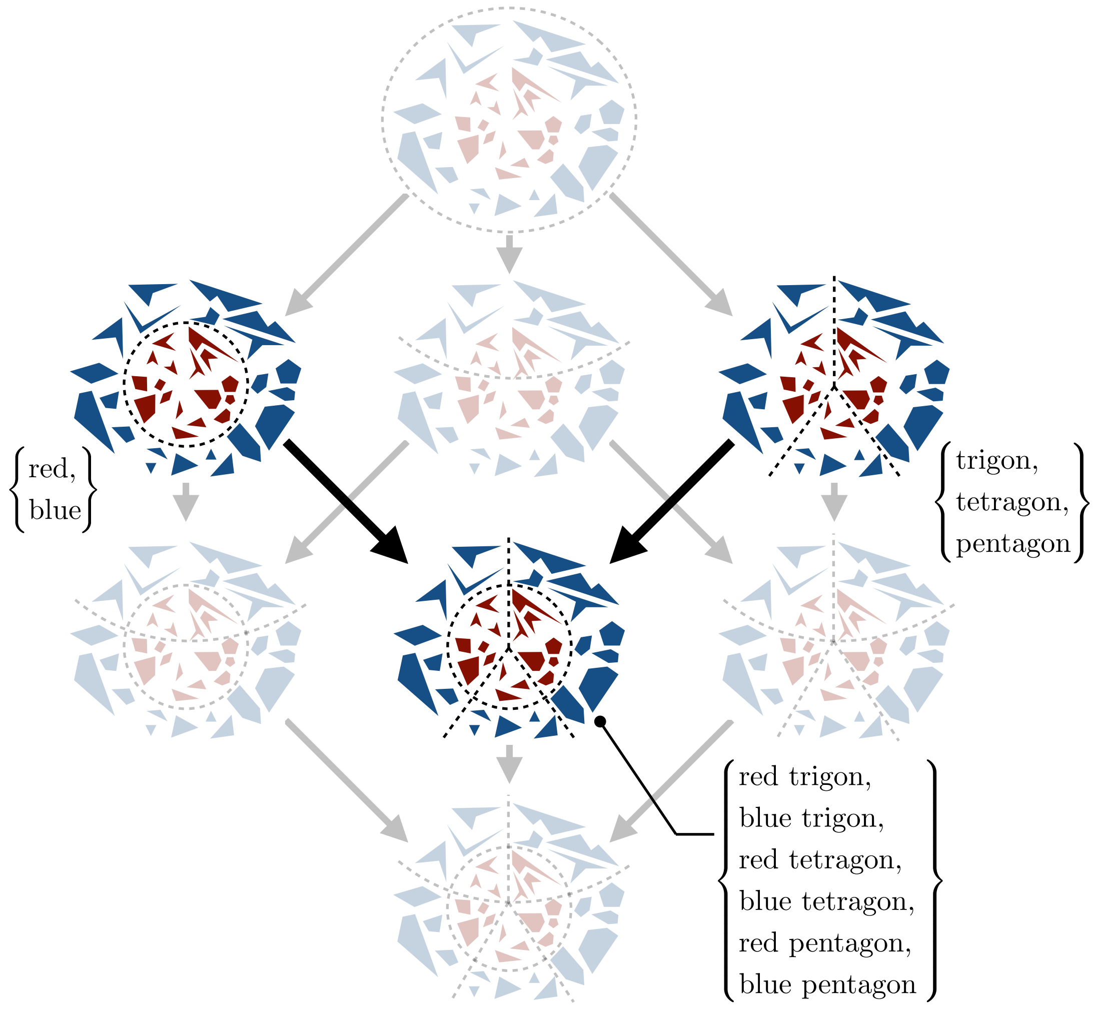


Figure 4: Two abstractions and their coarsest common refinement (foreground) as part of a global lattice structure (foreground & background)—a special type of partially ordered set. This whole lattice is pictured as a directed acyclic graph, where every vertex is a distinct abstraction of the given polygons and every edge $\mathcal{A} \to \mathcal{A}'$ denotes $\mathcal{A}$ is coarser than $\mathcal{A}'$.

in general, and the notion of abstraction level is only relative due to incomparability. This is one way in which this paper extends traditional (linear) hierarchical clustering.

More than partial order: lattice. Given any two abstractions as two coarse-grained views of the same set, there is a clear notion of their so-called *finest common coarsening* and oppositely, their *coarsest common refinement*. We exemplify the latter in the foreground of Figure 4, where a color-based clustering and a shape-based clustering uniquely determines a new clustering based on both color and shape. The initial two abstractions of polygons, together with their coarsest common refinement, are part of a bigger hierarchy (Figure 4), which is a special type of partially ordered set called a *lattice*. Intuitions developed thus far about abstraction hierarchy motivate their formal descriptions in Section 3.2.

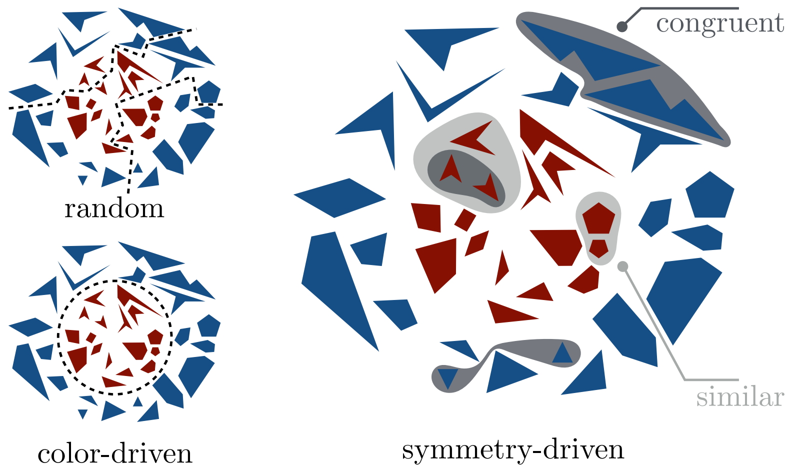


Figure 5: Various kinds of abstraction mechanisms.

2.3 Abstraction Mechanisms

In many cases, an abstraction is made from an explicable reason—a *mechanism* that drives the resulting abstraction. For instance, color is the underlying mechanism that drives the abstraction in Figure 2. An abstraction mechanism also naturally serves as a *prior* with respect to its induced clustering process. In this paper, we focus on abstractions made from clear mechanisms (rather than random clustering), and furthermore, from mechanisms that can be used as universal priors for multiple subject domains (Figure 5).

Mechanism-driven. In three ways, we prefer mechanism-driven abstractions to random abstractions. First, for the sake of *explainability*, the recognition of a mechanism allows us to understand the abstraction (e.g. from {men, women}, we recognize gender is under consideration). Second, for the sake of *generalizability*, the presence of a mechanism may allow us to transfer the same mechanism to a different set (e.g. from {men, women} to {roosters, hens}, {bulls, cows}). Third, for the sake of *computability*, restriction to a certain category of mechanisms allows us to practically generate all abstractions under the chosen category rather than to aimlessly consider all possible partitions (of a set), whose number grows faster than exponential (with respect to the size of the set).

Symmetry-driven. Mechanisms are of various kinds. In data-driven abstractions (i.e. data clustering), a mechanism is a pre-defined notion of data proximity. In category-driven abstractions (e.g. taxonomy of animals), a mechanism is usually a defining attribute of the instances themselves (ontology). Hence, while an attribute like number of sides is a valid mechanism for abstracting polygons, it is not so when trying to abstract animals. Rather than these two kinds of mechanisms which are often ad hoc and domain-specific, we study symmetry-driven abstractions, drawing general symmetries from nature. Again, take the polygons in Figure 2 as an example: we can cluster all congruent polygons together to get an abstraction admitting translation and rotation invariances; we can cluster all similar polygons together to get a coarser abstraction admitting an additional scaling invariance (Figure 5). These intuitions about being symmetry-driven are formalized in Section 3.3.

3. Abstraction: Mathematical Formalism

We formalize an abstraction process on an underlying space as a clustering problem. In this process, elements of the space are grouped into clusters, abstracting away within-cluster variations. The outcome is a coarse-grained abstraction space whose elements are the clusters. Clustering is performed based on certain symmetries such that the resulting clusters are invariant with respect to the symmetries.

3.1 Abstraction as Partition (Clustering)

We formalize an *abstraction* of a set as a partition of the set, which is a mathematical representation of the outcome of a clustering process. Throughout this paper, we reserve X to exclusively denote a set which we make abstractions of. The set X can be as intangible as a mathematical space, e.g. $\mathbb{R}^n$, $\mathbb{Z}^n$, a general manifold; or as concrete as a collection of items, e.g. {rat, ox, tiger, rabbit, dragon, snake, horse, sheep, monkey, rooster, dog, pig}.

Preliminaries (Appendix A.1): partition of a set ($\mathcal{P}$), partition cell ($P \in \mathcal{P}$); equivalence relation on a set ($\sim$), quotient ($X/\sim$).

An abstraction is a partition, and vice versa. The two terms refer to the same thing, except that one is used less formally whereas the other is used in the mathematical language. When used as a single noun, these two terms are interchangeable in this paper.

A partition is not an equivalence relation. The two terms do not refer to the same thing (one is a set, the other is a binary relation), but convey equivalent ideas since they induce each other bijectively (Appendix A.1). In this paper, we use an equivalence relation to explain a partition: elements of a set X are put in the same cell because they are equivalent. Based on this reason, abstracting the set X is about treating equivalent elements as the same, i.e. collapsing equivalent elements in X into a single entity (namely, an equivalence class or a cell) where collapsing is formalized by taking the quotient.

3.2 Abstraction Universe as Partition Lattice (Hierarchical Clustering)

A set X can have multiple partitions for $|X| > 1$. The number of all partitions of X is called the *Bell number* $B_{|X|}$, which grows extremely fast with the size of the set: the first few Bell numbers are $1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, \ldots$ We use $\mathfrak{P}^*_X$ to denote the family of all partitions of a set X, so $|\mathfrak{P}^*_X| = B_{|X|}$. We can compare partitions of a set in two ways. One simple way is to compare by size: given two partitions $\mathcal{P}, \mathcal{Q}$ of a set, we say that $\mathcal{P}$ is no larger than (resp. no smaller than) $\mathcal{Q}$ if $|\mathcal{P}| \leq |\mathcal{Q}|$ (resp. $|\mathcal{P}| \geq |\mathcal{Q}|$). Another way of comparison considers the structure of partitions via a *partial order* on $\mathfrak{P}^*_X$. The partial order further yields a *partition lattice*, a hierarchical representation of a family of partitions.

Preliminaries (Appendix A.2): partial order, poset; lattice, join ($\vee$), meet ($\wedge$), sublattice, join-semilattice, meet-semilattice, bounded lattice.

Definition 1 *Let $\mathcal{P}$ and $\mathcal{Q}$ be two abstractions of a set X. We say that $\mathcal{P}$ is at a higher level than $\mathcal{Q}$, denoted $\mathcal{P} \preceq \mathcal{Q}$, if as partitions, $\mathcal{P}$ is coarser than $\mathcal{Q}$. For ease of description, we expand the vocabulary for this definition, so the following are all equivalent:*

1. *$\mathcal{P} \preceq \mathcal{Q}$, or equivalently $\mathcal{Q} \succeq \mathcal{P}$ (Figure 6).*
2. *As abstractions, $\mathcal{P}$ is at a higher level than $\mathcal{Q}$ (or $\mathcal{P}$ is an abstraction of $\mathcal{Q}$).*
3. *As partitions, $\mathcal{P}$ is coarser than $\mathcal{Q}$ (or $\mathcal{P}$ is a coarsening of $\mathcal{Q}$).*
4. *As abstractions, $\mathcal{Q}$ is at a lower level than $\mathcal{P}$ (or $\mathcal{Q}$ is a realization of $\mathcal{P}$).*
5. *As partitions, $\mathcal{Q}$ is finer than $\mathcal{P}$ (or $\mathcal{Q}$ is a refinement of $\mathcal{P}$).*
6. *Any $x, x' \in X$ in the same cell in $\mathcal{Q}$ are also in the same cell in $\mathcal{P}$.*
7. *Any $x, x' \in X$ in different cells in $\mathcal{P}$ are also in different cells in $\mathcal{Q}$.*

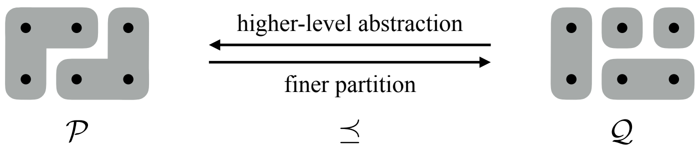


Figure 6: The partial order $\preceq$ compares the levels of abstractions.

It is known that the binary relation "coarser than" on the family $\mathfrak{P}_X^*$ of all partitions of a set X is a partial order, so is the binary relation "at a higher level than" on abstractions. Given two partitions $\mathcal{P}, \mathcal{Q}$ of a set, we can have $\mathcal{P} \preceq \mathcal{Q}$, $\mathcal{Q} \preceq \mathcal{P}$, or they are incomparable. Further, $(\mathfrak{P}_X^*, \preceq)$ is a bounded lattice, in which the greatest element is the finest partition $\{\{x\} \mid x \in X\}$ and the least element is the coarsest partition $\{X\}$. For any pair of partitions $\mathcal{P}, \mathcal{Q} \in \mathfrak{P}_X^*$, their join $\mathcal{P} \vee \mathcal{Q}$ is the coarsest common refinement of $\mathcal{P}$ and $\mathcal{Q}$; their meet $\mathcal{P} \wedge \mathcal{Q}$ is the finest common coarsening of $\mathcal{P}$ and $\mathcal{Q}$ (Figure 7).

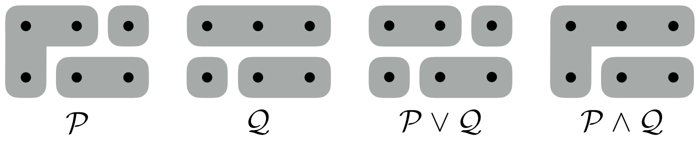

$\mathcal{P}$
$\mathcal{Q}$
$\mathcal{P} \vee \mathcal{Q}$
$\mathcal{P} \wedge \mathcal{Q}$

Figure 7: Two abstractions $\mathcal{P}, \mathcal{Q}$ and their join $\mathcal{P} \vee \mathcal{Q}$ and meet $\mathcal{P} \wedge \mathcal{Q}$.

Definition 2 *An abstraction universe for a set X is a sublattice of $\mathfrak{P}_X^*$, or a partition (sub)lattice in short. In particular, we call the partition lattice $\mathfrak{P}_X^*$ itself the complete abstraction universe for X. An abstraction join-semiuniverse (resp. meet-semiuniverse) for a set X is a join-semilattice (resp. meet-semilattice) of $\mathfrak{P}_X^*$. An abstraction family for a set X, an even weaker notion, is simply a subset of $\mathfrak{P}_X^*$.*

If the complete abstraction universe $(\mathfrak{P}_X^*, \preceq)$ is finite, we can visualize its hierarchy as a directed acyclic graph where vertices denote partitions and edges denote the partial order. The graph is constructed as follows: plot all distinct partitions of X starting at the bottom with the finest partition $\{\{x\} \mid x \in X\}$, ending at the top with the coarsest partition $\{X\}$ and, roughly speaking, with coarser partitions positioned higher than finer ones. Draw edges downwards between partitions using the rule that there will be an edge downward from $\mathcal{P}$ to $\mathcal{Q}$ if $\mathcal{P} \preceq \mathcal{Q}$ and there does not exist a third partition $\mathcal{R}$ such that $\mathcal{P} \preceq \mathcal{R} \preceq \mathcal{Q}$. Thus, if $\mathcal{P} \preceq \mathcal{Q}$, there is a path (possibly many paths) downward from $\mathcal{P}$ to

$\mathcal{Q}$ passing through a chain of intermediate partitions (and a path upward from $\mathcal{Q}$ to $\mathcal{P}$ if $\mathcal{Q} \succeq \mathcal{P}$). For any pair of partitions $\mathcal{P}, \mathcal{Q} \in \mathfrak{P}_X^*$, the join $\mathcal{P} \vee \mathcal{Q}$ can be read from the graph as follows: trace paths downwards from $\mathcal{P}$ and $\mathcal{Q}$ respectively until a common partition $\mathcal{R}$ is reached (note that the finest partition $\{\{x\} \mid x \in X\}$ at the bottom is always the end of all downward paths in the graph, so it is guaranteed that $\mathcal{R}$ always exists). To ensure that $\mathcal{R} = \mathcal{P} \vee \mathcal{Q}$, make sure there is no $\mathcal{R}' \preceq \mathcal{R}$ (indicated by an upward path from $\mathcal{R}$ to $\mathcal{R}'$) with upward paths towards both $\mathcal{P}$ and $\mathcal{Q}$ (otherwise replace $\mathcal{R}$ with $\mathcal{R}'$ and repeat the process). Symmetrically, one can read the meet $\mathcal{P} \wedge \mathcal{Q}$ from the graph.

There are limitations to this process, especially if the set X is infinite. Even for a finite set X of relatively small size, the complete abstraction universe $\mathfrak{P}_X^*$ can be quite complicated to visualize (recall that we have to draw $|\mathfrak{P}_X^*| = B_{|X|}$ vertices where $B_{|X|}$ grows extremely fast with $|X|$, let alone the edges). However, not all arbitrary partitions are of interest to us. In the following subsections, we study symmetry-generated abstractions and abstraction universes. So, later we can focus on certain partitions by considering certain symmetries.

3.3 Symmetry-Generated Abstraction

Recall that we explain an abstraction of a set by its inducing equivalence relation, where equivalent elements are treated as the same. Instead of considering arbitrary equivalence relations or arbitrary partitions, we construct every abstraction from an explicit mechanism—a symmetry—so the resulting equivalence classes or partition cells are invariant under this symmetry. To capture various symmetries, we consider groups and group actions.

Preliminaries (Appendix A.3): group ($(G, *)$ or G), subgroup ($\leq$), trivial subgroup ($\{e\}$), subgroup generated by a set ($\langle S \rangle$), cyclic subgroup ($\langle s \rangle$); group action, G-action on X ($\cdot : G \times X \to X$), orbit of $x \in X$ (Gx), set of all orbits (X/G).

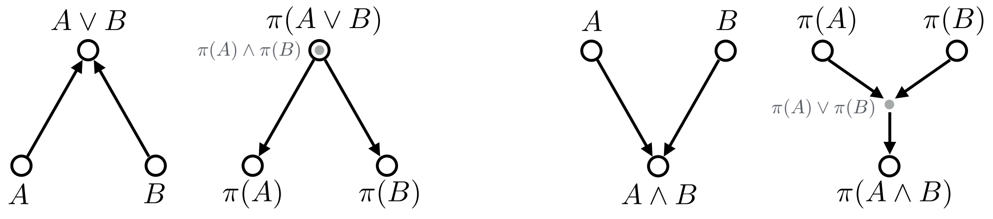
Consider a special type of group, namely the symmetric group $(S_X, \circ)$ defined over a set X, whose group elements are all the bijections from X to X and whose group operation is (function) composition. The identity element of S_X is the identity function, denoted id. A bijection from X to X is also called a *transformation* of X. Therefore, the symmetric group S_X comprises all transformations of X, and is also called the transformation group of X, denoted $\mathsf{F}(X)$. We use these two terms and notations interchangeably in this paper, with a preference for $\mathsf{F}(X)$ in general, while reserving S_X mostly for a finite X.

Given a set X and a subgroup $H \leq \mathsf{F}(X)$, we define an H-action on X by $h \cdot x := h(x)$ for any $h \in H, x \in X$; the *orbit* of $x \in X$ under H is the set $Hx := \{h(x) \mid h \in H\}$. Orbits in X under H define an equivalence relation: $x \sim x'$ if and only if x, x' are in the same orbit, and each orbit is an equivalence class. Thus, the quotient $X/H = X/{\sim}$ is a partition of X. It is known that every cell (or orbit) in the abstraction (or quotient) X/H is a minimal non-empty invariant subset of X under transformations in H. Therefore, we say this abstraction respects the so-called *H-symmetry* or *H-invariance*.

We succinctly record the above process of constructing an abstraction X/H (of X) from a given subgroup $H \leq \mathsf{F}(X)$ in the following *abstraction generating chain*:

$$\text{a subgroup of } \mathsf{F}(X) \xrightarrow{\textit{group action}} \text{orbits} \xrightarrow{\textit{equiv. rel.}} \text{a partition} \xrightarrow{\textit{is}} \text{an abstraction of } X,$$

which can be further encapsulated by the *abstraction generating function* defined as follows.



(a) From join to meet. (b) From meet to join.

Figure 8: Duality of join and meet between the subgroup lattice (left in each subfigure) and the partition lattice (right in each subfigure). In (a), the gray vertex denoting $\pi(A) \wedge \pi(B)$, i.e. the actual meet in the partition lattice, is equal to $\pi(A \vee B)$; in (b), the gray vertex denoting $\pi(A) \vee \pi(B)$, i.e. the actual join in the partition lattice, can be any vertex below $\pi(A), \pi(B)$ and above $\pi(A \wedge B)$ or even equal to these three end points.

Definition 3 *The abstraction generating function is the mapping $\pi : \mathcal{H}^*_{\mathsf{F}(X)} \to \mathfrak{P}^*_X$ where $\mathcal{H}^*_{\mathsf{F}(X)}$ is the collection of all subgroups of $\mathsf{F}(X)$, $\mathfrak{P}^*_X$ is the family of all partitions of X, and for any $H \in \mathcal{H}^*_{\mathsf{F}(X)}$, $\pi(H) := X/H := \{Hx \mid x \in X\}$, where $Hx := \{h(x) \mid h \in H\}$.*

Theorem 4 *π is not necessarily injective. (Proof in Appendix B.1.)*

Theorem 5 *π is surjective. (Proof in Appendix B.2)*

3.4 Duality: from Subgroup Lattice to Abstraction (Semi)Universe

Given a subgroup of $\mathsf{F}(X)$, we can generate an abstraction of X via π. Given a collection of subgroups of $\mathsf{F}(X)$, we can generate a family of abstractions of X. Further, given a collection of subgroups of $\mathsf{F}(X)$ with a hierarchy, we can generate a family of abstractions of X with an induced hierarchy. This leads us to a *subgroup lattice* generating a partition (semi)lattice; the latter is dual to the former via the abstraction generating function π.

Preliminaries (Appendix A.4): the (complete) subgroup lattice for a group $(\mathcal{H}^*_G, \leq)$, join ($A \vee B = \langle A \cup B \rangle$), meet ($A \wedge B = A \cap B$).

We consider the subgroup lattice for $\mathsf{F}(X)$, denoted $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$. Similar to the complete abstraction universe $(\mathfrak{P}^*_X, \preceq)$, we can draw a directed acyclic graph to visualize $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$ if it is finite, where vertices denote subgroups and edges denote the partial order. The graph is similarly constructed by plotting all distinct subgroups of $\mathsf{F}(X)$ starting at the bottom with $\{\text{id}\}$, ending at the top with $\mathsf{F}(X)$ and, roughly speaking, with larger subgroups positioned higher than smaller ones. Draw an upward edge from A to B if $A \leq B$ and there are no subgroups properly between A and B. For any pair of subgroups $A, B \in \mathcal{H}^*_{\mathsf{F}(X)}$, the join $A \vee B$ can be read from the graph by tracing paths upwards from A and B respectively until a common subgroup containing both is reached, and making sure there are no smaller such subgroups; the meet $A \wedge B$ can be read from the graph in a symmetric manner. For any subgroup $C \in \mathcal{H}^*_{\mathsf{F}(X)}$, the subgroup sublattice $(\mathcal{H}^*_C, \leq)$ for C is part of the subgroup lattice $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$ for $\mathsf{F}(X)$, which can be read from the graph for $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$ by extracting the part below C and above $\{\text{id}\}$.

Theorem 6 (Duality) *Let $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$ be the subgroup lattice for $\mathsf{F}(X)$, and π be the abstraction generating function. Then $(\pi(\mathcal{H}^*_{\mathsf{F}(X)}), \preceq)$ is an abstraction meet-semiuniverse for X. More specifically, for any $A, B \in \mathcal{H}^*_{\mathsf{F}(X)}$, the following hold (proof in Appendix B.3):*

1. *partial-order reversal: if $A \leq B$, then $\pi(A) \succeq \pi(B)$;*
2. *strong duality: $\pi(A \vee B) = \pi(A) \wedge \pi(B)$ (Figure 8a);*
3. *weak duality: $\pi(A \wedge B) \succeq \pi(A) \vee \pi(B)$ (Figure 8b).*

Remark 7 (Practical implication) *The strong duality in Theorem 6 suggests a quick way of computing abstractions. If one has already computed abstractions $\pi(A)$ and $\pi(B)$, then instead of computing $\pi(A \vee B)$ from $A \vee B$, one can compute the meet $\pi(A) \wedge \pi(B)$, which is less expensive than computing $A \vee B$ and identifying all orbits in $\pi(A \vee B)$.*

Theorem 6 further allows us to build an abstraction semiuniverse with a partial hierarchy directly inherited from the hierarchy of the subgroup lattice. Nevertheless, there are cases where $\pi(A) \preceq \pi(B)$ with incomparable A and B since the abstraction generating function π is not injective (Theorem 4). If desired, one needs additional steps to complete the hierarchy.

3.5 More on Duality: from Conjugation to Group Action

Partitions of a set X generated from two conjugate subgroups of $\mathsf{F}(X)$ can be related by a group action. We present this relation as another duality between subgroups and abstractions, which can also simplify the computation of abstractions.

Preliminaries (Appendix A.5): conjugate, conjugacy class.

Theorem 8 *Let G be a group, X be a set, and $\cdot : G \times X \to X$ be a G-action on X. Then*

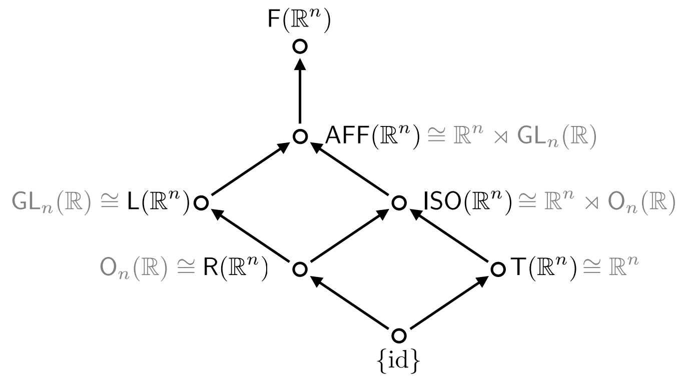
1. *for any $g \in G, Y \in 2^X$, $g \cdot Y := \{g \cdot y \mid y \in Y\} \in 2^X$, and the corresponding function $\cdot : G \times 2^X \to 2^X$ defined by $g \cdot Y$ is a G-action on 2^X;*
2. *for any $g \in G, \mathcal{P} \in \mathfrak{P}^*_X$, $g \cdot \mathcal{P} := \{g \cdot P \mid P \in \mathcal{P}\} \in \mathfrak{P}^*_X$, and the corresponding function $\cdot : G \times \mathfrak{P}^*_X \to \mathfrak{P}^*_X$ defined by $g \cdot \mathcal{P}$ is a G-action on $\mathfrak{P}^*_X$. (Proof in Appendix B.4.)*

Theorem 9 (Duality) *Let X be a set, $\mathsf{F}(X)$ be the transformation group of X, and π be the abstraction generating function. Then for any $H \leq \mathsf{F}(X)$ and $g \in \mathsf{F}(X)$,*

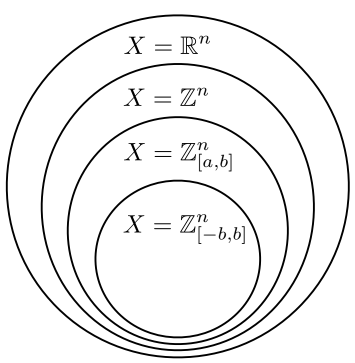
$$\pi(g \circ H \circ g^{-1}) = g \cdot \pi(H),$$

where $\cdot$ is the group action defined in Statement 2 in Theorem 8. (Proof in Appendix B.5.)

Remark 10 (Practical implication) *Theorem 9 relates conjugation in the subgroup lattice $\mathcal{H}^*_{\mathsf{F}(X)}$ to group action on the partition lattice $\mathfrak{P}^*_X$. In other words, the group action on the partition lattice is dual to the conjugation in the subgroup lattice. This duality suggests a quick way of computing abstractions. If one has already computed abstraction $\pi(H)$, then instead of computing $\pi(g \circ H \circ g^{-1})$ from $g \circ H \circ g^{-1}$, one can compute $g \cdot \pi(H)$, which is generally a less expensive operation than computing $g \circ H \circ g^{-1}$ and identifying all orbits in $\pi(g \circ H \circ g^{-1})$.*


$\mathsf{F}(\mathbb{R}^n)$
$\mathsf{AFF}(\mathbb{R}^n) \cong \mathbb{R}^n \rtimes \mathsf{GL}_n(\mathbb{R})$
$\mathsf{GL}_n(\mathbb{R}) \cong \mathsf{L}(\mathbb{R}^n)$
$\mathsf{ISO}(\mathbb{R}^n) \cong \mathbb{R}^n \rtimes \mathsf{O}_n(\mathbb{R})$
$\mathsf{O}_n(\mathbb{R}) \cong \mathsf{R}(\mathbb{R}^n)$
$\mathsf{T}(\mathbb{R}^n) \cong \mathbb{R}^n$
$\{\mathrm{id}\}$

(a) Subgroups under consideration.



(b) Spaces under consideration.

Figure 9: Special subgroups and spaces as well as their hierarchies. (a) presents a backbone of the complete subgroup lattice $\mathcal{H}^*_{\mathsf{F}(\mathbb{R}^n)}$, including important subgroups and their breakdowns. One can check the above directed acyclic graph indeed represents a sublattice: it is closed under both join and meet. (b) presents important subspaces of $\mathbb{R}^n$, where restrictions are gradually added to eventually lead to practical abstraction-construction algorithms.

Theoretically, through the abstraction generating function π and hierarchy completion, the complete abstraction universe $\mathfrak{P}^*_X$ can be built from the complete subgroup lattice $\mathcal{H}^*_{\mathsf{F}(X)}$. This is because the subgroup lattice is a larger space that "embeds" the partition lattice (Theorems 4 and 5). However, it is impractical to store $\mathfrak{P}^*_X$, and not all arbitrary partitions of X are equally useful. Instead of all subgroups of $\mathsf{F}(X)$, we constrain our attention to certain parts of the complete subgroup lattice $\mathcal{H}^*_{\mathsf{F}(X)}$. To do so, we introduce two algorithmic principles: the top-down approach and the bottom-up approach.

4. The Top-Down Approach: Special Subgroups

We consider the sublattice $(\mathcal{H}^*_G, \leq)$ for some subgroup $G \leq \mathsf{F}(X)$, i.e. the part below G and above $\{\mathrm{id}\}$ in the directed acyclic graph for the complete subgroup lattice $(\mathcal{H}^*_{\mathsf{F}(X)}, \leq)$. The top-down approach first specifies a "top" in $\mathcal{H}^*_{\mathsf{F}(X)}$ (i.e. a subgroup $G \leq \mathsf{F}(X)$) and then extracts everything below the "top" (i.e. the subgroup lattice $\mathcal{H}^*_G$). The computer algebra system GAP (The GAP Group, 2018) offers methods for constructing the subgroup lattice for a given group, and stores several data libraries for special groups and their subgroup lattices. Computing all subgroups of a group is generally intense, primarily applied to small groups. Common practices includes enumerating subgroups up to conjugacy (also supported in GAP), then computing abstractions within the conjugacy class is easy by Theorem 9.

To identify the "top", we start with the transformation group of $X = \mathbb{R}^n$ and then consider special subgroups of $\mathsf{F}(\mathbb{R}^n)$ and special subspaces of $\mathbb{R}^n$. We derive a principle that allows us to hierarchically and recursively break the enumeration problem into smaller and smaller enumeration subproblems. The breakdown can guide us in restricting both the type of subgroups and the type of subspaces, so that the resulting abstraction (semi)universe fits our desiderata and can be computed in practice. Figure 9 presents an outline consisting of special subgroups and subspaces considered in this section as well as their hierarchies.

We do not claim originality of the content in this section, many of which are known but discussed in different contexts (e.g. isometries, crystallography, music). Our contribution is to unify various existing pieces in a single, general setting regarding affine transformations, and draw connections to applications.

Preliminaries (Appendix A.6): group homomorphism, isomorphism ($\cong$); normalizer of a set in a group ($\mathrm{N}_G(S) := \{g \in G \mid gSg^{-1} = S\}$), normal subgroup ($\trianglelefteq$); group decomposition, inner semi-direct product, outer semi-direct product ($\rtimes$).

4.1 The Affine Transformation Group $\mathsf{AFF}(\mathbb{R}^n)$

An *affine transformation* of $\mathbb{R}^n$ is a function $f_{A,u} : \mathbb{R}^n \to \mathbb{R}^n$ of the form

$$f_{A,u}(x) = Ax + u \quad \text{for any } x \in \mathbb{R}^n,$$

where $A \in \mathsf{GL}_n(\mathbb{R})$ is an $n \times n$ matrix (invertible) and $u \in \mathbb{R}^n$ is an n-dimensional vector. Let $\mathsf{AFF}(\mathbb{R}^n)$ denote the set of all affine transformations of $\mathbb{R}^n$. There are two special cases:

1. A *translation* of $\mathbb{R}^n$ is a function $t_u : \mathbb{R}^n \to \mathbb{R}^n$ of the form $x \mapsto x + u$ where $u \in \mathbb{R}^n$; we use $\mathsf{T}(\mathbb{R}^n)$ to denote the set of all translations of $\mathbb{R}^n$.
2. A *linear transformation* of $\mathbb{R}^n$ is a function $r_A : \mathbb{R}^n \to \mathbb{R}^n$ of the form $x \mapsto Ax$ where $A \in \mathsf{GL}_n(\mathbb{R})$; we use $\mathsf{L}(\mathbb{R}^n)$ to denote the set of all linear transformations of $\mathbb{R}^n$.

It is easy to check that $\mathsf{T}(\mathbb{R}^n), \mathsf{L}(\mathbb{R}^n) \le \mathsf{AFF}(\mathbb{R}^n) \le \mathsf{F}(\mathbb{R}^n)$; further, $(\mathsf{T}(\mathbb{R}), \circ)$ and $(\mathsf{L}(\mathbb{R}^n), \circ)$ are isomorphic to $(\mathbb{R}^n, +)$ and $(\mathsf{GL}_n(\mathbb{R}), \cdot)$, respectively. It is known that

$$\mathsf{AFF}(\mathbb{R}^n) = \mathsf{T}(\mathbb{R}^n) \circ \mathsf{L}(\mathbb{R}^n) \cong \mathsf{T}(\mathbb{R}^n) \rtimes \mathsf{L}(\mathbb{R}^n) \cong \mathbb{R}^n \rtimes \mathsf{GL}_n(\mathbb{R}).$$

So every affine transformation can be uniquely identified with a pair $(u, A) \in \mathbb{R}^n \rtimes \mathsf{GL}_n(\mathbb{R})$. In particular, the identity transformation is identified with $(\mathbf{0}, I)$, the translation group $\mathsf{T}(\mathbb{R}^n)$ is identified with $\{(u, I) \mid u \in \mathbb{R}^n\}$, and the linear transformation group $\mathsf{L}(\mathbb{R}^n)$ is identified with $\{(\mathbf{0}, A) \mid A \in \mathsf{GL}_n(\mathbb{R})\}$. Under this identification, compositions and inverses of affine transformations become

$$(u, A)(u', A') = (u + Au', AA') \quad \text{and} \quad (u, A)^{-1} = (-A^{-1}u, A^{-1}). \tag{1}$$

The above identification well defines two functions $\ell : \mathsf{AFF}(\mathbb{R}^n) \to \mathsf{GL}_n(\mathbb{R})$ and $\tau : \mathsf{AFF}(\mathbb{R}^n) \to \mathbb{R}^n$ to extract the linear and translation part of an affine transformation, respectively, where

$$\ell(f_{A,u}) = A, \ \tau(f_{A,u}) = u \quad \text{for any } f_{A,u} \in \mathsf{AFF}(\mathbb{R}^n).$$

Now we can start our journey towards a complete identification of every subgroup H of $\mathsf{AFF}(\mathbb{R}^n)$. We introduce the first foundational quantity $T := \mathsf{T}(\mathbb{R}^n) \cap H$, which is the set of pure translations in H, called the *translation subgroup* of H. It is easy to check that $T \trianglelefteq H$ since translations are normal in affine transformations. Therefore, the quotient group $H/T = \{T \circ h \mid h \in H\}$ is well-defined. The elements in H/T are called *cosets*. The following theorems reveal more structures of H/T, the second foundational quantity.

Lemma 11 $\ell : \mathsf{AFF}(\mathbb{R}^n) \to \mathsf{GL}_n(\mathbb{R})$ *is a homomorphism. (An easy check.)*

Theorem 12 *Let $H \leq \mathsf{AFF}(\mathbb{R}^n)$, $T = \mathsf{T}(\mathbb{R}^n) \cap H$. Then $h, h' \in H$ are in the same coset in H/T if and only if they have the same linear part, i.e. $\ell(h) = \ell(h')$. (Proof in Appendix B.6.)*

Theorem 13 *Let $H \leq \mathsf{AFF}(\mathbb{R}^n)$, $T = \mathsf{T}(\mathbb{R}^n) \cap H$. If $h, h' \in H$ are in the same coset in H/T, then $\tau(h') - \tau(h) \in \tau(T) := \{u \mid t_u \in T\}$. (Proof in Appendix B.7.)*

Remark 14 *Theorems 12 and 13 present two characterizations of elements in the same coset in H/T, respectively. The former, through the linear part, is an if-and-only-if characterization; whereas the latter, through the translation part, is necessary but not sufficient.*

Theorem 15 *Let H, T as defined in Theorem 12. $H/T \cong \ell(H)$. (Proof in Appendix B.8.)*

Remark 16 *Theorem 15 can be proved directly from the first isomorphism theorem, by recognizing $\ell|_H$ is a homomorphism whose kernel and image are T and $\ell(H)$, respectively. However, the above proof explicitly gives the isomorphism $\bar{\ell}$ which is useful in the sequel.*

Theorem 17 (Compatibility) *Let H, T as defined in Theorem 12. For any $A \in \ell(H)$ and $v \in \tau(T)$, we have $Av \in \tau(T)$. Further, if we define a function $\cdot : \ell(H) \times \tau(T) \to \tau(T)$ of the form $(A, v) \mapsto Av$, then $\cdot$ is a group action of $\ell(H)$ on $\tau(T)$. (Proof in Appendix B.9.)*

So far, we have seen that for any subgroup $H \leq \mathsf{AFF}(\mathbb{R}^n)$, its subset of pure translations $T := \mathsf{T}(\mathbb{R}^n) \cap H$ is a normal subgroup of H; T is also a normal subgroup of $\mathsf{T}(\mathbb{R}^n)$, since $\mathsf{T}(\mathbb{R}^n)$ is a commutative group. As a result, both quotient groups H/T and $\mathsf{T}(\mathbb{R}^n)/T$ are well-defined. We next introduce a function, called a *vector system*, which connects the two quotient groups. It turns out that vector systems comprise the last piece of information that leads to a complete identification of every subgroup of $\mathsf{AFF}(\mathbb{R}^n)$. Note that $H/T \cong \ell(H)$ (Theorem 15) and $\mathsf{T}(\mathbb{R}^n)/T \cong \mathbb{R}^n/\tau(T)$; thus for conceptual ease (think in terms of matrices and vectors), we introduce vector systems connecting $\ell(H)$ and $\mathbb{R}^n/\tau(T)$ instead.

Definition 18 (Vector system) *For any $L \leq \mathsf{GL}_n(\mathbb{R})$ and $V \leq \mathbb{R}^n$, an (L, V)-vector system is a function $\xi : L \to \mathbb{R}^n/V$, which in addition satisfies the following two conditions:*

1. *compatibility condition: for any $A \in L$, $AV = \{Av \mid v \in V\} = V$;*
2. *cocycle condition: for any $A, A' \in L$, $\xi(AA') = \xi(A) + A\xi(A')$.*

Note: *elements in $\mathbb{R}^n/V$ are cosets of the form $V + u$ for $u \in \mathbb{R}^n$. It is easy to check: for any two cosets in $\mathbb{R}^n/V$, the sum*

$$(V + u) + (V + u') = \{v + u + v' + u' \mid v, v' \in V\} = V + (u + u');$$

for any $A \in L$ and any coset in $\mathbb{R}^n/V$, the product

$$A(V + u) = \{A(v + u) \mid v \in V\} = V + Au.$$

So, the sum and product in the cocycle condition are defined in the above sense.

We use $\Xi_{L,V}$ to denote the family of all (L, V)-vector systems. One can check that $\Xi_{L,V} \neq \emptyset$ if and only if L, V are compatible (consider the trivial vector system $\xi^0_{L,V}$ given by $\xi^0_{L,V}(A) = V$ for all $A \in L$). We use $\Xi^* := \{\Xi_{L,V} \mid L \leq \mathsf{GL}_n(\mathbb{R}), V \leq \mathbb{R}^n \text{ compatible}\}$ to denote the universe of all vector systems. The universe of all vector systems Ξ^* can be parameterized by the set of compatible pairs $(L, V) \in \mathcal{H}^*_{\mathsf{GL}_n(\mathbb{R})} \times \mathcal{H}^*_{\mathbb{R}^n}$. The reason is straightforward: L and V respectively define the domain and codomain of a function, and two functions are different if either their domains or their codomains are different.

Lemma 19 *Let $L \leq \mathsf{GL}_n(\mathbb{R})$, $V \leq \mathbb{R}^n$, and $\xi \in \Xi_{L,V}$, then 1) for the identity matrix $I \in L$, $\xi(I) = V$; 2) for any $A \in L$, $\xi(A^{-1}) = -A^{-1}\xi(A)$. (Proof in Appendix B.10.)*

Theorem 20 (Affine subgroup identification) *Let*

$$\Sigma := \{(L, V, \xi) \mid L \leq \mathsf{GL}_n(\mathbb{R}), V \leq \mathbb{R}^n, \xi \in \Xi_{L,V}\},$$

then there is a bijection between $\mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$ and Σ. (Proof in Appendix B.11.)

Remark 21 *The bijection Ψ from $\mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$ to Σ allows us to use the latter to parameterize the former. Further, through the inverse function Ψ^{-1}, we can enumerate affine subgroups by enumerating triplets $(L, V, \xi) \in \Sigma$, or more specifically, by enumerating matrix subgroups of $\mathsf{GL}_n(\mathbb{R})$, vector subgroups of $\mathbb{R}^n$, and then vector systems for every compatible pair of a matrix subgroup and a vector subgroup. Note that enumeration for each element in the triplet is still not practical if no restriction is imposed. Nevertheless, we have broken the original subgroup enumeration problem into three smaller enumeration problems. More importantly, we are now more directed in imposing restrictions on both subgroups and spaces, under which the three smaller enumerations become practical. We will discuss these restrictions (e.g. being isometric, finite, discrete, compact) in more detail in the sequel.*

4.2 The Isometry Group $\mathsf{ISO}(\mathbb{R}^n)$

One way to restrict $\Sigma := \{(L, V, \xi) \mid L \leq \mathsf{GL}_n(\mathbb{R}), V \leq \mathbb{R}^n, \xi \in \Xi_{L,V}\}$ is to consider a special subgroup of $\mathsf{GL}_n(\mathbb{R})$. Instead of all subgroups of $\mathsf{GL}_n(\mathbb{R})$, we consider only subgroups consisting of *orthogonal matrices*. This restriction gives rise to the subgroup lattice $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$ where $\mathsf{ISO}(\mathbb{R}^n)$ denotes the group of isometries of $\mathbb{R}^n$. In this subsection, we first give an overview of $\mathsf{ISO}(\mathbb{R}^n)$, and then cast $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$ in the big picture of $\mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$ and Σ.

An *isometry* of $\mathbb{R}^n$, with respect to the Euclidean distance d, is a transformation $h : \mathbb{R}^n \to \mathbb{R}^n$ which preserves distances: $\mathrm{d}(h(x), h(x')) = \mathrm{d}(x, x')$, for all $x, x' \in \mathbb{R}^n$. We use $\mathsf{ISO}(\mathbb{R}^n)$ to denote the set of all isometries of $\mathbb{R}^n$, which is a subgroup of the transformation group $\mathsf{F}(\mathbb{R}^n)$. So, we call $\mathsf{ISO}(\mathbb{R}^n)$ the *isometry group* of $\mathbb{R}^n$.

A *(generalized) rotation* of $\mathbb{R}^n$ is a linear transformation $r_A : \mathbb{R}^n \to \mathbb{R}^n$ given by $x \mapsto Ax$, for some orthogonal matrix $A \in \mathsf{O}_n(\mathbb{R}) := \{A \in \mathbb{R}^{n\times n} \mid A^\top = A^{-1}\} \leq \mathsf{GL}_n(\mathbb{R})$. We use $\mathsf{R}(\mathbb{R}^n)$ to denote the set of all rotations of $\mathbb{R}^n$, which is a subgroup of the linear transformation group $\mathsf{L}(\mathbb{R}^n)$. So, we call $\mathsf{R}(\mathbb{R}^n)$ the *rotation group* of $\mathbb{R}^n$.

There are two key characterizations of $\mathsf{ISO}(\mathbb{R}^n)$. The first one regards its components:

$$\mathsf{ISO}(\mathbb{R}^n) = \langle \mathsf{T}(\mathbb{R}^n) \cup \mathsf{R}(\mathbb{R}^n) \rangle \quad \text{where } \mathsf{T}(\mathbb{R}^n) \cap \mathsf{R}(\mathbb{R}^n) = \{\mathrm{id}\}.$$

This characterization says that $\mathsf{ISO}(\mathbb{R}^n)$ comprises exclusively translations, rotations, and their finite compositions. Note that we can rewrite the above characterization as $\mathsf{T}(\mathbb{R}^n) \vee \mathsf{R}(\mathbb{R}^n) = \mathsf{ISO}(\mathbb{R}^n)$ and $\mathsf{T}(\mathbb{R}^n) \wedge \mathsf{R}(\mathbb{R}^n) = \{\mathrm{id}\}$. This determines the positions of the four subgroups $\{\mathrm{id}\}$, $\mathsf{T}(\mathbb{R}^n)$, $\mathsf{R}(\mathbb{R}^n)$, and $\mathsf{ISO}(\mathbb{R}^n)$ in the subgroup lattice $(\mathcal{H}^*_{\mathsf{F}(\mathbb{R}^n)}, \leq)$, which forms a diamond shape in the direct acyclic graph in Figure 9a. The second characterization of $\mathsf{ISO}(\mathbb{R}^n)$ regards a unique representation for every isometry of $\mathbb{R}^n$, which is done by a group decomposition of $\mathsf{ISO}(\mathbb{R}^n)$ as semi-direct products:

$$\mathsf{ISO}(\mathbb{R}^n) = \mathsf{T}(\mathbb{R}^n) \circ \mathsf{R}(\mathbb{R}^n) \cong \mathbb{R}^n \rtimes \mathsf{O}_n(\mathbb{R}).$$

This characterization says that every isometry of $\mathbb{R}^n$ can be uniquely represented as an affine transformation $f_{A,u} \in \mathsf{AFF}(\mathbb{R}^n)$ where $A \in \mathsf{O}_n(\mathbb{R})$ and $u \in \mathbb{R}^n$. This further implies that $\mathsf{ISO}(\mathbb{R}^n)$ is a special subgroup of $\mathsf{AFF}(\mathbb{R}^n)$.

Let $\Psi : \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)} \to \Sigma$ be the bijection defined in the proof of Theorem 20, and let

$$\Sigma' := \{(L, V, \xi) \mid L \leq \mathsf{O}_n(\mathbb{R}), V \leq \mathbb{R}^n, \xi \in \Xi_{L,V}\} \subseteq \Sigma.$$

One can check: $\Psi^{-1}(\Sigma') = \mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$. This means $\Psi|_{\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}} : \mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)} \to \Sigma'$ is well-defined and bijective. Therefore, the subgroups of $\mathsf{ISO}(\mathbb{R}^n)$ can be enumerated by the triplets in Σ' in a similar manner as in Remark 21. The only difference is that we now enumerate subgroups of $\mathsf{O}_n(\mathbb{R})$ instead of the entire $\mathsf{GL}_n(\mathbb{R})$.

Note that restricting to subgroups of $\mathsf{O}_n(\mathbb{R})$ does not really make the enumeration problem practical. However, there are many ways of imposing additional restrictions on $\mathsf{ISO}(\mathbb{R}^n)$ to eventually achieve practical enumerations. We want to point out that there is no universal way of constraining the infinite enumeration problem into a practical one: the design of restrictions is most effective if it is consistent with the underlying topic domain. So, for instance, one can start with his/her intuition to try out some restrictions whose effectivenesses can be verified via a subsequent learning process (cf. Section 7). In the next subsection, we give two examples to illustrate some of the existing design choices that have been made in two different domains.

4.3 Special subgroups of $\mathsf{ISO}(\mathbb{R}^n)$ used in Chemistry and Music

From two examples, we show how additional restrictions can be imposed to yield a finite collection of subgroups of $\mathsf{ISO}(\mathbb{R}^n)$, capturing different parts of the infinite subgroup lattice $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$. The two examples are from two different topic domains: one is from chemistry (more precisely, crystallography), the other is from music. The ways of adding restrictions in these two examples are quite different: one introduces conjugacy relations to obtain a finite collection of subgroup types; the other restricts the space to be discrete or even finite.

4.3.1 Crystallographic Space Groups

In crystallography, symmetry is used to characterize crystals, to identify repeating parts of molecules, and to simplify both data collection and subsequent calculations. Further, the symmetry of physical properties of a crystal such as thermal conductivity and optical activity has a strong connection with the symmetry of the crystal. So, a thorough knowledge of symmetry is crucial to a crystallographer. A complete set of symmetry classes is captured by a collection of 230 unique 3-dimensional space groups. However, space groups represent a special type of subgroups of $\mathsf{ISO}(\mathbb{R}^n)$ which can be defined in general for any dimension.

We give a short review of known results from crystallography, and then identify space groups in the parametrization set Σ that we derived earlier. A *crystallographic space group* or *space group* Γ is a discrete (with respect to the subset topology) and cocompact (i.e. the abstraction space $\pi(\Gamma) := \mathbb{R}^n/\Gamma$ is compact with respect to the quotient topology) subgroup of $\mathsf{ISO}(\mathbb{R}^n)$. So, if the underlying topic domain indeed considers only compact abstractions, space groups are good candidates. A major reason is that for a given dimension, there exist only finitely many space groups (up to isomorphism or affine conjugacy) by Bieberbach's second and third theorems (Bieberbach, 1911; Charlap, 2012).

Bieberbach's first theorem (Bieberbach, 1911; Charlap, 2012) gives an equivalent characterization of space groups: a subgroup Γ of $\mathsf{ISO}(\mathbb{R}^n)$ is a space group if $T := \mathsf{T}(\mathbb{R}^n) \cap \Gamma$ is isomorphic to $\mathbb{Z}^n$ and $\tau(T)$ spans $\mathbb{R}^n$. In particular, for a space group Γ in standard form, we have $\ell(\Gamma) \leq \mathsf{GL}_n(\mathbb{Z})$, $\tau(T) = \mathbb{Z}^n$ (Eick and Souvignier, 2006). Therefore, we can use

$$\Sigma''_{cryst} := \{(L, V, \xi) \mid L \leq \mathsf{GL}_n(\mathbb{Z}), V = \mathbb{Z}^n, \xi \in \Xi_{L,V}\} \subseteq \Sigma' \subseteq \Sigma$$

to parameterize all space groups in standard form. For every $L \leq \mathsf{GL}_n(\mathbb{Z})$, the enumeration of vector systems $\xi \in \Xi_{L,\mathbb{Z}^n}$ is made feasible in Zassenhaus (1948) and implemented in the GAP package CrystCat (Felsch and Gähler, 2000).

4.3.2 Isometries of $\mathbb{Z}^n$ in Music

Another example of obtaining a finite collection of subgroups of $\mathsf{ISO}(\mathbb{R}^n)$ regards computational music theory and an extension to our earlier work (Yu et al., 2016; Yu and Varshney, 2017; Yu et al., 2017). We impose restrictions on the space, focusing on discrete subsets of $\mathbb{R}^n$ that represent equal-tempered music pitches. These further yield restrictions on the subgroups under consideration, namely only those *stabilizing* the restricted subsets of $\mathbb{R}^n$. We consider isometries of $\mathbb{Z}^n$, while further restricting to a finite subspace such as $\mathbb{Z}^n_{[a,b]}$ or $\mathbb{Z}^n_{[-b,b]}$ (Figure 9b) will be presented in Section 6. We first introduce a few definitions regarding the space $\mathbb{Z}^n$ in parallel with their counterparts regarding $\mathbb{R}^n$, and then establish their equivalences under *restricted setwise stabilizers*.

Definition 22 *An isometry of $\mathbb{Z}^n$, under the Euclidean distance* d *(or more precisely* $\mathrm{d}|_{\mathbb{Z}^n}$*) is a function $h' : \mathbb{Z}^n \to \mathbb{Z}^n$ preserving distances: $\mathrm{d}(h'(x), h'(x')) = \mathrm{d}(x, x')$, for all $x, x' \in \mathbb{Z}^n$.*

Definition 23 *A translation of $\mathbb{Z}^n$ is a function $t'_u : \mathbb{Z}^n \to \mathbb{Z}^n$ of the form $x \mapsto x + u$, where $u \in \mathbb{Z}^n$ is called a translation vector.*

Definition 24 *A (generalized) rotation of $\mathbb{Z}^n$ is a function $r'_A : \mathbb{Z}^n \to \mathbb{Z}^n$ of the form $x \mapsto Ax$, where $A \in \mathsf{O}_n(\mathbb{Z}) := \{A \in \mathbb{Z}^{n\times n} \mid A^\top = A^{-1}\}$ is called a rotation matrix.*

We use $\mathsf{ISO}(\mathbb{Z}^n), \mathsf{T}(\mathbb{Z}^n), \mathsf{R}(\mathbb{Z}^n)$ to respectively denote the set of all isometries, translations, and rotations of $\mathbb{Z}^n$. It is easy to check that $(\mathsf{T}(\mathbb{Z}^n), \circ) \cong (\mathbb{Z}^n, +)$ and $(\mathsf{R}(\mathbb{Z}^n), \circ) \cong (\mathsf{O}_n(\mathbb{Z}), \cdot)$; further, $\mathsf{T}(\mathbb{Z}^n), \mathsf{R}(\mathbb{Z}^n) \leq \mathsf{F}(\mathbb{Z}^n)$, and $\mathsf{T}(\mathbb{Z}^n), \mathsf{R}(\mathbb{Z}^n) \subseteq \mathsf{ISO}(\mathbb{Z}^n)$, so translations and rotations of $\mathbb{Z}^n$ are transformations and are also isometries. Next, we show $\mathsf{ISO}(\mathbb{Z}^n) \leq \mathsf{F}(\mathbb{Z}^n)$.

Definition 25 *Let $G \leq \mathsf{F}(X)$, $Y \subseteq X$, and $G_Y := \{g \in G \mid g(Y) = Y\}$ be the setwise stabilizer of Y under G. The restricted setwise stabilizer of Y under G is the set $G_Y|_Y := \{g|_Y \mid g \in G_Y\}$, where $g|_Y : Y \to Y$ is the (surjective) restriction of the function g to Y.*

Theorem 26 *For any $Y \subseteq X$, $\mathsf{F}(Y) = \mathsf{F}(X)_Y|_Y$. (Proof in Appendix B.12.)*

Corollary 27 *$\mathsf{F}(\mathbb{Z}^n) = \mathsf{F}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$.*

Theorem 28 *$\mathsf{T}(\mathbb{Z}^n) = \mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, and $\mathsf{R}(\mathbb{Z}^n) = \mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$. (Proof in Appendix B.13.)*

Theorem 29 *$\mathsf{ISO}(\mathbb{Z}^n) = \mathsf{ISO}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$. (Proof in Appendix B.14.)*

Remark 30 *Through restricted setwise stabilizers, Corollary 27 and Theorems 28 and 29 collectively verify that transformations, translations, rotations, and isometries of $\mathbb{Z}^n$ are precisely those transformations, translations, rotations, and isometries of $\mathbb{R}^n$ that stabilize $\mathbb{Z}^n$, respectively. In particular, $\mathsf{T}(\mathbb{Z}^n), \mathsf{R}(\mathbb{Z}^n) \leq \mathsf{ISO}(\mathbb{Z}^n) \leq \mathsf{F}(\mathbb{Z}^n)$.*

The parallels between translations, rotations, isometries of $\mathbb{Z}^n$ and their counterparts of $\mathbb{R}^n$ yield the two characterizations of $\mathsf{ISO}(\mathbb{Z}^n)$ which are parallel to the those of $\mathsf{ISO}(\mathbb{R}^n)$:

$$\begin{aligned} &\mathsf{ISO}(\mathbb{Z}^n) = \langle \mathsf{T}(\mathbb{Z}^n) \cup \mathsf{R}(\mathbb{Z}^n) \rangle \quad \text{where } \mathsf{T}(\mathbb{Z}^n) \cap \mathsf{R}(\mathbb{Z}^n) = \{\mathrm{id}\}; \\ &\mathsf{ISO}(\mathbb{Z}^n) = \mathsf{T}(\mathbb{Z}^n) \circ \mathsf{R}(\mathbb{Z}^n) \cong \mathbb{Z}^n \rtimes \mathsf{O}_n(\mathbb{Z}). \end{aligned}$$

This further yields the parametrization of $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$ by

$$\Sigma''_{isozn} := \{(L, V, \xi) \mid L \leq \mathsf{O}_n(\mathbb{Z}), V \leq \mathbb{Z}^n, \xi \in \Xi''_{L,V}\} \subseteq \Sigma' \subseteq \Sigma,$$

where $\Xi''_{L,V} := \{\xi \in \Xi_{L,V} \mid \xi(L) \subseteq \mathbb{Z}^n/V\} \subseteq \Xi_{L,V}$. Note that $\mathsf{ISO}(\mathbb{Z}^n)$ still have infinitely many subgroups, since the choices for V and ξ are still unlimited. Next we will show how to enumerate a finite subset from $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$ when considering the music domain.

The space of equal-tempered music pitches can be denoted by $\mathbb{Z}$. Every adjacent pitch is separated by a half-step denoted by the integer 1, i.e. the distance between every adjacent keys (regardless of black or white) in a piano keyboard. While the absolute integer assigned to each music pitch is not essential, in the standard MIDI convention, C_4 (the middle C) is 60, $\mathrm{C}\sharp_4$ is 61, and so forth. Therefore, the space $\mathbb{Z}^n$ represents the space of chords consisting of n pitches. For instance, $\mathbb{Z}^3$ denotes the space of trichords, $\mathbb{Z}^4$ denotes the space of tetrachords, and so forth. Known music transformations of fixed-size chords (Tymoczko, 2010; Lewin, 2010) can be summarized as a subset of the following parametrization set

$$\Sigma''_{music} := \{(L, V, \xi) \mid L \leq \mathsf{O}_n(\mathbb{Z}), V \in \mathcal{H}^M_{\mathbb{Z}^n}, \xi = \xi^0_{L,V} \in \Xi''_{L,V}\} \subseteq \Sigma''_{isozn},$$

where $\mathcal{H}^M_{\mathbb{Z}^n} := \{\langle \mathbf{1} \rangle, (12\mathbb{Z})^n, \langle \mathbf{1} \rangle \vee (12\mathbb{Z})^n\}$ is a finite collection of music translation subgroups including music transpositions, octave shifts, and their combinations; $\xi^0_{L,V}$ is the trivial vector system given by $\xi^0_{L,V}(A) = V$ for any $A \in L$ requiring the inclusion of all rotations to include music permutations and inversions. Together with the fact that $\mathsf{O}_n(\mathbb{Z})$ is finite, the enumeration of each element in the triplet (L, V, ξ) is finite, yielding a finite Σ''_{music}.

Notably, the parametrization set Σ''_{music} is not just for recovering known music-theoretic concepts (e.g. major, minor chords and their inversions) but also completing the known by forming the music "closure" Σ''_{music}. Such a "closure" can be further explored for new music concepts such as new scales or new harmonic foundations (e.g. figured soprano).

4.4 Section Summary

We started from the full transformation group of $\mathbb{R}^n$—the top vertex in the subgroup lattice $(\mathcal{H}^*_{\mathsf{F}(\mathbb{R}^n)}, \leq)$—and moved down to the affine group of $\mathbb{R}^n$. Focusing on $\mathsf{AFF}(\mathbb{R}^n)$, we derived a complete identification of its subgroups by constructing a parametrization set Σ and a bijection $\Psi : \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)} \to \Sigma$, where each subgroup of $\mathsf{AFF}(\mathbb{R}^n)$ bijectively corresponds to a triplet in Σ. We further moved down in $(\mathcal{H}^*_{\mathsf{F}(\mathbb{R}^n)}, \leq)$ from the affine group of $\mathbb{R}^n$ to the isometry group of $\mathbb{R}^n$. Focusing on $\mathsf{ISO}(\mathbb{R}^n)$, we identified the parametrization of $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$

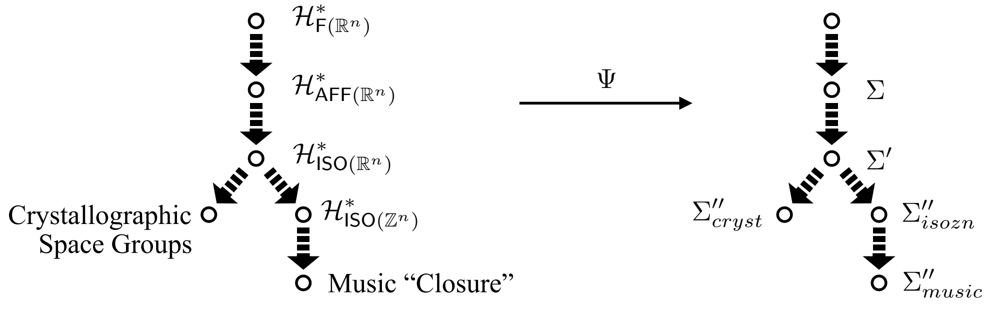

$\mathcal{H}^*_{\mathsf{F}(\mathbb{R}^n)}$
$\mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$
$\mathcal{H}^*_{\mathsf{ISO}(\mathbb{R}^n)}$
Crystallographic Space Groups
$\mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$
Music "Closure"
Ψ
Σ
Σ'
Σ''_{cryst}
Σ''_{isozn}
Σ''_{music}

Figure 10: Roadmap of the top-down paths in terms of collection of subgroups (left) and the corresponding parametrization paths (right) under the parametrization map Ψ.

by a subset $\Sigma' \subseteq \Sigma$. From there, we made a dichotomy in our top-down path, and presented two examples to obtain two collections of subgroups used in two different topic domains. One is a finite collection of space groups (in standard form and up to affine conjugacy) used in crystallography and parameterized by $\Sigma''_{cryst} \subseteq \Sigma'$; the other is a finite completion of existing music concepts parameterized by $\Sigma''_{music} \subseteq \Sigma''_{isozn} \subseteq \Sigma'$. A complete roadmap that we have gone through is summarized in Figure 10.

5. The Bottom-Up Approach: Generating Set

Our bottom-up approach first identifies a finite subset $S \subseteq \mathsf{F}(X)$ as a generating set, and then generate a partial subgroup lattice $\mathcal{H}_{\langle S\rangle}$ by computing $\langle S'\rangle$ for each $S' \subseteq S$, starting from smaller subgroups. More specifically, we start from the trivial subgroup $\langle\emptyset\rangle = \{\mathrm{id}\}$, i.e. the bottom vertex in the direct acyclic graph for $\mathcal{H}^*_{\mathsf{F}(X)}$, and first construct cyclic subgroups $\langle s\rangle$ for each $s \in S$. We then recursively construct larger subgroups from smaller ones via the join operation, which corresponds to gradually moving upwards in the graph for $\mathcal{H}^*_{\mathsf{F}(X)}$. Computing abstractions via this bottom-up approach is more efficient by leveraging the strong duality in Theorem 6. The dual of the above induction procedure induces a mirrored induction that directly computes the abstraction semiuniverse without explicitly computing the corresponding subgroups. In general, the induction procedure will yield at most $2^{|S|}$ subgroups for any given subset $S \subseteq \mathsf{F}(X)$, and will not produce the complete subgroup sublattice $\mathcal{H}^*_{\langle S\rangle}$ unless $S = \langle S\rangle$. The missing subgroups can be made up by adding more generators, which may hinder efficiency. At the end of this section, we discuss the trade-off between expressiveness and efficiency when designing a generating set in practice.

5.1 From Generating Set to Subgroup (Semi)Lattice

Let $S \subseteq \mathsf{F}(X)$ be a finite subset consisting of transformations of a set X. We construct a collection $\mathcal{H}_{\langle S\rangle}$ consisting of subgroups of $\langle S\rangle$ where every subgroup is generated by a subset of S. To succinctly record this process and concatenate it with the abstraction generating chain, we introduce the following one-step *subgroup generating chain*:

$$\text{a subset of } \mathsf{F}(X) \xrightarrow{\textit{group generation}} \text{a subgroup of } \mathsf{F}(X),$$

which can be further encapsulated by the *subgroup generating function* defined as follows.

Definition 31 *The subgroup generating function is the mapping $\pi' : 2^{\mathsf{F}(X)} \to \mathcal{H}^*_{\mathsf{F}(X)}$ where $2^{\mathsf{F}(X)}$ is the power set of $\mathsf{F}(X)$, $\mathcal{H}^*_{\mathsf{F}(X)}$ is the collection of all subgroups of $\mathsf{F}(X)$, and for any $S \in 2^{\mathsf{F}(X)}$, $\pi'(S) := \langle S \rangle := \{s_k \circ \cdots \circ s_1 \mid s_i \in S \cup S^{-1}, i = 1, \ldots, k, k \in \mathbb{Z}_{\geq 0}\}$ where $S^{-1} := \{s^{-1} \mid s \in S\}$. By convention, $s_k \circ \cdots \circ s_1 = \mathrm{id}$ for $k = 0$, and $\pi'(\emptyset) = \langle \emptyset \rangle = \{\mathrm{id}\}$.*

We can now write $\mathcal{H}_{\langle S \rangle} = \pi'(2^S)$ for any finite subset $S \subseteq \mathsf{F}(X)$. Further, the subgroup generating chain and the abstraction generating chain can now be concatenated, denoted by the composition $\Pi := \pi \circ \pi'$. Like the abstraction generating function π, the subgroup generating function π' is not necessarily injective, since a generating set of a group is generally not unique; π' is surjective, since every subgroup per se is also its own generating set. The following theorem captures the structure of $\pi'(2^S)$ for a finite subset $S \subseteq \mathsf{F}(X)$.

Theorem 32 *Let $S \subseteq \mathsf{F}(X)$ be a finite subset, and π' be the subgroup generating function. Then $(\pi'(2^S), \leq)$ is a join-semilattice, but not necessarily a meet-semilattice. In particular,*

$$\pi'(A \cup B) = \pi'(A) \vee \pi'(B) \quad \text{for any } A, B \subseteq S. \qquad \text{(Proof in Appendix B.15.)}$$

Although the collection of subgroups generated by π' is not a lattice in general, it suffices to be a join-semilattice, since the family of abstractions generated by π is a meet-semiuniverse (Theorem 6). Through the strong duality, the closedness of $\pi'(2^S)$ under join is carried over to preserve the closedness of $\pi(\pi'(2^S))$ under meet. Preserving closednesses under join and meet has a significant practical implication: it directly yields an induction algorithm that implements $\Pi(2^S) := \pi \circ \pi'(2^S)$ from a finite subset S.

5.2 An Induction Algorithm

We describe an algorithmic implementation of $\Pi(2^S)$ from a finite generating set S for a finite space X. More specifically, the algorithmic problem here is as follows.

$$\begin{array}{rl} \text{Inputs:} & \text{1) a finite set } X \text{ (to be abstracted),} \\ & \text{2) a finite generating set } S \subseteq \mathsf{F}(X); \\ & \text{the underlying group action is: } \langle S \rangle \text{ acting on } X. \\ \text{Output:} & \text{the abstraction semiuniverse: } \Pi(2^S) = \pi \circ \pi'(2^S) = \{X/\langle S' \rangle \mid S' \subseteq S\}. \end{array} \qquad (2)$$

A naive way to solve Problem (2) may just follow the definition $\Pi = \pi \circ \pi'$ in two steps. It first computes the subgroup join-semilattice $\pi'(2^S)$ and then the abstraction meet-semiuniverse $\pi(\pi'(2^S))$. Yet, as mentioned in Remark 7, computing every abstraction of X by identifying orbits from a subgroup action can be expensive. Below, we first inspect the naive thinking and then introduce an induction algorithm that solves Problem (2).

A Naive Two-Step Implementation. Step 1: compute subgroups $\pi'(2^S) = \{\langle S' \rangle \mid S' \subseteq S\}$. Step 2: compute partitions $\pi(\pi'(2^S))$, i.e. for each $S' \subseteq S$ and $\pi'(S') = \langle S' \rangle$, compute the set of orbits $\pi(\pi'(S')) = X/\langle S' \rangle = \{\langle S' \rangle x \mid x \in X\}$. For each pair $x, x' \in X$, we need check whether they are in the same orbit—known as the *orbit identification* problem. The number of checks needed is $O(|X|^2)$ which is computationally expensive for large $|X|$. Yet, what makes this naive approach fail is that most checks may not end in finite time. Take $S' = \{s_1, s_2\}$ as an example, a brute-force check may be endless: "$x' = s_1(x)$?", "$x' = s_1^{-1}(x)$?", "$x' = s_2(x)$?", "$x' = s_1 \circ s_2 \circ s_2 \circ s_1^{-1}(x)$?", and so forth.

```
Input: a generator $s$ and a set $X$
Output: the base partition $\Pi(\{s\})$
Function BasePartn($s$):
    initialize label id: $l = 0$;
    for each point $x \in X$ do
        if $x$ is not labeled then
            initialize a new orbit:
              $O = \{x\}$;
            transform:
              $y = s(x)$;
            while $y \in X$ and $y \neq x$
             and $y$ is not labeled do
                enlarge the orbit:
                  $O = O \cup \{y\}$;
                transform:
                  $y = s(y)$;
            end
            if $y \notin X$ or $y = x$ then
                create a new label:
                  $l = l + 1$;
            end
            if $y$ is labeled then
                use $y$'s label:
                  $l = y$'s label;
            end
            label every point in the
             orbit $O$ by $l$;
        end
    end
    return the partition according
     to the labels;
```

Algorithm 1: Computing base partitions by tracing orbits: $O(|X|)$.

```
Input: two partitions $\mathcal{P}$ and $\mathcal{Q}$ of
       a set $X$
Output: the meet $\mathcal{P} \wedge \mathcal{Q}$, i.e.
        finest common
        coarsening of $\mathcal{P}$ and $\mathcal{Q}$
Function Meet($\mathcal{P}$, $\mathcal{Q}$):
    for each cell $Q \in \mathcal{Q}$ do
        initialize a new cell:
          $P_{merge} = \emptyset$;
        for each cell $P \in \mathcal{P}$ do
            if $P \cap Q \neq \emptyset$ then
                merge:
                  $P_{merge} = P_{merge} \cup P$;
                remove:
                  $\mathcal{P} = \mathcal{P} \backslash \{P\}$;
            end
        end
        insert:
          $\mathcal{P} = \mathcal{P} \cup \{P_{merge}\}$;
    end
    return $\mathcal{P}$;
```

Algorithm 2: Computing partitions generated from more than one generators inductively by taking the meet of two partitions computed earlier: $O(|\mathcal{P}||\mathcal{Q}|)$. Normally, all base partitions should be already computed and cached before running the induction steps.

An Induction Algorithm. Instead, we give an algorithm based on induction on $|S'|$ for all nonempty subsets $S' \subseteq S$. (Note: $\Pi(\emptyset) = X/\langle\emptyset\rangle$ is simply the finest partition of X.)
Base case: compute $\Pi(S')$ for $|S'| = 1$ (Algorithm 1) as orbits under a cyclic subgroup:

$$\Pi(S') = \{\langle S' \rangle x \mid x \in X\}. \tag{3}$$

Induction step: compute $\Pi(S')$ for $|S'| > 1$ (Algorithm 2) as the meet of two partitions:

$$\Pi(S') = \Pi(S'') \wedge \Pi(S''') \quad \text{for any } S'', S''' \subset S' \text{ and } S'' \cup S''' = S'. \tag{4}$$

In the base case, each partition, called a *base partition*, is explicitly computed by orbits. Yet orbit identification is feasible for one generator, say s, since $x, x' \in X$ are in the same

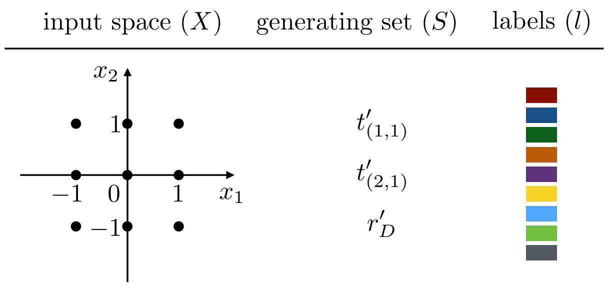


Figure 11: A toy example on which the induction algorithm is to be run.

orbit if and only if $x' = s^n(x)$ for some $n \in \mathbb{Z}$. We can do even better than the quadratic-time orbit identification: Algorithm 1 uses *orbit tracing*—linear in $|X|$. In the induction step, the meet operation bypasses the endless brute-force checks in orbit identification. Its correctness can be proved by leveraging Theorem 32 and the strong duality in Theorem 6:

$$\Pi(S') = \pi \circ \pi'(S'' \cup S''') = \pi(\pi'(S'') \vee \pi'(S''')) = \pi \circ \pi'(S'') \wedge \pi \circ \pi'(S''') = \Pi(S'') \wedge \Pi(S''').$$

The above holds for any $S'' \cup S''' = S'$. Yet, the run time can differ, to be discussed later.

Extension to Infinite Spaces. The induction algorithm, consisting of several runs of Algorithm 1 followed by several runs of Algorithm 2, works for any finitely generated group $\langle S \rangle$ acting on any finite set X, i.e. Problem (2). We can apply it in a more general setting, where $\langle S \rangle$ acts on a possibly infinite set $\tilde{X}$ and our interested input $X \subseteq \tilde{X}$.

$$\begin{array}{rl} \text{Inputs:} & \text{1) a possibly infinite set } \tilde{X} \text{ (to be abstracted),} \\ & \text{2) a finite generating set } S \subseteq \mathsf{F}(\tilde{X}), \\ & \text{3) an (arbitrary) finite subset } X \subseteq \tilde{X}; \\ & \text{the underlying group action is: } \langle S \rangle \text{ acting on } \tilde{X} \text{ (not } X). \\ \text{Output:} & \text{the abstraction semiuniverse (for } \tilde{X} \text{ restricted to } X) \\ & \Pi(2^S) = \pi \circ \pi'(2^S) = \{X/\langle S' \rangle \mid S' \subseteq S\}; \quad X/\langle S' \rangle := \{\langle S' \rangle x \cap X \mid x \in X\}. \end{array} \tag{5}$$

Here we are using a non-standard yet unambigiuous notation $X/\langle S' \rangle$ to mean the partition of X obtained from restricting $\tilde{X}/\langle S' \rangle$ to X, since the group action is on $\tilde{X}$ instead of X. Problem (5) is more general since it includes Problem (2) as a special case by setting $X = \tilde{X}$ for a finite $\tilde{X}$. So now, we can do computational abstraction on an infinite input space, and presents the result on whatever finite subspace is asked for. However, we need to be careful here. Even though the algorithm will run, it is not always accurate in the general setting (it is accurate for Problem (2) only): both Algorithm 1 and 2 may only give an approximation for Problem (5). A full discussion on this general setting, including where the approximation might occur and how to correct it, will be detailed in Section 6.

Run on an Toy Example. We walk through a complete run of the above induction algorithm on a toy example (Figure 11). In order to hit all cases and show the algorithm's full functionality, we draw the example from the general setting formalized in Problem (5):

$$X = \{x \in \mathbb{Z}^2 \mid \|x\|_\infty \le 1\} \subseteq \mathbb{Z}^2; \quad S = \{t'_{(1,1)}, t'_{(2,1)}, r'_D\} \subseteq \mathsf{F}(\mathbb{Z}^2) \text{ where } D = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

So here, $X = \{(-1,-1),(0,-1),(1,-1),(-1,0),(0,0),(1,0),(-1,1),(0,1),(1,1)\}$ is a nine-element set; S contains three generators, where $t'_{(1,1)}, t'_{(2,1)}$ are two translations by vectors $(1,1),(2,1)$, respectively, and r'_D is the reflection about the diagonal line ($x_2 = x_1$). More explicitly, for any $x = (x_1, x_2) \in X$, $t'_{(1,1)}(x) = (x_1+1, x_2+1)$, $t'_{(2,1)}(x) = (x_1+2, x_2+1)$, and $r'_D(x) = (x_2, x_1)$. The desired output for this toy example is the following abstraction semiuniverse consisting of $2^{|S|} = 2^3 = 8$ abstractions:

$$\begin{aligned}\Pi(2^S) = \{\ & X/\langle\emptyset\rangle,\\ & X/\langle t'_{(1,1)}\rangle,\ \ X/\langle t'_{(2,1)}\rangle,\ \ X/\langle r'_D\rangle,\\ & X/\langle\{t'_{(1,1)}, t'_{(2,1)}\}\rangle,\ \ X/\langle\{t'_{(2,1)}, r'_D\}\rangle,\ \ X/\langle\{r'_D, t'_{(1,1)}\}\rangle,\\ & X/\langle\{t'_{(1,1)}, t'_{(2,1)}, r'_D\}\rangle\ \}.\end{aligned}$$

Global procedure. In a full run of the induction procedure, we first run Algorithm 1 three independent times to compute the three base partitions $X/\langle t'_{(1,1)}\rangle$, $X/\langle t'_{(2,1)}\rangle$, $X/\langle r'_D\rangle$; we then run Algorithm 2 three independent times to compute the next three partitions $X/\langle\{t'_{(1,1)}, t'_{(2,1)}\}\rangle$, $X/\langle\{t'_{(2,1)}, r'_D\}\rangle$, $X/\langle\{r'_D, t'_{(1,1)}\}\rangle$, each of which is computed from its corresponding base partitions, e.g. $X/\langle\{t'_{(1,1)}, t'_{(2,1)}\}\rangle$ is computed from $X/\langle t'_{(1,1)}\rangle$ and $X/\langle t'_{(2,1)}\rangle$. Lastly, we run Algorithm 2 one more time to compute $X/\langle\{t'_{(1,1)}, t'_{(2,1)}, r'_D\}\rangle$ from earlier computed partitions, e.g. from $X/\langle\{t'_{(1,1)}, t'_{(2,1)}\}\rangle$ and $X/\langle r'_D\rangle$ (among many other choices). Note that $X/\langle\emptyset\rangle := \{\{x\} \mid x \in X\}$ is the finest partition of X, which does not require any algorithmic run. Next, we will zoom into this global process, and walk the readers through each individual run of Algorithm 1 and Algorithm 2 one at a time.

Algorithm 1. The three independent runs of this algorithm are illustrated in Figure 12. The sequential coloring of the nine dots in the input set X denotes the labeling process; the double slash (//) denotes the end of an iteration in the for-loop (so, the first two runs cost nine iterations each, and the third run costs six iterations). Both the coloring sequence and the double slashes mark the linear progress of the algorithm. Closer attention should be paid to the arrows: every arrow, regardless of its type, denotes an action of applying the transformation (specified by the single generator). However, the result of the transformation includes four cases (corresponding to the four conditions in Algorithm 1, namely $y \notin X$, $y = x$, y *is labeled*, and otherwise). Therefore, we adopt four different types of arrows to denote these four cases, respectively.

1. *An open arrow* indicates that the transformed point jumps out of the input set (see condition $y \notin X$). There is only one type of open arrow (arrowhead is $>$); contrarily, there are three types of triangle arrows (arrowhead is a filled triangle).
2. *A solid triangle arrow* indicates that the transformed point is an already labeled point (see condition y *is labeled*).
3. *A dashed triangle arrow* indicates that the transformed point coincides with an earlier point in the transformation sequence (see condition $y = x$).
4. *A dotted triangle arrow* indicates that the transformed point does not fall under any of the above conditions (see condition of the while-loop).

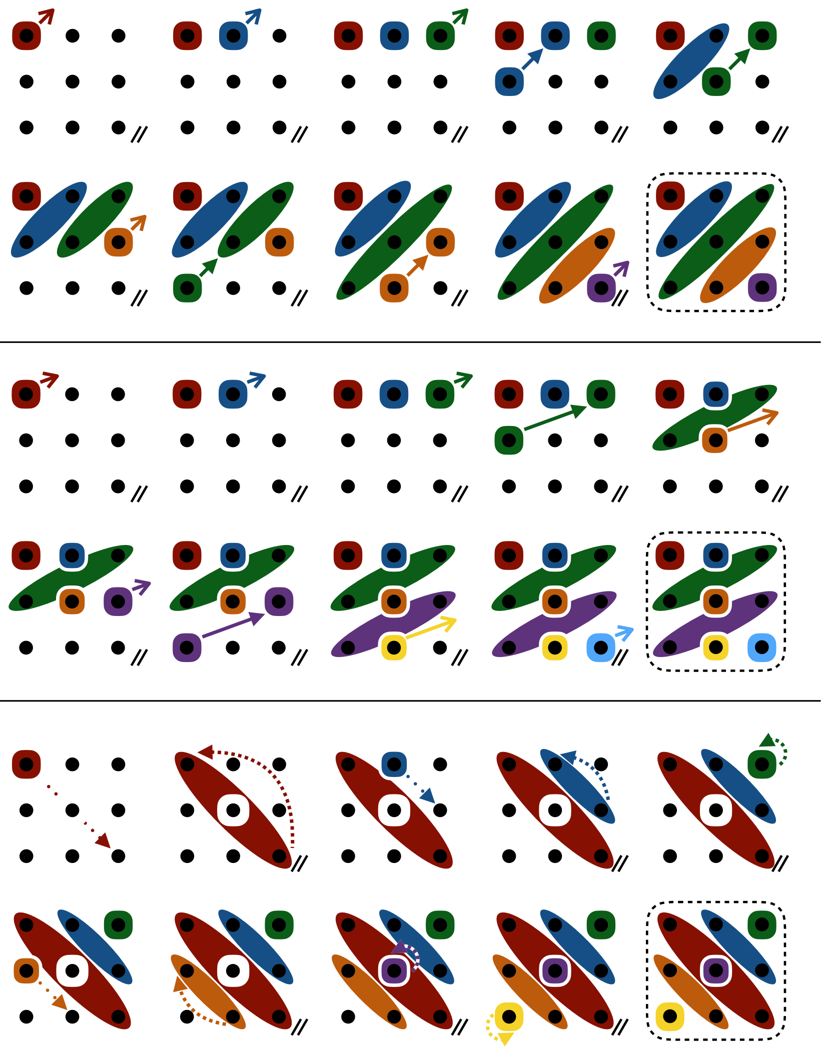


Figure 12: Three independent runs of Algorithm 1 for generators $t'_{(1,1)}$ (top), $t'_{(2,1)}$ (middle), and r'_D (bottom), respectively. Every color denotes the label of a partition cell; every double slash (//) marks the end of a for-loop iteration; every arrow indicates a transformation (see more details in the text for the four types of arrows). Final outputs are in dashed boxes.

The final output of each run of Algorithm 1—a partition of X whose cells are marked by colors—is shown in a dashed box in Figure 12.

Algorithm 2. One run of this algorithm is illustrated in Figure 13 (other runs are similar). On this run, the algorithm takes as inputs two partitions $\mathcal{P} = X/\langle t'_{(2,1)}\rangle$, $\mathcal{Q} = X/\langle t'_{(1,1)}\rangle$, and computes their meet $\mathcal{P} \wedge \mathcal{Q}$ for our desired output $X/\langle\{t'_{(1,1)}, t'_{(2,1)}\}\rangle$. This is a pretty standard algorithm that computes the meet of two partitions in general. The idea is to take one of the input partitions, say $\mathcal{Q}$, as a fixed reference partition, and iterates its cells in the outer loop; we then gradually change the other partition $\mathcal{P}$ *in place* from an input partition to the output partition. The changes are made in each inner loop: for each $Q \in \mathcal{Q}$, we collect

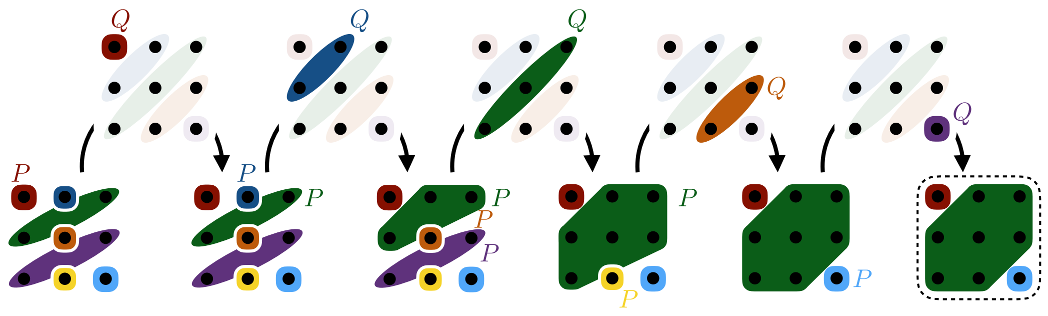


Figure 13: One run of Algorithm 2 from input partitions $\mathcal{P} = X/\langle t'_{(2,1)} \rangle$ and $\mathcal{Q} = X/\langle t'_{(1,1)} \rangle$. The outer for-loop over $Q \in \mathcal{Q}$ is shown in the top row; the inner for-loop over $P \in \mathcal{P}$ is shown in the bottom row. The marked P cells are the ones that intersect with the following Q cell. During the entire process, $\mathcal{Q}$ remains unchanged, whereas $\mathcal{P}$ changes in place from one of the algorithm inputs to the algorithm output. The final output is in the dashed box.

all cells in the current $\mathcal{P}$ that intersect with Q, and merge them into one cell. Take the third outer iteration in Figure 13 as an example where $Q = \{(-1,-1),(0,0),(1,1)\}$ (green): three out of the six cells in $\mathcal{P}$—namely $\{(-1,0),(0,1),(1,1)\}$ (green), $\{(0,0)\}$ (orange), and $\{(-1,-1),(1,0)\}$ (purple)—intersect with Q, so they are merged into one single cell (the green cell in the following $\mathcal{P}$). The output of this run of Algorithm 2—again a partition of X whose cells are marked by colors—is shown in a dashed box in Figure 13. After all runs of Algorithm 2 finish, the final output of the entire induction algorithm is in Figure 14.

Caution: correctness. It is important to notice that this toy example falls under the general setting formalized in Problem (5). The underlying group action here is $\langle S \rangle$ acting on $\mathbb{Z}^n$ (not X) and the input space X is a finite subset of $\mathbb{Z}^n$. So, not all output partitions in Figure 14 are correct. It is an easy check that while all base partitions are correct, the leftmost partition in the second row from the top in Figure 14 is only an approximation of the desired partition $X/\langle \{t_{(1,1)}, t_{(2,1)}\} \rangle = \{X\}$ (since $(1,1)$ and $(2,1)$ form a basis of $\mathbb{Z}^n$). However, every incorrect partition in Figure 14 can be corrected by running the additional "Expand-and-Restrict" technique right after each run of Algorithm 2. See Section 6 for more details, and particularly Section 6.1 for the rectification technique.

Computational Complexity. Recall in the general setting formalized in Problem (5), each of the desired abstractions of X is the partition $X/\langle S' \rangle = \{\langle S' \rangle x \cap X \mid x \in X\} = (\tilde{X}/\langle S' \rangle)|_X$, i.e. the partition $\tilde{X}/\langle S' \rangle$ restricted to the subspace X (see the precise definition in Definition 45). The crux of this problem is *orbit identification*: given any $x, x' \in X$, check whether there exists an $s \in \langle S' \rangle$ such that $x' = s \cdot x$. In some special cases, there might be smart ways of conducting these checks in a magnitude lower than $O(|X|^2)$ times (like in our *orbit tracing*). But things may get tough in general. So, before analyzing the computational complexity of our induction algorithm, we first discuss the complexity of the problem.

Complexity of the Problem. We analyze the complexity of the problem—particularly, the problem of orbit identification—in the worst case. Here, the worst case is among all possible S (which implies all possible $S' \subseteq S$) and all possible $\tilde{X}$ (which implies all possible $X \subseteq \tilde{X}$). Unfortunately, we can show that in the worst case, this problem is *unsolvable*: it is even

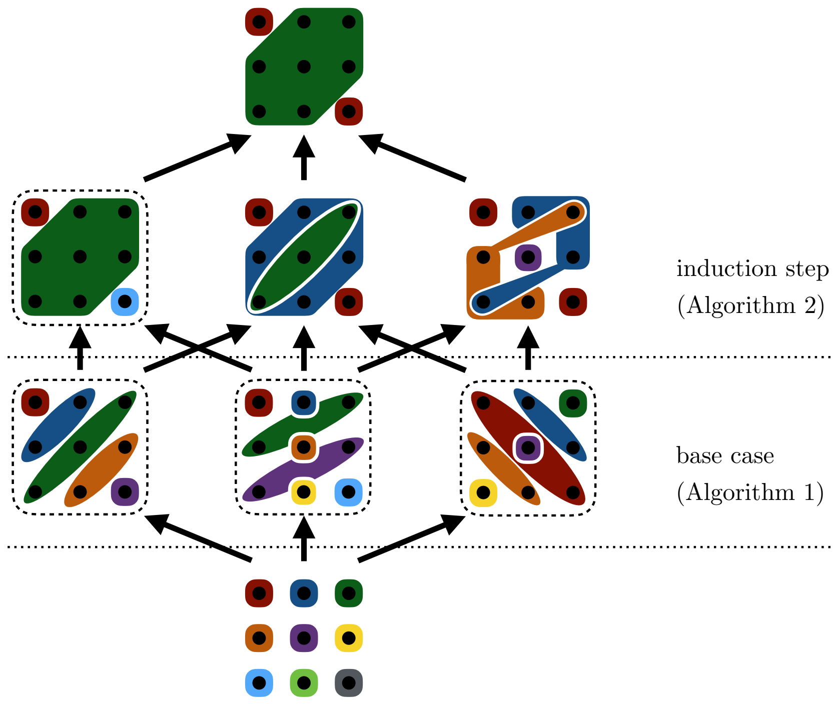


Figure 14: The final output of the induction algorithm run on the toy example in Figure 11. Every directed edge ($\mathcal{A} \to \mathcal{A}'$) denotes the "finer than" relation ($\mathcal{A} \succeq \mathcal{A}'$), since the input partitions of Algorithm 2 are both finer than the output partition. Partitions in the dashed boxes are the same partitions in the dashed boxes from Figures 12 and 13.

harder than the famous *word problem* for groups, and the word problem was shown to be unsolvable (Novikov, 1955; Boone, 1958; Britton, 1958). Our problem is harder because it includes the word problem as a special case. To see this, let $G = \langle U|R \rangle$ be a group presentation, and consider the group action of $\langle\langle R \rangle\rangle$ acting on $F(U)$, where $\langle\langle R \rangle\rangle$ is the normally generated subgroup by R, $F(U)$ is the free group over U, and the group action is the group operation of $F(U)$ restricted to $\langle\langle R \rangle\rangle \times F(U)$. The orbit identification problem for this group action is: given any $w, w' \in F(U)$, decide whether there exists an $r \in \langle\langle R \rangle\rangle$ such that $w' = r \cdot w$. This is equivalent to decide when the two words w, w' (elements of a free group are called *words*) represent the same element in $G = F(U)/\langle\langle R \rangle\rangle$, or equivalently, when $w' \cdot w^{-1} \in \langle\langle R \rangle\rangle$, which is precisely the statement of the word problem. A simple example of $G = \langle U|R \rangle$ with unsolvable word problem can be found in Collins (1986). Although our problem is unsolvable in the worst case, it is solvable in many special cases. For example, our induction algorithm can solve all instances of Problem (2) exactly; when further equipped with the "Expand-and-Restrict" technique (Section 6.1), our algorithm can also solve many instances of Problem (5) either exactly or approximately.

Complexity of the Induction Algorithm. Our induction algorithm always runs Algorithm 1 $|S|$ times for the base case, then runs Algorithm 2 in the induction step $2^{|S|} - |S| - 1$ times. In any run of Algorithm 1, every point in the input space X will be transformed exactly once and be labeled exactly once. Thus, the complexity of Algorithm 1 is always $O(|X|)$ in all runs, yielding a total complexity of $O(|S||X|)$ for the base case. Regarding the complexity of Algorithm 2, we count the total number of merge steps. The outer for-loop has exactly $|\mathcal{Q}|$ iterations, and the number of merge steps in every outer iteration is precisely the number of the corresponding inner iterations. The number of iterations in each inner for-loop is the number of cells in the current $\mathcal{P}$ that intersect with the current $Q \in \mathcal{Q}$, which unfortunately varies from case to case. However, given that the size of $\mathcal{P}$ is monotonically decreasing, we can upper bound this number by the initial $|\mathcal{P}|$. So, the total number of merge steps is upper bounded by $|\mathcal{P}||\mathcal{Q}|$; the complexity of Algorithm 2 is upper bounded by $O(|\mathcal{P}||\mathcal{Q}|)$. Unlike Algorithm 1 which has a fixed complexity for every individual run, the complexity of Algorithm 2 varies from run to run according to the sizes of each run's input partitions. In case a rough estimation is useful, we can further upper bound $O(|\mathcal{P}||\mathcal{Q}|)$ consistently by $O(|X|^2)$ for $\mathcal{P}$, $\mathcal{Q}$ in all runs of Algorithm 2. Therefore, the total complexity of the entire induction algorithm is very loosely upper bounded by $O(|S||X|) + O(2^{|S|}|X|^2)$.

We make two comments about this upper bound. First, the exponential growth with respect to $|S|$ is desired, since we want a rich universe of abstractions. Second, while the first term in the upper bound is tight, the second term with a quadratic dependence on $|X|$ is loose. In practice, we can do (much) better than $|X|^2$. Notably, how loose the bound is depends on the generators in S, the input space X, and where in the hierarchy Algorithm 2 is run. While the first two are given (so we do not have a choice), we often have a choice for $\mathcal{P}$ and $\mathcal{Q}$ when running Algorithm 2, since in Equation (4) in the induction step, we can often have many choices for selecting S'' and S''' especially when S' is large. One obvious strategy is to pick $\mathcal{P} = \Pi(S'')$ and $\mathcal{Q} = \Pi(S''')$ among the candidate set $\{(S'', S''') | S'' \subset S', S''' \subset S', S'' \cup S''' = S'\}$ such that the product $|\mathcal{P}||\mathcal{Q}|$ is minimized.

In summary, the essence of the induction step is to do orbit identification indirectly. However, we cannot get around orbit identification, but always have to do it in some form. Therefore, it is important to notice that while our induction algorithm runs on all cases (including the worst case), this does *not* contradict the fact that the problem in the worst case is unsolvable. This is because the output from running the induction algorithm alone is not always accurate in the general setting (i.e. Problem (5)), especially when considering a group action on an infinite set whose orbits are further to be restricted to a finite subset. As will be detailed in Section 6, we need the Expand-and-Restrict technique to correct errors. In the worst case and when approximations are unacceptable, we need an infinite expand, which is computationally infeasible and agrees with the unsolvability of the problem.

5.3 Finding a Generating Set of $\mathsf{ISO}(\mathbb{Z}^n)$

We give an example of finding a finite generating set. The key idea is based on recursive group decompositions. In light of storing abstractions of a set X in digital computers, we consider the discrete space $X = \mathbb{Z}^n (\subseteq \mathbb{R}^n)$. Further, we restrict our attention to generators that are isometries of $\mathbb{Z}^n$, since $\mathsf{ISO}(\mathbb{Z}^n)$ is finitely generated. We show this by explicitly finding a finite generating set of $\mathsf{ISO}(\mathbb{Z}^n)$.

Recall that (in Section 4.3.2) we presented one of the characterizations of $\mathsf{ISO}(\mathbb{Z}^n)$ as

$$\mathsf{ISO}(\mathbb{Z}^n) = \langle \mathsf{T}(\mathbb{Z}^n) \cup \mathsf{R}(\mathbb{Z}^n) \rangle \quad \text{where } \mathsf{T}(\mathbb{Z}^n) \cap \mathsf{R}(\mathbb{Z}^n) = \{\text{id}\}. \tag{6}$$

We start from this characterization, and seek a generating set of $\mathsf{T}(\mathbb{Z}^n)$ and a generating set of $\mathsf{R}(\mathbb{Z}^n)$. Finding generators of $\mathsf{T}(\mathbb{Z}^n)$ is easy: $\mathsf{T}(\mathbb{Z}^n) = \langle t'_{\boldsymbol{e_1}} \cup \cdots \cup t'_{\boldsymbol{e_n}} \rangle$. However, finding generators of $\mathsf{R}(\mathbb{Z}^n)$ requires more structural inspections. The strategy is to first study the matrix group $\mathsf{O}_n(\mathbb{Z})$ which is isomorphic to $\mathsf{R}(\mathbb{Z}^n)$, and then transfer results to $\mathsf{R}(\mathbb{Z}^n)$. Interestingly, $\mathsf{O}_n(\mathbb{Z})$ has a decomposition similar to what $\mathsf{ISO}(\mathbb{Z}^n)$ has in Equation (6). By definition, $\mathsf{O}_n(\mathbb{Z})$ consists of all orthogonal matrices with integer entries. For any $A \in \mathsf{O}_n(\mathbb{Z})$, the orthogonality and integer-entry constraints restrict every column vector of A to be a unique standard basis vector or its negation. This will lead to the decomposition of $\mathsf{R}(\mathbb{Z}^n)$.

Notations. $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$ is the all-ones vector; $\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}$ are the standard basis vectors of $\mathbb{R}^n$; $\boldsymbol{\nu_1}, \ldots, \boldsymbol{\nu_n}$ are the so-called unit negation vectors of $\mathbb{R}^n$ where $\boldsymbol{\nu_i} \in \{-1, 1\}^n$ has a -1 in the ith coordinate and 1s elsewhere.

Definition 33 (Permutation) *A permutation matrix is a matrix obtained by permuting the rows of an identity matrix; we denote the set of all $n \times n$ permutation matrices by P_n. A permutation of an index set is a bijection $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$; the set of all permutations of the size-n index set is known as the symmetric group S_n. A permutation of (integer-valued) vectors is a rotation $r'_P : \mathbb{Z}^n \to \mathbb{Z}^n$ for some $P \in \mathsf{P}_n$; we denote the set of all permutations of n-dimensional vectors by $\mathsf{R}_\mathsf{P}(\mathbb{Z}^n) \subseteq \mathsf{R}(\mathbb{Z}^n)$.*

Definition 34 (Negation) *A (partial) negation matrix is a diagonal matrix whose diagonal entries are drawn from $\{-1, 1\}$; we denote the set of all $n \times n$ negation matrices by N_n. A (partial) negation of (integer-valued) vectors is a rotation $r'_N : \mathbb{Z}^n \to \mathbb{Z}^n$ for some $N \in \mathsf{N}_n$; we denote the set of all negations of n-dimensional vectors by $\mathsf{R}_\mathsf{N}(\mathbb{Z}^n) \subseteq \mathsf{R}(\mathbb{Z}^n)$.*

Remark 35 *Under Definitions 33 and 34, one can verify that a permutation (of vectors) maps x to Px by permuting x's coordinates according to $P \in \mathsf{P}_n$; likewise, a negation (of vectors) maps x to Nx by negating x's coordinates according to $N \in \mathsf{N}_n$.*

Theorem 36 $(\mathsf{R}_\mathsf{P}(\mathbb{Z}^n), \circ) \cong (\mathsf{P}_n, \cdot) \cong (S_n, \circ)$*;* $(\mathsf{R}_\mathsf{N}(\mathbb{Z}^n), \circ) \cong (\mathsf{N}_n, \cdot)$*. These further imply that $|\mathsf{R}_\mathsf{P}(\mathbb{Z}^n)| = |\mathsf{P}_n| = |S_n| = n!$ and $|\mathsf{R}_\mathsf{N}(\mathbb{Z}^n)| = |\mathsf{N}_n| = 2^n$. (Proof in Appendix B.16.)*

Theorem 37 $\mathsf{O}_n(\mathbb{Z}) = \langle \mathsf{N}_n \cup \mathsf{P}_n \rangle$*, where $\mathsf{N}_n \cap \mathsf{P}_n = \{I\}$. (Proof in Appendix B.17.)*

Corollary 38 $\mathsf{R}(\mathbb{Z}^n) = \langle \mathsf{R}_\mathsf{N}(\mathbb{Z}^n) \cup \mathsf{R}_\mathsf{P}(\mathbb{Z}^n) \rangle$*, where $\mathsf{R}_\mathsf{N}(\mathbb{Z}^n) \cap \mathsf{R}_\mathsf{P}(\mathbb{Z}^n) = \{\mathrm{id}\}$.*

Remark 39 *The decomposition of the rotation group $\mathsf{R}(\mathbb{Z}^n)$ in Corollary 38 has a similar form compared to the decomposition of the isometry group $\mathsf{ISO}(\mathbb{Z}^n)$ in Equation* (6)*. One can show that $\mathsf{R}(\mathbb{Z}^n)$ has a second characterization that is similar to the second characterization of $\mathsf{ISO}(\mathbb{Z}^n)$, where $\mathsf{R}(\mathbb{Z}^n)$ can also be decomposed as semi-direct products: $\mathsf{R}(\mathbb{Z}^n) = \mathsf{R}_\mathsf{N}(\mathbb{Z}^n) \circ \mathsf{R}_\mathsf{P}(\mathbb{Z}^n) \cong \mathsf{N}_n \rtimes \mathsf{P}_n$. Yet, this characterization is not used in this paper, so we omit its proof.*

By Corollary 38, finding generators of $\mathsf{R}(\mathbb{Z}^n)$ breaks down into finding those of $\mathsf{R}_\mathsf{N}(\mathbb{Z}^n)$ and $\mathsf{R}_\mathsf{P}(\mathbb{Z}^n)$, respectively. First, from unit negation vectors, we can find generators for negations:

$$\mathsf{R}_\mathsf{N}(\mathbb{Z}^n) = \langle \{ r'_{\mathrm{diag}(\boldsymbol{\nu_1})}, \ldots, r'_{\mathrm{diag}(\boldsymbol{\nu_n})} \} \rangle.$$

Second, from the fact that the symmetric group is generated by 2-cycles of the form $(i, i+1)$: $S_n = \langle \{(1,2), \ldots, (n-1, n)\} \rangle$ (Conrad, 2016), we can find generators for permutations:

$$\mathsf{R}_\mathsf{P}(\mathbb{Z}^n) = \langle \{ r'_{P(1,2)}, \ldots, r'_{P(n-1,n)} \} \rangle,$$

where $P^{(i,i+1)} \in \mathsf{P}_n$ is obtained by swapping the ith and $(i+1)$th rows of I. Finally,

$$\begin{aligned}\mathsf{ISO}(\mathbb{Z}^n) &= \langle \mathsf{T}(\mathbb{Z}^n) \cup \mathsf{R}(\mathbb{Z}^n)\rangle \\ &= \langle \mathsf{T}(\mathbb{Z}^n) \cup \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n) \cup \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n)\rangle \\ &= \langle \mathsf{T}_0 \cup \mathsf{R}_{\mathsf{N}0} \cup \mathsf{R}_{\mathsf{P}0}\rangle\,, \end{aligned} \tag{7}$$

where $\mathsf{T}_0 := \{t'_{\boldsymbol{e_i}}\}_{i=1}^n$, $\mathsf{R}_{\mathsf{N}0} := \{r'_{\mathrm{diag}(\boldsymbol{\nu_i})}\}_{i=1}^n$, and $\mathsf{R}_{\mathsf{P}0} := \{r'_{P^{(i,i+1)}}\}_{i=1}^{n-1}$. Here, we performed recursive group decompositions to yield the generating set $\mathsf{T}_0 \cup \mathsf{R}_{\mathsf{N}0} \cup \mathsf{R}_{\mathsf{P}0}$ with finite size $n + n + (n-1) = 3n-1$. This verifies that $\mathsf{ISO}(\mathbb{Z}^n)$ is indeed finitely generated.

5.4 Trade-off: Minimality or Diversity (Efficiency or Expressiveness)

A generating set of a group is not unique. There are two extremes when considering the size of a generating set. One considers the *largest* generating set of a group which is the group itself; the other considers a *minimal* generating set which is not unique either.

Definition 40 *Let G be a group, $S \subseteq G$, and $\langle S\rangle$ be the subgroup of G generated by S. We say that S is a minimal generating set (of $\langle S\rangle$) if for any $s \in S$, $\langle S\backslash\{s\}\rangle \neq \langle S\rangle$.*

Theorem 41 *Let G be a group and $S \subseteq G$, then S is a minimal generating set if and only if for any $s \in S$, $s \notin \langle S\backslash\{s\}\rangle$. (Proof in Appendix B.18.)*

Considering $\mathsf{ISO}(\mathbb{Z}^n) = \langle \mathsf{T}_0 \cup \mathsf{R}_{\mathsf{N}0} \cup \mathsf{R}_{\mathsf{P}0}\rangle$, it is easy to check that T_0, $\mathsf{R}_{\mathsf{N}0}$, and $\mathsf{R}_{\mathsf{P}0}$ are minimal individually; whereas their union is not. Nevertheless, it is not hard to show that $S^\star := \{t'_{\boldsymbol{e_1}}, r'_{\mathrm{diag}(\boldsymbol{\nu_1})}\} \cup \mathsf{R}_{\mathsf{P}0}$ is a minimal generating set of $\mathsf{ISO}(\mathbb{Z}^n)$ with size $n+1$.

There is a trade-off between minimality and diversity, further leading to the trade-off between efficiency and expressiveness. Again, use $\mathsf{ISO}(\mathbb{Z}^n)$ as an example. To one extreme, a minimal generating set is most efficient in the following sense: $S \subseteq \mathsf{ISO}(\mathbb{Z}^n)$ is a minimal generating set if and only if $\pi'|_{2^S}$ is injective, i.e. for any $S', S'' \subseteq S$, if $S' \neq S''$, then $\pi'(S') = \langle S'\rangle \neq \langle S''\rangle = \pi'(S'')$ (an easy check). So, whenever S is not minimal, there are duplicates in the generated subgroups, and hence, duplicates in the subsequent abstraction generations. Each duplication is a wasted computation since it does not result in a new abstraction in the end. Intuitively, if a generating set is further away from being minimal, more duplicates tend to occur and the abstraction generating process is less efficient. To the other extreme, the largest generating set is most expressive in the following sense: if $S = \mathsf{ISO}(\mathbb{Z}^n)$, then $\pi'(2^S) = \mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$, i.e. the collection of all subgroups of $\mathsf{ISO}(\mathbb{Z}^n)$; and in general, for any $S \subset S_+ \subseteq \mathsf{ISO}(\mathbb{Z}^n)$, the monotonicity property $\pi'(2^S) \subset \pi'(2^{S_+})$ holds (an easy check). Yet, the largest generating set is also the least efficient not only because it has the largest number of duplicates but also it is infinite. So, regarding the trade-off between efficiency and expressiveness, we need to find a balance between the two extremes.

Our plan is to start from a minimal generating set $S^\star$ of $\mathsf{ISO}(\mathbb{Z}^n)$ and then gradually enlarge it by adding the so-called *derived generators*. In other words, we aim for a filtration: $S^\star \subseteq S^\star_{+1} \subseteq S^\star_{+2} \subseteq S^\star_{+3} \subseteq \cdots$ such that the corresponding collections of subgroups satisfy

$$\pi'(2^{S^\star}) \subseteq \pi'(2^{S^\star_{+1}}) \subseteq \pi'(2^{S^\star_{+2}}) \subseteq \pi'(2^{S^\star_{+3}}) \subseteq \cdots \quad \text{and} \quad \bigcup_{m=1}^{\infty} \pi'(2^{S^\star_{+m}}) = \mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}.$$

Definition 42 *Let $S^\star$ be a minimal generating set of $\mathsf{ISO}(\mathbb{Z}^n)$, and define*

$$S^\star_{+m} := \{s_k^{\alpha_k} \circ \cdots \circ s_1^{\alpha_1} \mid k \in \mathbb{Z}_{\geq 0}, s_k, \ldots, s_1 \in S^\star, \alpha_k, \ldots, \alpha_1 \in \mathbb{Z}, \textstyle\sum_{i=1}^k |\alpha_i| \leq m\}.$$

A derived generator of length m is an $s \in S^\star_{+m} \backslash S^\star_{+(m-1)}$.

Remark 43 *In Definition 42, $S^\star_{+m}$ is the "ball" with center* id *and radius m in the Cayley graph of $S^\star \cup (S^\star)^{-1}$. It is an easy check that $S^\star \cup (S^\star)^{-1} = S^\star_{+1} \subseteq S^\star_{+2} \subseteq S^\star_{+3} \subseteq \cdots$.*

Note that $\cup_{m=1}^\infty S^\star_{+m} = \langle S^\star \rangle = \mathsf{ISO}(\mathbb{Z}^n)$, since the growing "ball" will eventually cover the whole Cayley graph. Therefore, $\cup_{m=1}^\infty \pi'(2^{S^\star_{+m}}) = \mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$. This suggests we gradually add derived generators of increasing length to $S^\star$, and approximate $\mathcal{H}^*_{\mathsf{ISO}(\mathbb{Z}^n)}$ by $\pi'(2^{S^\star_{+m}})$ for some large m. Without any prior preference, one must go through this full procedure to grow the ball $S^\star_{+m}$ from radius $m = 1$. Although computationally intense, it is incremental. More importantly, this is a *one-time* procedure, but the resulting abstraction (semi)universe is *universal*: computed abstractions can be used in different topic domains.

However, like human perception with innate preference for certain stimuli, having prior preference for certain derived generators can make the abstraction (semi)universe grow more efficiently. As an illustration and a design choice, we start from the minimal generating set $S^\star := \{t'_{\boldsymbol{e_1}}, r'_{\mathrm{diag}(\boldsymbol{\nu_1})}\} \cup \mathsf{R_{P0}}$ and prioritize three types of derived generators. We first add back the basis generators $\{t'_{\boldsymbol{e_i}}\}_{i=2}^n$ and $\{r'_{\mathrm{diag}(\boldsymbol{\nu_i})}\}_{i=2}^n$ to get the generating set $\mathsf{T}_0 \cup \mathsf{R_{N0}} \cup \mathsf{R_{P0}}$ from Equation (7). This restores the complete sets of translations, negations, and permutations—the three independent pillars generating $\mathsf{ISO}(\mathbb{Z}^n)$. The other two types, inspired by innate preference for *periodicity* and *synchronization* (VanRullen et al., 2014; Von Der Heydt et al., 1992; Tymoczko, 2010), are *circulators* and *synchronizers* defined below.

Definition 44 *Let $S = \{s_1, \ldots, s_k\}$ be a minimal generating set. A circulator of S with period α is: s^α for some $s \in S$ and $\alpha \in \mathbb{Z}_{>0}$. (Consider group action on X and any $x \in X$: the orbit $\langle s^\alpha \rangle x$ consists of periodic points from $\langle s \rangle x$.) If $\langle S \rangle$ is Abelian, the synchronizer of S is: $s_k \circ \cdots \circ s_1$.*

We denote the set of all circulators of T_0 with a fixed period α by $\mathsf{T}_0^\alpha := \{t'_{\alpha\boldsymbol{e_i}}\}_{i=1}^n$. Inspecting circulators of $\mathsf{R_{N0}}$ and $\mathsf{R_{P0}}$ does not yield new generators, since for any $s \in \mathsf{R_{N0}} \cup \mathsf{R_{P0}}$, $s^2 = \mathrm{id}$. The synchronizers of T_0 and $\mathsf{R_{N0}}$ are $t'_{\boldsymbol{1}}$ and r'_{-I} ($\langle \mathsf{R_{P0}} \rangle$ is not Abelian). Adding these circulators and synchronizers to $\mathsf{T}_0 \cup \mathsf{R_{N0}} \cup \mathsf{R_{P0}}$ yields the generating set:

$$S^\star_+ := \mathsf{T}_0 \cup \mathsf{T}_0^2 \cup \cdots \cup \mathsf{T}_0^\tau \cup \{t'_{\boldsymbol{1}}\} \cup \mathsf{R_{N0}} \cup \{r'_{-I}\} \cup \mathsf{R_{P0}}, \tag{8}$$

where τ denotes an upper bound on period exploration. Notably, $S^\star_+$ is of size $\tau n + 1 + n + 1 + (n-1) = (\tau+2)n + 1$ and is used in music and chemistry applications (Yu et al., 2020).

6. Implementing the General Setting: Restriction to Finite Subspaces

Computers have to work with finite spaces for finite execution time. If the underlying space X is finite, our induction algorithm in Section 5.2 can solve any instances of Problem (2) exactly, regarding any finitely generated subgroup acting on any finite set. If X is infinite (e.g. $\mathbb{Z}^n$), we can still mathematically imagine abstractions of X. However, to compute an abstraction (or just to represent it) in practice, we have to work with finite subspaces of X, e.g. Problem (5). In this setting, we must be careful about both what an abstraction of a subspace means and what potential problems might occur.

Definition 45 *Let X be a set and $\mathcal{P}$ be an abstraction of X. For any $Y \subseteq X$, the restriction of $\mathcal{P}$ to Y is an abstraction of Y given by $\mathcal{P}|_Y := \{P \cap Y \mid P \in \mathcal{P}\} \backslash \{\emptyset\}$.*

Remark 46 *Unless otherwise stated, the term "an abstraction of a subspace" means an abstraction of the ambient space restricted to that subspace. Under this definition, we need extra caution when computing an abstraction of a subspace.*

Let X be a set and $H \leq \mathsf{F}(X)$ be a subgroup of the transformation group of X. For any $Y \subseteq X$, according to Definition 45, the abstraction of the subspace Y from H is:

$$\pi(H)|_Y = (X/H)|_Y = \{Hx \cap Y \mid x \in X\} \backslash \{\emptyset\}.$$

A risky way of computing the above abstraction is focusing on Y while forgetting its ambient space X. Let $\mathcal{R}^H_Y$ be the abstraction of Y obtained by running the induction algorithm on Y (instead of X) in the bottom-up approach. The risk is that $\mathcal{R}^H_Y$ may be strictly finer than $\pi(H)|_Y = \mathcal{R}^H_X|_Y$, i.e. there may be cells in $\mathcal{R}^H_Y$ that should be merged but are not if they are connected via points outside Y. For instance, let $X = \mathbb{Z}^2$, $Y = \{(0,0), (1,0), (0,1), (1,1)\} \subseteq X$, and the subgroup $H = \langle\{t'_{\mathbf{1}}, r'_{-I}\}\rangle \leq \mathsf{ISO}(\mathbb{Z}^2) \leq \mathsf{F}(\mathbb{Z}^2)$. One can check:

$$\begin{aligned} \mathcal{R}^H_Y &= \{\ \{(0,0),(1,1)\},\ \{(0,1)\},\ \{(1,0)\}\ \}; \\ \pi(H)|_Y &= \{\ \{(0,0),(1,1)\},\ \{(0,1),(1,0)\}\ \}. \end{aligned}$$

The two points $(1,0)$ and $(0,1)$ should be in one cell since $(1,0) \xmapsto{r'_{-I}} (-1,0) \xmapsto{t'_{\mathbf{1}}} (0,1)$, but are not in $\mathcal{R}^H_Y$ since the via-point $(-1,0) \notin Y$. In general, the risk is present if we compute an abstraction of a subspace Y from other abstractions of Y or from orbit tracing.

However, for computational reasons, we want to forget the ambient space X! In particular, the risky way is the only practical way if X is infinite and Y is finite, since it is not realistic to identify all orbits in an infinite space. This suggests that we take the risk to generate $\mathcal{R}^H_Y$ as the first step, and rectify the result in a second step to merge cells that are missed in the first step. As a result, we introduce a technique called "expand-and-restrict".

6.1 Expand-and-Restrict

"Expand-and-restrict" is an empirical technique which first expands the subspace and then restricts it back, i.e. to compute $\mathcal{R}^H_{Y_+}|_Y$ for some finite subspace Y_+ such that $Y \subset Y_+ \subset X$. The expansion Y_+ takes more via-points into consideration, so it helps merge cells that are missed in $\mathcal{R}^H_Y$. In practice, we carry out this technique gradually in a sequential manner, which is similar to what we did in enlarging a minimal generating set (cf. Section 5.4). Given an infinite space X and a finite subspace $Y \subset X$, we first construct a filtration

$$Y = Y_{+0} \subset Y_{+1} \subset Y_{+2} \subset \cdots \subset X \quad \text{where } Y_{+k} \text{ is finite } \forall k \in \mathbb{Z}_{\geq 0} \text{ and } \bigcup_{k=0}^{\infty} Y_{+k} = X.$$

We then start a search process for a good expansion Y_{+k}. More specifically, we iteratively compute $\mathcal{R}^{H}_{Y_{+k}}|_Y$ for expansion factors $k = 0, 1, 2, \ldots$ until the results reach a consensus among consecutive iterations. To determine a consensus, theoretically, we need to find the smallest k such that $\mathcal{R}^{H}_{Y_{+k}}|_Y = \mathcal{R}^{H}_{Y_{+k'}}|_Y$ for all $k' > k$, which requires an endless search. In practice, we stop the search whenever $\mathcal{R}^{H}_{Y_{+k}}|_Y = \mathcal{R}^{H}_{Y_{+(k+1)}}|_Y = \cdots = \mathcal{R}^{H}_{Y_{+(k+\Delta k)}}|_Y$ for some positive integer Δk. We call this an *early stop*, whose resulting abstraction $\mathcal{R}^{H}_{Y_{+k}}|_Y$ is an empirical approximation of the true abstraction $\pi(H)|_Y$. Note that without early stopping, we will have the correct result $\pi(H)|_Y = \mathcal{R}^{H}_{X}|_Y$ in the limit of this infinite search process. Therefore, even in cases where the space X is finite, if X is much larger than the subspace Y, this empirical search can be more efficient than computing $\pi(H)|_Y$ directly, since earlier search iterations will be extremely cheap and if an early stop happens early there is a win.

6.2 An Implementation Example

We give an example to illustrate some implementation details on generating abstractions of a finite subspace. In this example, we consider finite subspaces of $X = \mathbb{Z}^n$ to be the centered hypercubes of the form $Y = \mathbb{Z}^n_{[-b,b]}$ where $\mathbb{Z}_{[-b,b]} := \mathbb{Z} \cap [-b, b]$ and $b > 0$ is finite. To construct an abstraction semiuniverse for such a finite hypercube, we adopt the bottom-up approach and pick the generating set to be $S^\star_+$ defined in Equation (8). Taking $S^\star_+$ and $\mathbb{Z}^n_{[-b,b]}$ as inputs, we run the induction algorithm, where both Algorithm 1 (for base cases) and Algorithm 2 (for the induction step) are run on the finite subspace $\mathbb{Z}^n_{[-b,b]}$ instead of the infinite space $\mathbb{Z}^n$. This is the first step which gives abstractions $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n_{[-b,b]}}$ for $S \subseteq S^\star_+$.

As mentioned earlier, for every $S \subseteq S^\star_+$, the correct abstraction should be $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n}|_{\mathbb{Z}^n_{[-b,b]}}$ which is generally not equal to $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n_{[-b,b]}}$. So, we run the "expand-and-restrict" technique as the second step. We first construct a filtration: let $Y_{+k} = \mathbb{Z}^n_{[-b-k,b+k]}$ be a finite expansion of $Y = \mathbb{Z}^n_{[-b,b]}$, then it is clear that $Y = Y_{+0} \subset Y_{+1} \subset Y_{+2} \subset \cdots \subset X$ and $\cup_{k=0}^{\infty} Y_{+k} = X = \mathbb{Z}^n$. We then start the empirical search process and set $\Delta k = 1$ (the most greedy search). This means we will stop the search whenever $\mathcal{R}^{\langle S\rangle}_{Y_{+k}}|_Y = \mathcal{R}^{\langle S\rangle}_{Y_{+(k+1)}}|_Y$, and return the abstraction $\mathcal{R}^{\langle S\rangle}_{Y_{+k}}|_Y = \mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n_{[-b-k,b+k]}}|_{\mathbb{Z}^n_{[-b,b]}}$ as the final result to approximate $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n}|_{\mathbb{Z}^n_{[-b,b]}} = \Pi(S)|_{\mathbb{Z}^n_{[-b,b]}}$.

There are three additional implementation tricks that are special to this example. The first trick applies to cases where the subspace $\mathbb{Z}^n_{[-b,b]}$ is large (i.e. a large b). In these cases, every search iteration in the "expand-and-restrict" technique is expensive and gets more expensive as the search goes. Yet, for $S^\star_+$, we typically have $b \gg \tau$ to reveal strong periodic patterns. So, we run the entire two-step abstraction generating process for $\mathbb{Z}^n_{[-\tau,\tau]}$ instead of $\mathbb{Z}^n_{[-b,b]}$, pretending $\mathbb{Z}^n_{[-\tau,\tau]}$ is the subspace to be abstracted. This yields a faster abstraction process since $\mathbb{Z}^n_{[-\tau,\tau]}$ is smaller than $\mathbb{Z}^n_{[-b,b]}$. The result is an abstraction $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n_{[-\tau-k,\tau+k]}}|_{\mathbb{Z}^n_{[-\tau,\tau]}}$ for some expansion factor k. We reuse this same k and compute $\mathcal{R}^{\langle S\rangle}_{\mathbb{Z}^n_{[-b-k,b+k]}}|_{\mathbb{Z}^n_{[-b,b]}}$ as the final result, which is the only expensive computation. Note that this trick adds an additional empirical approximation, assuming that the same expansion factor k works for both small and large subspaces. While we have not yet found a theoretical guarantee for this assumption, this trick works well in practice and provides huge computational savings.

Note: for some generating subsets $S \subseteq S_{+}^{\star}$, we can prove (so no approximations) that the expansion factor $k = 0$ (no need to expand) or 1. Although this provides theoretical guarantees in certain cases, the tricks used in the current proofs are case-by-case depending on the chosen generators. Thus, before we find a universal way of proving things, we prefer empirical strategies—like the above search process—which work universally in any event.

The second trick considers the subspace to be any general hypercube in $\mathbb{Z}^n$, which is not necessarily square or centered. The trick here is simply to find a minimum centered square hypercube containing the subspace. If the ambient space X happens to be "spatially stationary"—the absolute location of each element in the space is not important but only their relative position matters (e.g. the space of music pitches)—then we find a minimum square hypercube containing the subspace and center it via a translation. Centering is very important and specific to the chosen generating set $S_{+}^{\star}$. This is because $S_{+}^{\star}$ contains only pure translations and pure rotations; and centering square hypercubes makes pure rotations safe: no rotation maps a point in $\mathbb{Z}^n_{[-b,b]}$ outside (one can check that for any $r'_A \in \mathsf{R}(\mathbb{Z}^n)$, $r'_A(\mathbb{Z}^n_{[-b,b]}) = \mathbb{Z}^n_{[-b,b]}$). In practice, centering dramatically decreases the number of miss-merged cells, and makes it safe to choose small Δk for early stopping. This explains why we only consider subspaces of the form $\mathbb{Z}^n_{[-b,b]}$ in the first place, and boldly choose $\Delta k = 1$.

The third trick considers a quick-and-dirty pruning of duplicates in generating a family of partitions, leaving room for larger-period explorations. Without this trick, to generate the partition family $\Pi(2^{S_{+}^{\star}})$, we need $|2^{S_{+}^{\star}}| = 2^{(\tau+2)n+1} = O(2^{\tau})$ computations, which hinders exploration on period τ. However, $S_{+}^{\star}$ is not minimal, so $|\Pi(2^{S_{+}^{\star}})| < |2^{S_{+}^{\star}}|$, suggesting many computations are not needed since they yield the same abstraction. We focus on circulators, where we exclude computations on those $S \in 2^{S_{+}^{\star}}$ containing multiple periods. This reduces the number of computations to $(2^{3n+1} - 2^{2n+1})\tau + 2^{2n+1} = O(\tau)$.

A real run on $\mathbb{Z}^4_{[-12,12]}$ and $\tau = 4$ yields 31232 partitions, where all search processes in "expand-and-restrict" end in at most three iterations. This means in this experiment we only need to expand the subspace by $k = 0$ or 1 for all abstractions in the family.

Lastly, we discuss completing a global hierarchy on an abstraction family $\mathfrak{P}_Y$. A brute-force algorithm requires $O(|\mathfrak{P}_Y|^2)$ comparisons, determining the relation ($\preceq$ or incomparable) for every unordered pair of partitions $\mathcal{P}, \mathcal{Q} \in \mathfrak{P}_Y$. Locally, we run a subroutine `GetRelation(P,Q)` implemented via the contingency table (Hubert and Arabie, 1985) whenever we want to query a pair of partitions. Globally, we use two properties to reduce the number of calls to `GetRelation(P,Q)`: 1) transitivity: for any $\mathcal{P}, \mathcal{P}', \mathcal{P}'' \in \mathfrak{P}_Y$, $\mathcal{P} \preceq \mathcal{P}'$ and $\mathcal{P}' \preceq \mathcal{P}''$ implies $\mathcal{P} \preceq \mathcal{P}''$; 2) dualities in Theorem 6. The final output of our abstraction process is a directed acyclic graph of the abstraction (semi)universe $\Pi(2^{S_{+}^{\star}})$. Similar to the first trick above, in practice it suffices to complete the hierarchy for smaller subspaces like $\mathbb{Z}^n_{[-\tau,\tau]}$, assuming the same hierarchy holds for the actual subspace under consideration.

6.3 Algorithmic Generalization of Atomically Generated Subgroups

Generic orbit computation algorithms from computational group theory (Holt et al., 2005) is not applicable for solving our general Problem (5) due to the possibility of infinity (of the group or the data space or both). An efficient algorithm was developed to solve Problem (5) exactly but was specialized for so-called atomically generated subgroups (Yu et al., 2021). This specific setting restricts the finite generating set S to be *atomic* with respect to the nested semidirect product structure of $\mathsf{ISO}(\mathbb{Z}^n)$. More precisely, S can only contain (pure) translations, negations, or permutations, i.e. $S \subseteq \mathsf{T}(\mathbb{Z}^n) \cup \mathsf{R}_\mathsf{N}(\mathbb{Z}^n) \cup \mathsf{R}_\mathsf{P}(\mathbb{Z}^n)$. For instance, the generating set $S_+^\star$ introduced in Equation (8) is atomic (so is any subset of $S_+^\star$), whereas the singleton $\{t'_\mathbf{1} \circ r'_{-I}\}$ is not. Atomically generated subgroups have the nice property of inheriting the semidirect product structure from $\mathsf{ISO}(\mathbb{Z}^n)$ and hence are also decomposable into translation, negation, and permutation subgroups. Leveraging this special property yielded efficient and exact orbit computations regardless of the shape or location of the input subset X. However, not all subgroups of $\mathsf{ISO}(\mathbb{Z}^n)$ are atomically generated. Compared to the specialized algorithm, our algorithm in this paper is less efficient and not always accurate (requiring the "expand-and-restrict" technique to remedy possible errors near the boundary of X) but is applicable to arbitrary finitely generated subgroups (i.e. no restriction on S).

7. Theoretical Generalization of Information Lattice and Learning

In his 1953 work (Shannon, 1953), C. E. Shannon studied the nature of information beyond just quantifying its amount by entropy. In the specific context of communication problems, he used the term *information element* to denote the nature of information that is invariant under "(language) translations" or different encoding-decoding schemes. He further defined a partial order between information elements, yielding a lattice of information elements, or *information lattice* in short. In this section, we revisit information lattice in a more general setting, casting Shannon's lattice in our abstraction-generation framework without needing to introduce information-theoretic functionals. Besides the theoretical generalization, our formulation further reveals how learning enters the picture, serving as the backend theory of existing applications that discover human-level and human-interpretable rules and concepts from sensory data. In Section 7.1, we start with an overview of Shannon's original work and then reveal it in our formulation in the more general context of computational abstraction. In Section 7.2, we start with an overview of lattice-learning-based knowledge discovery applications (Yu et al., 2016; Yu and Varshney, 2017; Yu et al., 2017) and then formalize the related algorithms in our symmetry-driven abstraction framework.

7.1 Meeting Shannon: the Separation of Clustering and Statistics

We present an overview of Shannon's original work and a follow-up work by Li and Chong (2011) which formalizes Shannon's idea in a more principled way.

Nature of information:

information element   or   random variable?

An *information element* is an equivalence class of random variables (of a common sample space) with respect to the "being-informationally-equivalent" relation, where two random variables are informationally equivalent if they induce the same σ-algebra (of the sample space). Under this definition, the notion of an information element—essentially a probability space—is more abstract than that of a random variable: an information element can be realized by different random variables. The relationship between different but informationally-equivalent random variables and their corresponding information element is analogous to the relationship between different translations (say, English, French, or a code) of a message and the actual content of the message. Since different but faithful translations are viewed as different ways of describing the same information, the information itself is then regarded as the equivalence class of all translations or ways of describing the same information. Therefore, the notion of information element reveals the nature of information.

Group-theoretic interpretation:

$$\text{information lattice} \to \text{partition lattice} \to \text{subgroup lattice } (\to \text{interpretation}). \tag{9}$$

An *information lattice* is a lattice of information elements, where the partial order is defined by $x \leq y \iff H(x|y) = 0$ where H denotes the information entropy. The join of two information elements $x \vee y = x + y$ is called the total information of x and y; the meet of two information elements $x \wedge y = xy$ is called the common information of x and y. By definition, every information element can be uniquely determined by its induced σ-algebra. Also, it is known that every σ-algebra of a countable sample space can be uniquely determined by its generating (via union operation) sample-space-partition. Thus, an information lattice has a one-to-one correspondence to a partition lattice. Further, given a partition of a sample space, Li and Chong (2011) constructed a unique permutation subgroup whose group action on the sample space produces orbits that coincide with the given partition. Therefore, under this specific construction (which also validates our Theorem 5), any partition lattice has a one-to-one correspondence to the constructed subgroup lattice (see Li and Chong, 2011, General Isomorphism Theorem). This yields the above Chain (9) which further achieves group-theoretic interpretations of various information-theoretic results, bringing together information theory and group theory (Chan and Yeung, 2002; Yeung, 2008, chap. 16).

Now we cast the above results into our framework, and reveal the key differences.

Nature of abstraction:

$$\text{clustering} \quad \text{or} \quad \text{classification?}$$

Generalizing Shannon's insight on the nature of information essentially reveals the difference between clustering and classification in machine learning. We can similarly define an equivalence relation on the set of all classifications where two classifications are equivalent if they yield the same set of classes and only differ by class labels. For instance, given a set of animals, classifying them into {fish, amphibians, reptiles, birds, mammals} is equivalent to classifying them into {poisson, amphibiens, reptiles, des oiseaux, mammifères}, where the different class labels are only English and French translations of the same set of animal classes. So, clustering to classification is analogous to information element to random variable; and it is clustering rather than classification that captures the nature of abstraction. This again explains why abstraction is formalized as a clustering problem in this paper.

Partition lattice and information lattice: We summarize connections between a partition lattice and an information lattice in Table 1. The difference is rooted in the *separation of clustering from statistics*: partition lattice, being measure-free, can be viewed as information lattice without probability measure. So, abstraction is a more general concept than information. Abstraction is not specific to communication problems, and in particular, is not attached to stochastic processes or information-theoretic functionals such as entropy.

	Partition lattice	*Information lattice*
element	partition ($\mathcal{P}$);	information element (x);
	clustering ($X, \mathcal{P}$);	probability space (X, Σ, P);
	equiv. class of classifications	equiv. class of random variables
partial order	$\mathcal{P} \preceq \mathcal{Q}$	$x \leq y \iff H(x\|y) = 0$
join	$\mathcal{P} \vee \mathcal{Q}$	$x + y$
meet	$\mathcal{P} \wedge \mathcal{Q}$	xy
metric	undefined	$\rho(x, y) = H(x\|y) + H(y\|x)$

Table 1: Partition lattice and information lattice: the former is not coupled with a measure; whereas the latter is attached to a probability measure where both the partial order and the metric are defined in terms of entropies.

Group-theoretic learning:

$$\text{subgroup lattice} \to \text{partition lattice} \to \text{information lattice } (\to \text{learning}). \qquad (10)$$

The separation of clustering and statistics is important since it allows *interpretable statistical learning*, where *interpretability* is achieved by the explicit construction of a partition lattice and *learning* is achieved by subsequent statistical inference on this lattice. This is more precisely presented in Chain (10) aiming for learning, which at first glance, is merely a reverse process of Chain (9) aiming for re-interpretation. However, the subgroup lattices in both chains are in stark contrast: the subgroups considered in Chain (10) are based on certain symmetries—the underlying mechanism of abstraction—whereas the subgroups considered in Chain (9) are merely (isomorphic) re-statements of the given partitions. In other words, among possibly many subgroups (recall Theorem 4 and 5: π is surjective but not necessarily injective) that generate the same partition, we only pick the one that explains to us the types of symmetries under consideration. The preservation of interpretable symmetries through Chain (10) makes the subsequent learning transparent. Therefore, when abstraction does meet statistics, it will yield interpretable machine learning and knowledge discovery (cf. Section 7.2), which is beyond simply a re-interpretation of known results.

7.2 Theory Behind Knowledge Discovery Applications

The computational abstraction formulation derived in this paper can be cast as the theoretical prelude that formalizes existing lattice-learning methods used in knowledge discovery applications (Yu et al., 2016; Yu and Varshney, 2017; Yu et al., 2017, 2020). In those considered application areas (e.g. music, chemistry, image), domain knowledge is embedded in the so-called *probabilistic rules*. Each rule is in the form of an abstraction of the data space and its attaching probabilistic pattern, e.g. a music piece is largely diatonic, where *diatonic*, or more precisely a diatonic scale, is an abstraction of music pitches into pitches classes (to be formalized below), ignoring octave information. In Shannon's language, each rule is also a music realization of an information element. After hierarchical computational abstraction, rules are automatically extracted via learning algorithms developed in the applications. In what follows, we detail our computational abstraction formulation in the context of music, chemistry, and image recognition.

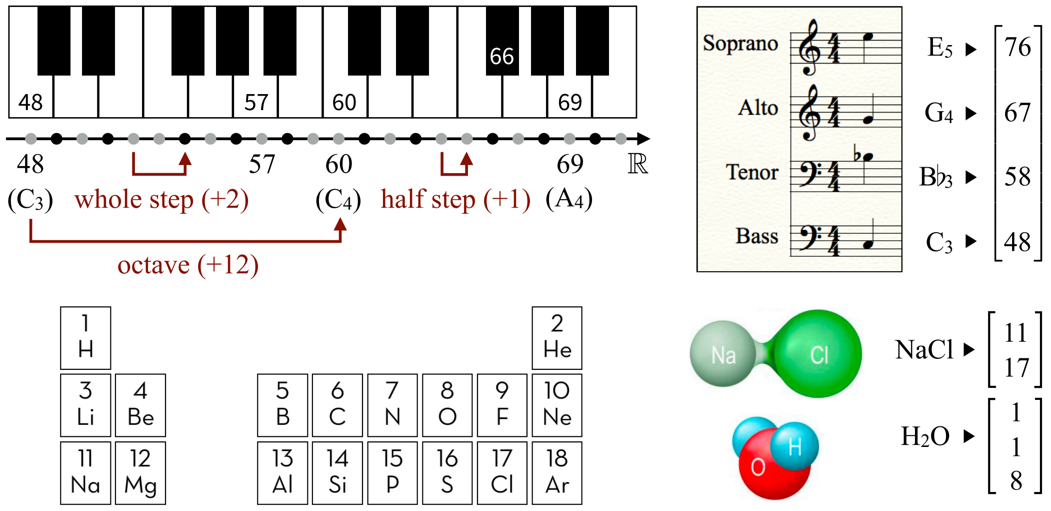


Figure 15: The numerical representation of music pitches (top left) and chords (top right), as well as that of chemical elements (bottom left) and compounds (bottom right).

Computational Music Abstraction. We first consider symmetry-driven partition lattices comprising hierarchical music abstractions. Notably, these music abstractions are not imported from known music definitions, but purely from symmetries. Existing knowledge discovery experiments showed that many abstractions embedded in the lattice coincided with a majority ($\sim 70\%$) of human-developed music-theoretic concepts, while the remaining abstractions suggest new abstraction possibilities that music theorists might have overlooked in the past, e.g. music intervals of intervals, "figured soprano" as a harmonic concept complementary to the classical figured bass (Sokol, 2016).

Music Raw Representation. We represent a music *pitch*—the highness of a music note—by a MIDI number m, which indicates the fundamental frequency f of that pitch but in the following logarithmic scale: $m = 69 + 12\log_2(f/440)$. As a frequency can take any positive value, MIDI numbers are generally real numbers. Yet in our music application, we work with sheet music, and for notes on a music staff, their MIDI numbers are always integers. For instance, the middle C (C_4) is 60, and the tuning pitch (A_4, $f = 440$Hz) is 69. Within the finite range $[21, 108]$, consecutive MIDI integers bijectively map to the 88 consecutive keys on a piano (regardless of being black or white), so two MIDI numbers that are 1, 2, 12 apart form a half step, a whole step, and an octave, respectively (Figure 15: top left).

Building from music pitches, we define a music *chord* by any collection of pitches (that are expected to be heard as a whole), which is naturally represented by a vector of MIDI numbers. More specifically, an n-chord is a chord consisting of n pitches, so the space of all n-chords is $\mathbb{R}^n$. In our music application, we work with the sheet music of Bach's four-voice chorales in particular, so the chord space in this specific context is $\mathbb{Z}^4$ (Figure 15: top right). This infinite space is our input space $X (= \mathbb{Z}^4)$, of which we make abstractions.

Music Abstractions. We study isometries of chords, i.e. $\mathsf{ISO}(\mathbb{Z}^4)$ in our specific context of four-voice chorales from sheet music. We take $S = S^{\star}_{+}$ in Equation (8) as our generating set, and set $\tau = 12$, i.e. considering no more than an octave when exploring periodicity of circulators (a design choice). Thus, the group action under consideration is $\mathsf{ISO}(\mathbb{Z}^4) = \langle S \rangle$ acting on $X = \mathbb{Z}^4$, and the abstraction semiuniverse from this generating set is

$$\Pi(2^S) = \pi \circ \pi'(2^S) = \{\mathbb{Z}^4/\langle S' \rangle \mid S' \subseteq S\}.$$

We give examples of $S' \subseteq S$ whose induced partitions $\Pi(S') = \mathbb{Z}^4/\langle S' \rangle$ subsume many human-invented music terms and concepts.

First, consider $S_1' = \mathsf{T}_0^{12} := \{t'_{12\boldsymbol{e_1}}, t'_{12\boldsymbol{e_2}}, t'_{12\boldsymbol{e_3}}, t'_{12\boldsymbol{e_4}}\}$. The induced abstraction $\Pi(S_1')$ contains 12^4 cells, each of which represents the concept of a *pitch class vector*, i.e. a vector of pitch classes. In music, a *pitch class* is a set of all pitches that are octave(s) apart, e.g. the pitch class C is the set $\{\mathrm{C}_n | n \in \mathbb{Z}\}$, treating a high C, a middle C, a low C indistinguishably as the pitch class C. Therefore, two chords $(\mathrm{B}_5, \mathrm{G}_4, \mathrm{E}_4, \mathrm{C}_3)$ and $(\mathrm{B}_2, \mathrm{G}_3, \mathrm{E}_4, \mathrm{C}_5)$ are in the same partition cell as the pitch class vector $(\mathrm{B}, \mathrm{G}, \mathrm{E}, \mathrm{C})$, whereas the chord $(\mathrm{E}_5, \mathrm{C}_4, \mathrm{A}_3, \mathrm{F}_3)$ is in a different cell as the pitch class vector $(\mathrm{E}, \mathrm{C}, \mathrm{A}, \mathrm{F})$ and notably, the chord $(\mathrm{G}_4, \mathrm{E}_4, \mathrm{C}_4, \mathrm{B}_3)$ is in another different cell as the pitch class vector $(\mathrm{G}, \mathrm{E}, \mathrm{C}, \mathrm{B})$. The point here is that, under this abstraction, the concept of a pitch class vector does not distinguish the register, but as a vector, distinguishes both the order and the multiplicity of its pitch class components.

Second, consider $S_2' = \mathsf{T}_0^{12} \cup \mathsf{R}_{\mathsf{P}0}$, where $\mathsf{R}_{\mathsf{P}0} := \{r'_{P(1,2)}, \ldots, r'_{P(n-1,n)}\}$ is the generating set of the subgroup comprising permutations (of vector coordinates). Compared to $\Pi(S_1')$ in the first example, the induced abstraction $\Pi(S_2')$ here is a coarser partition. In particular, $\Pi(S_2')$ can be obtained by merging cells in $\Pi(S_1')$ that represent pitch class vectors like $(\mathrm{B}, \mathrm{G}, \mathrm{E}, \mathrm{C})$ and $(\mathrm{G}, \mathrm{E}, \mathrm{C}, \mathrm{B})$ but not $(\mathrm{E}, \mathrm{C}, \mathrm{A}, \mathrm{F})$. Therefore, every cell in $\Pi(S_2')$ represents the concept of a *pitch class multi-set*, i.e. a multi-set of pitch classes. Many of these pitch class multi-sets correspond to known music-theoretic terms. For instance, one cell in $\Pi(S_2')$ comprises exclusively all C major seventh chords, musically denoted $\mathrm{CM7} := \{\mathrm{C}, \mathrm{E}, \mathrm{G}, \mathrm{B}\}$; another different cell comprises exclusively all C minor seventh chords, musically denoted $\mathrm{Cm7} := \{\mathrm{C}, \mathrm{E}\flat, \mathrm{G}, \mathrm{B}\flat\}$; a third different cell comprises exclusively all F major seventh chords, musically denoted $\mathrm{FM7} := \{\mathrm{F}, \mathrm{A}, \mathrm{C}, \mathrm{E}\}$; so forth.

Third, consider $S_3' = \mathsf{T}_0^{12} \cup \mathsf{R}_{\mathsf{P}0} \cup \{t'_{\boldsymbol{1}}\}$, where the added translation $t'_{\boldsymbol{1}}$ is the single generator that generates all music transpositions (which form a cyclic subgroup). So, the abstraction $\Pi(S_3')$ is an even coarser partition, compared to the two partitions $\Pi(S_1')$ and $\Pi(S_2')$ in the first and second examples. In particular, $\Pi(S_3')$ can be obtained by merging cells in $\Pi(S_2')$ that represent pitch class multi-sets like $\{\mathrm{C}, \mathrm{E}, \mathrm{G}, \mathrm{B}\}$ and $\{\mathrm{F}, \mathrm{A}, \mathrm{C}, \mathrm{E}\}$ but not $\{\mathrm{C}, \mathrm{E}\flat, \mathrm{G}, \mathrm{B}\flat\}$. Therefore, every cell in $\Pi(S_3')$ represents the concept of a transposition invariant multi-set of pitch classes. As examples from this abstraction, one cell in $\Pi(S_3')$ comprises exclusively all major seventh chords, musically denoted M7; another different cell comprises exclusively all minor seventh chords, musically denoted m7.

To reveal part of the hierarchy, the three examples are picked in a way $(S_1' \subseteq S_2' \subseteq S_3')$ such that they clearly form a sequence of coarser and coarser partitions of our chord space $X = \mathbb{Z}^4$ $(\Pi(S_1') \succeq \Pi(S_2') \succeq \Pi(S_3'))$. This implies that when we further zoom in and inspect things at the level of cells, we can trace out a nested sequence of cells, each of which is from a different partition in the coarsening sequence. The nested cells can be used as sequential descriptions of an element $x \in X$, i.e. a chord in our music example, from more specific descriptions to more general ones. For instance, considering a specific chord $(\mathrm{B}_4, \mathrm{G}_4, \mathrm{E}_4, \mathrm{C}_4)$ which is $(71, 67, 64, 60) \in \mathbb{Z}^4$ and looking at it through a series of nested cells from the coarsening sequence $\Pi(S_1') \succeq \Pi(S_2') \succeq \Pi(S_3')$, we can say that it is a special

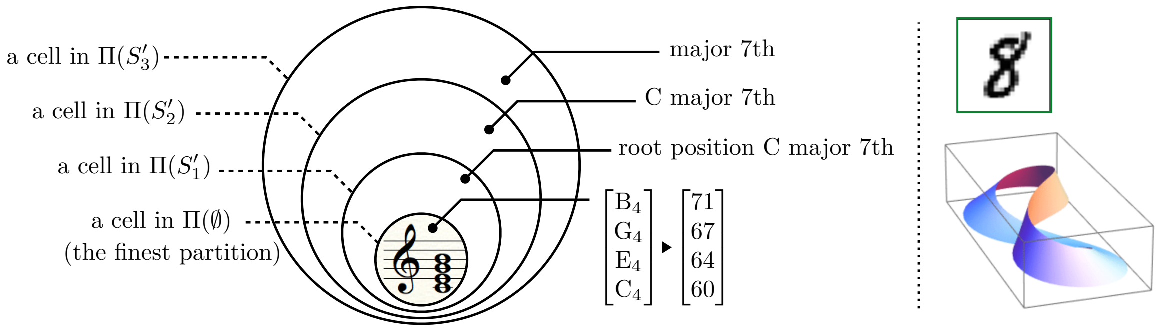

a cell in $\Pi(S_3')$
a cell in $\Pi(S_2')$
a cell in $\Pi(S_1')$
a cell in $\Pi(\emptyset)$
(the finest partition)
major 7th
C major 7th
root position C major 7th
$\begin{bmatrix} B_4 \\ G_4 \\ E_4 \\ C_4 \end{bmatrix} \blacktriangleright \begin{bmatrix} 71 \\ 67 \\ 64 \\ 60 \end{bmatrix}$
8

Figure 16: Abstraction hierarchy in music (left) and abstracting "8" as Möbius strip (right).

type of root position C major seventh chord, and more generally, a C major seventh chord, and even more generally, a major seventh chord (Figure 16). During this sequential process, we are incrementally ignoring the register, the inversion, and the root of a chord one after another, moving toward higher and higher-level music abstractions. On a last note, there are certainly partitions that are incomparable with this line of partitions.

Finally when coming down to real implementations, note that we cannot compute partitions of the infinite chord space $X = \mathbb{Z}^n$; we have to consider a finite subspace $Y \subseteq X$. It is natural to take a hypercube $Y = \mathbb{Z}^4_{[a,b]}$, where the range $[a, b]$ typically denotes the range of a music instrument (from the lowest note a to the highest note b). For instance, in our case regarding chorales, $[a, b]$ denotes the finite human vocal range. Now we can run the induction algorithm (Section 5.2) equipped with the "Expand-and-Restrict" technique (Section 6) for our music application. In particular, the chord space $X = \mathbb{Z}^4$ is "spatially stationary", so all implementation tricks introduced in Section 6.2 are applicable.

It is worth noting that in each of the above three abstractions $\Pi(S_1')$, $\Pi(S_2')$, and $\Pi(S_3')$, there are many cells representing musically unnamed chord types, as well as cells that are traditionally unnamed but were later named in music from more recent eras, e.g. in secundal, quartal harmonies. It is even more important to note that the value of our symmetry-driven abstractions is not just to reproduce known music terms/concepts with a group-theoretic reinterpretation, but to reveal all other traditionally less concerned concepts which yet fall under the same symmetry category. The ability to present a bigger abstraction "map" from various symmetry families/subfamilies can systematically suggest new composition possibilities and guidelines for new music, and the ability to computationally build it allows faster explorations and experimentations in music composition through algorithms.

Computational Chemistry Abstraction. In the chemistry application, raw data were taken as the compound formulas in the molecule database from the Materials Project (Jain et al., 2013). Similar to numerically representing music chords as vectors of their composing pitches in terms of MIDI keys, we represent compound formulas as vectors of their composing elements in terms of atomic numbers (Figure 15: bottom), e.g. NaCl is encoded as $(11, 17)$. So, the data space in the chemistry case is also $\mathbb{Z}^n$ ($n = 1, 2, \ldots$), or its subspaces. Also like music abstractions, similar symmetry-induced partition lattices were used to learn laws in chemistry, e.g. partitioning elements into cells like {H, Li, Na, K, . . . }, {F, Cl, Br, . . . } to rediscover the chemical concept of *groups* as well as partitioning compounds into

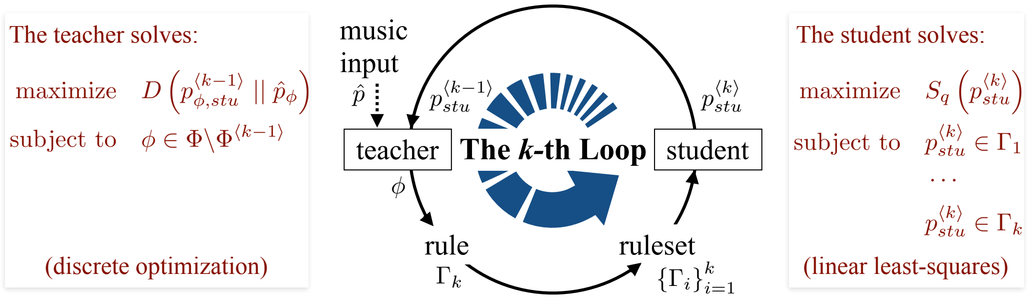

The teacher solves:
maximize $D\left(p_{\phi,stu}^{\langle k-1\rangle} \,||\, \hat{p}_{\phi}\right)$
subject to $\phi \in \Phi \backslash \Phi^{\langle k-1\rangle}$
(discrete optimization)
music input
$\hat{p}$
$p_{stu}^{\langle k-1\rangle}$
teacher
The k-th Loop
student
$p_{stu}^{\langle k\rangle}$
ϕ
rule
Γ_k
ruleset
$\{\Gamma_i\}_{i=1}^{k}$
The student solves:
maximize $S_q\left(p_{stu}^{\langle k\rangle}\right)$
subject to $p_{stu}^{\langle k\rangle} \in \Gamma_1$
...
$p_{stu}^{\langle k\rangle} \in \Gamma_k$
(linear least-squares)

Figure 17: MUS-ROVER's self-learning loop. The teacher distinguishes the student's latest style $p_{stu}^{\langle k-1\rangle}$ from the input style $\hat{p}$ by identifying an abstraction ϕ that best reveals the gap $D(\cdot||\cdot)$. The identified abstraction is turned into a rule and added into the ruleset $\{\Gamma_i\}_{i=1}^{k}$. The student takes the augmented ruleset to update its writing style into $p_{stu}^{\langle k\rangle}$. It favors novelty (by maximizing entropy S_q) subject to the rule constraints. In short, the teacher extracts rules while the student applies rules, both done by solving optimization problems.

cells like {HCl, NaCl, NaF, . . . } to reveal chemical reaction patterns and nomenclature. Existing experiments discovered that music theory and chemistry laws share many abstractions induced from similar types of symmetries (Yu et al., 2020). For instance, the notions of music pitches and chords were shown as in parallel to those of chemical elements and compounds. Moreover, the abstraction of music pitches into pitch classes were in parallel to that of chemical elements into chemical groups, which further results in rules of music scales being in parallel to those of "cation scale" and "anion scale" in chemistry.

Computational Image Abstraction. Digital images viewed as painting pixel intensity on a pixel grid can also be studied under our computational abstraction framework to reveal intrinsic symmetry patterns. One can come up with different abstractions of the 2D grid (a subspace of $\mathbb{Z}^2$) as different ways of segmenting pixels into super pixels. Here, a super pixel means a partition cell in our abstraction framework, which is generalized to be any subset of raw pixels and is not necessarily one single, connected piece. Existing experiments showed that images of handwritten "8" from the MNIST dataset exhibited more symmetries compared to other digits. In particular, our computational abstraction was able to explicitly relate the symmetry in figure "8" to that in a Möbius strip strip (Figure 16).

Learning in Lattices. Besides formalizing abstractions in specific topic domains, we can further formalize the learning process used in the referenced applications (e.g. music, chemistry, images) under our computational abstraction framework. The learning process learns domain-relevant abstractions as *probabilistic rules* to discover domain knowledge. Based on the separation principle (Section 7.1), rule learning is done by operating data statistics on partition lattices, or more precisely, by iteratively projecting raw data distributions to produce probabilistic rules as *abstracted distributions* and inversely, lifting rules to reconstruct raw data distributions for evaluation. Below, we use the learning algorithm from the music application as an example and formalize it in our computational abstraction framework.

Designed as an automatic music theorist, the MUS-ROVER system was developed to learn music-composition rules from symbolic music datasets and further offer people personalized composition lessons (Yu et al., 2016; Yu and Varshney, 2017; Yu et al., 2017). Rules were learned from a "teacher $\rightleftharpoons$ student" model (Figure 17). This model was realized by a self-learning loop between a *discriminative* component (teacher) and a *generative* component (student), where both entities cooperated to iterate through the rule-learning process. The student starts as a *tabula rasa* that picks pitches uniformly at random to form chords and chord progressions. In each iteration, the teacher compares the student's writing style (represented by a probabilistic model) with the input style (represented by empirical statistics) to identify one music abstraction (represented by a partition of the chord space from our aforementioned computational music abstraction) that best reveals the gap between the two styles; and then make it a rule for the student. Consequently, the student becomes less and less random by obeying more and more rules, and thus, approaches the input style. From its rule-learning process on a dataset consisting of Bach's chorales, MUS-ROVER successfully recovered many known rules such as "parallel perfect octaves/fifths are rare" and "tritones are often resolved either inward or outward"; MUS-ROVER also suggests new probabilistic patterns on new music abstractions such as intervals of intervals, figured soprano, and the harmonic counterpart of escape/changing tones.

In MUS-ROVER's self-learning loop, the teacher and the student perform rule extraction and realization, respectively. For the student, the rule realization problem is about finding the most random probability distribution over the chord space (i.e. maximizing novelty) as long as it satisfies all the probabilistic rules. This is formalized by the optimization problem:

$$\text{maximize} \;\; S_q(x) \qquad\qquad \text{subject to} \;\; x \in \Gamma_k, \; k = 1, \dots, K,$$

where the optimization variable x denotes the probability distribution over the chord space (note: we pre-specify a finite range of the pitches under consideration, e.g. piano range, vocal range, so the chord space is finite and x is a vector), the objective $S_q(x)$ is the Tsallis entropy of x measuring the randomness of x, and the constraint sets $\Gamma_1, \dots, \Gamma_K$ denote K rules learned so far. We mention two facts here. First, in the limit as $q \to 1$, $S_q(x) \to H(x)$ which is the Shannon entropy. Second, $x \in \Gamma_k$ is more explicitly represented as a linear equation $A^k x = y^k$ where the pair (A^k, y^k) denotes the kth rule. More specifically, A^k is a boolean matrix (called a partition matrix) which stores the full information of a partition: $A^k_{ij} = 1$ if and only if the jth chord belongs to the ith partition cell; y^k is a probability distribution over the partition cells. We slightly overload the notations and let $\boldsymbol{x}$ and $\boldsymbol{y^k}$ be the information elements that represent the probability space $(X, \Sigma(I), x)$ and $(X, \Sigma(A^k), y^k)$, respectively. In this notation, the sample space X is the chord space and, for a partition matrix P, $\Sigma(P)$ denotes the σ-algebra generated by the partition represented by P (so $\Sigma(I)$ denotes the σ-algebra generated by the finest partition). Under this setting, the equality constraint $A^k x = y^k$ becomes $H(\boldsymbol{y^k} | \boldsymbol{x}) = 0$, i.e. $\boldsymbol{y^k}$ is an abstraction of $\boldsymbol{x}$ as information elements or $\boldsymbol{y^k} \le \boldsymbol{x}$ by Shannon's definition. Therefore, the student's optimization problem for rule realization can be rewritten as follows:

$$\text{maximize} \;\; H(\boldsymbol{x}) \qquad\qquad \text{subject to} \;\; H(\boldsymbol{y^k} | \boldsymbol{x}) = 0, \; k = 1, \dots, K,$$

which describes an inference problem in an information lattice. In words, this means that we want to find an information lattice for the student (used as its mental model) such that it agrees on all K abstractions from the information lattice for the dataset and meanwhile achieves the largest randomness in the chord space. As a result, a good student

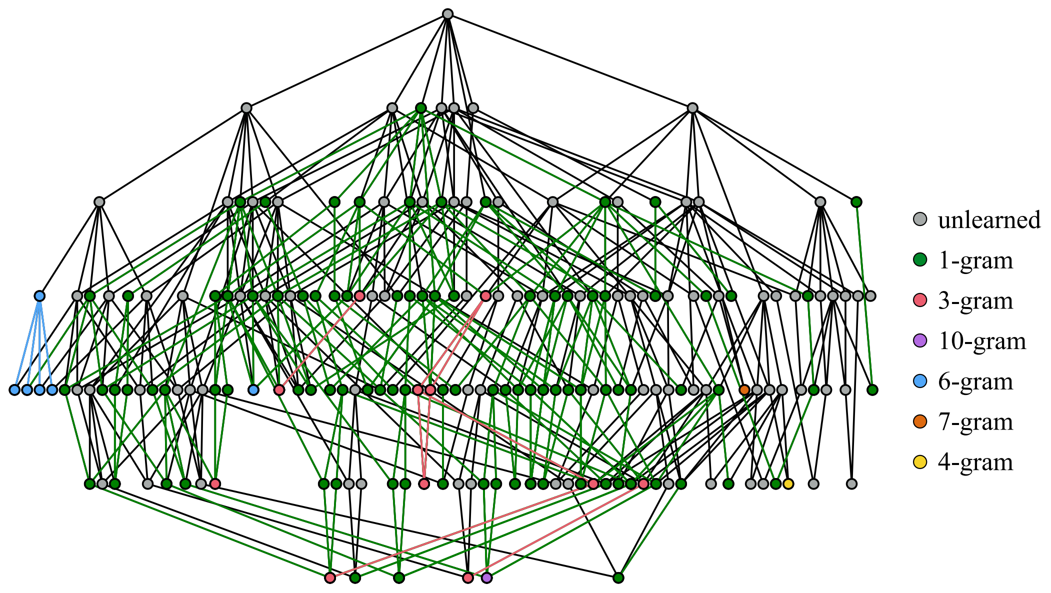


Figure 18: Visualization of Bach's mind for writing chorales. The underlying directed acyclic graph signifies an upside-down information lattice. (Note: edges are oriented upwards according to the convention of a partition lattice; the coarsest partition at the bottom is omitted.) Colors are used to differentiate rule activations from different n-gram settings.

memorizes high-level principles—rules in terms of high-level abstractions and their statistical patterns—in music composition rather than the actual pieces. Indeed the student is encouraged to be as creative as possible as long as the high-level principles are satisfied.

Since learning an information lattice requires both the construction of an abstraction (semi)universe and statistical inference from data, the learning paradigm in MUS-ROVER differs from pure rule-based systems or pure data-driven models, creating a middle ground between the two. This mimics how babies learn from both empirical experience and biological instincts (Hutson, 2018). In MUS-ROVER, the constructed abstraction semiuniverse resembles biological instincts on perceiving sound, and the statistical inference in this semiuniverse resembles experiential learning developed from the instincts. So, the entire concept learning process is transparent, in contrast with black-box algorithms. When running MUS-ROVER on Bach's four-voice chorales, the learning process in the information lattice can be visualized as mimicking Bach's mental activities during chorale composition (Figure 18).

Discussions on continuous structure and data noise. In this paper, we only consider groups with discrete structures like $\mathsf{ISO}(\mathbb{Z}^n)$; moreover, we only consider discrete data spaces like $\mathbb{Z}^n$, i.e. symbolic domains such as sheet music and chemical formulas. For symbolic data, learning in partition lattices is robust to small amounts of random noise due to the way probabilistic rules are defined therein and due to the continuity of information functionals such as entropy. If there are consistent distortions in the dataset rather than random noise, however, these distortions may well be learned as rules. For example, consider learning the music concept of a C major triad made of C-E-G. If 3 out of 100 C-E-Gs were mistakenly transcribed as C-F-G, or D-E-G, or the like (i.e. occasional random noise), the dominance of

C-E-G in probability is not affected by these fluctuations and may still be outputted by the algorithm as the most important chord e.g. in a corpus or in a specific chord progression. On the other hand, if the majority of C-E-Gs were all mistakenly transcribed as C-F-G (i.e. consistent noise), the learning algorithm will instead learn C-F-G as the "C major chord". Notably, symbolic music pitches represent a particular quantization of note frequencies, e.g. equal temperament. Dealing with continuous data directly like working with audio signals would require additional data cleaning techniques and quantization can be one of them. It remains future work to further study continuous groups acting on continuous data spaces.

Acknowledgments

The authors would like to thank the editor and reviewers for their great feedback. This work was funded in part by the IBM-Illinois Center for Cognitive Computing Systems Research (C3SR), a research collaboration as part of the IBM AI Horizons Network; in part by grant number 2018-182794 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation; and in part by the Well-being for Planet Earth Foundation.

References

Sourya Basu, Akshayaa Magesh, Harshit Yadav, and Lav R Varshney. Autoequivariant network search via group decomposition. arXiv:2104.04848 [cs.LG], 2021.

Henda Bélai and Ali Jaoua. Abstraction of objects by conceptual clustering. *Inf. Sci.*, 109 (1-4):79–94, 1998.

Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1): 1–127, 2009.

Ludwig Bieberbach. Über die bewegungsgruppen der euklidischen räume. *Math. Ann.*, 70 (3):297–336, 1911.

Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychol. Rev.*, 94(2):115, 1987.

William W Boone. The word problem. *Proc. Natl. Acad. Sci. U.S.A.*, 44(10):1061–1065, 1958.

Nicolas Bredeche, Zhongzhi Shi, and Jean-Daniel Zucker. Perceptual learning and abstraction in machine learning: an application to autonomous robotics. *IEEE Trans. Syst., Man, Cybern. C*, 36(2):172–181, 2006.

John L Britton. The word problem for groups. *Proc. Lond. Math. Soc.*, 3(4):493–506, 1958.

Alan Bundy, Fausto Giunchiglia, and Toby Walsh. *Building Abstractions.* University of Edinburgh, Department of Artificial Intelligence, 1990.

Terence Ho Leung Chan and Raymond Yeung. On a relation between information inequalities and group theory. *IEEE Trans. Inf. Theory*, 48(7):1992–1995, 2002.

Leonard S Charlap. *Bieberbach Groups and Flat Manifolds.* Springer Science & Business Media, 2012.

Donald J Collins. A simple presentation of a group with unsolvable word problem. *Ill. J. Math.*, 30(2):230–234, 1986.

Keith Conrad. Generating sets. `http://www.math.uconn.edu/~kconrad/blurbs/grouptheory/genset.pdf`, 2016.

Richard M Cormack. A review of classification. *J. R. Stat. Soc. Ser. A. Gen.*, 134(3): 321–367, 1971.

Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with Lie algebra convolutional network. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS 2021)*, pages 2503–2515, 2021.

Thomas G Dietterich. Reflections on innateness in machine learning. `https://medium.com/@tdietterich/reflections-on-innateness-in-machine-learning-4eebefa3e1af`, 2018.

Bettina Eick and Bernd Souvignier. Algorithms for crystallographic groups. *Int. J. Quantum Chem.*, 106(1):316–343, 2006.

Volkmar Felsch and Franz Gähler. CrystCat-a library of crystallographic groups. *A Refereed Gap*, 4, 2000.

Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2):139–172, 1987.

Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artif. Intell.*, 57(2-3):323–389, 1992.

Rebecca L Gómez and Laura Lakusta. A first step in form-based category abstraction by 12-month-old infants. *Developmental Sci.*, 7(5):567–580, 2004.

Derek F Holt, Bettina Eick, and Eamonn A O'Brien. *Handbook of Computational Group Theory.* Chapman and Hall/CRC, 2005.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *J. Classif.*, 2(1):193–218, 1985.

Matthew Hutson. How researchers are teaching AI to learn like a child. `http://www.sciencemag.org/news/2018/05/how-researchers-are-teaching-ai-learn-child`, 2018.

Anil K Jain and Richard C Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

Leonard Kaufman and Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344. John Wiley & Sons, 2009.

Risi Kondor. *Group Theoretical Methods in Machine Learning.* Columbia University, 2008.

Eyal Krupka and Naftali Tishby. Incorporating prior knowledge on features into learning. In *Proc. 10th Int. Conf. Artif. Intell. Stat. (AISTATS 2007)*, pages 227–234, 2007.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

David Lewin. *Generalized Musical Intervals and Transformations.* Oxford University Press, 2010.

Hua Li and Edwin KP Chong. On a connection between information and group lattices. *Entropy*, 13(3):683–708, 2011.

Kenneth R Livingston. *Rationality and the Psychology of Abstraction.* Institute for Objectivist Studies, 1998.

C Neil Macrae and Galen V Bodenhausen. Social cognition: Thinking categorically about others. *Annu. Rev. Psychol.*, 51(1):93–120, 2000.

Jean M Mandler. Perceptual and conceptual processes in infancy. *J. Cogn. Develop.*, 1(1): 3–36, 2000.

Gary Marcus. Innateness, AlphaZero, and artificial intelligence. arXiv:1801.05667 [cs.AI], 2018.

Ryszard S Michalski and Robert E Stepp. Learning from observation: Conceptual clustering. *Mach. Learn.*, 1:331–363, 1983.

Petr Sergeevich Novikov. On the algorithmic unsolvability of the word problem in group theory. *Proc. Steklov. Inst. Math.*, pages 3–143, 1955.

Steve Y Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*, volume 209. American Mathematical Society Providence, RI, 2015.

Rajat Raina, Andrew Y Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proc. 23rd Int. Conf. Mach. Learn. (ICML 2006)*, pages 713–720, 2006.

Ashwin Ram and Eric K Jones. *Foundations of Foundations of Artificial Intelligence.* Department of Computer Science, Victoria University of Wellington, 1994.

Ravi K Raman and Lav R Varshney. Universal clustering. In Yonina Eldar and Miguel Rodrigues, editors, *Information-Theoretic Methods in Data Science.* Cambridge University Press, 2019.

William M Rand. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.*, 66(336):846–850, 1971.

Lior Rokach and Oded Maimon. Clustering methods. In Lior Rokach and Oded Maimon, editors, *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

David W Romero and Suhas Lohit. Learning equivariances and partial equivariances from data. arXiv:2110.10211 [cs.CV], 2021.

Lorenza Saitta and Jean-Daniel Zucker. Semantic abstraction for concept representation and learning. In *Proc. Symp. Abstr., Reformul. and Approx.*, pages 103–120, 1998.

Lorenza Saitta and Jean-Daniel Zucker. *Abstraction in Artificial Intelligence and Complex Systems*. Springer, 2013.

Claude Shannon. The lattice theory of information. *Trans. IRE Prof. Group Inf. Theory*, 1(1):105–107, 1953.

Mina Sheikhalishahi, Mohamed Mejri, and Nadia Tawbi. On the abstraction of a categorical clustering algorithm. In *Proc. 12th Int. Conf. Mach. Learn. Data Min. (MLDM 2016)*, pages 659–675, 2016.

Casey Sokol. Figured soprano. `https://caseysokol.com/?page_id=1067`, 2016.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental Sci.*, 10(1): 89–96, 2007.

The GAP Group. GAP – Groups, Algorithms, and Programming, Version 4.9.1. `https://www.gap-system.org`, 2018.

Dmitri Tymoczko. *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford University Press, 2010.

Rufin VanRullen, Benedikt Zoefel, and Barkin Ilhan. On the cyclic nature of perception in vision versus audition. *Phil. Trans. R. Soc. B*, 369(1641):20130214, 2014.

Rüdiger Von Der Heydt, Esther Peterhans, and MR Dursteler. Periodic-pattern-selective cells in monkey visual cortex. *J. Neurosci.*, 12(4):1416–1434, 1992.

Ulrike Von Luxburg, Robert C Williamson, and Isabelle Guyon. Clustering: Science or art? In *Proc. 2012 ICML Workshop Unsuperv. Transf. Learn. (UTL 2012)*, pages 65–79, 2012.

Julius R Weinberg. *Abstraction, Relation, and Induction: Three Essays in the History of Thought*. University of Wisconsin Press, 1968.

Raymond W Yeung. *Information Theory and Network Coding*. Springer Science & Business Media, 2008.

Haizi Yu and Lav R. Varshney. Towards deep interpretability (MUS-ROVER II): Learning hierarchical representations of tonal music. In *Proc. 5th Int. Conf. Learn. Represent. (ICLR 2017)*, 2017.

Haizi Yu, Lav R Varshney, Guy E Garnett, and Ranjitha Kumar. MUS-ROVER: A self-learning system for musical compositional rules. In *Proc. 4th Int. Workshop Music. Metacreation (MUME 2016)*, 2016.

Haizi Yu, Tianxi Li, and Lav R Varshney. Probabilistic rule realization and selection. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS 2017)*, pages 1561–1571, 2017.

Haizi Yu, James A Evans, and Lav R Varshney. Mimicking human minds via information lattices. In *Proc. 2020 NeurIPS Workshop Babymind*, 2020.

Haizi Yu, Igor Mineyev, and Lav R Varshney. Orbit computation for atomically generated subgroups of isometries of $\mathbb{Z}^n$. *SIAM Journal on Applied Algebra and Geometry*, 5(3): 479–505, 2021.

Ting Yu, Tony Jan, Simeon Simoff, and John Debenham. Incorporating prior domain knowledge into inductive machine learning. *University of Technology, Sydney*, 2007.

Hans Zassenhaus. Über einen algorithmus zur bestimmung der raumgruppen. *Comm. Math. Helv.*, 21(1):117–141, 1948.

Jean-Daniel Zucker. A grounded theory of abstraction in artificial intelligence. *Phil. Trans. R. Soc. B*, 358(1435):1293–1309, 2003.

Appendix A. Mathematical Preliminaries

A.1 For Section 3.1

A *partition* $\mathcal{P}$ of a set X is a collection of mutually disjoint non-empty subsets of X whose union is X. Elements in $\mathcal{P}$ are called *cells* (or less formally, *clusters*); the size of $\mathcal{P}$ is $|\mathcal{P}|$, i.e. the number of cells in $\mathcal{P}$. An *equivalence relation* on a set X, denoted $\sim$, is a binary relation satisfying reflexivity, symmetry, and transitivity. An equivalence relation $\sim$ on X induces a partition of X: $\mathcal{P} = X/{\sim} := \{[x] \mid x \in X\}$, where the quotient $X/{\sim}$ is the set of *equivalence classes* $[x] := \{x' \in X \mid x \sim x'\}$. Conversely, a partition $\mathcal{P}$ of X also induces an equivalence relation $\sim$ on X: $x \sim x'$ if and only if x, x' are in the same cell in $\mathcal{P}$.

A.2 For Section 3.2

A *partial order* is a binary relation that satisfies reflexivity, antisymmetry, and transitivity. A *lattice* is a partially ordered set (or a *poset*) in which every pair of elements has a unique supremum (i.e. least upper bound) called the *join* and a unique infimum (i.e. greatest lower bound) called the *meet*. For any pair of elements p, q in a lattice, we denote their join and meet by $p \vee q$ and $p \wedge q$, respectively. A *sublattice* is a nonempty subset of a lattice, which is closed under join and meet. A *join-semilattice* (resp. *meet-semilattice*) is a poset in which every pair of elements has a join (resp. meet). So, a lattice is both a join-semilattice and a meet-semilattice. A lattice is *bounded* if it has a greatest element and a least element.

A.3 For Section 3.3

A *group* is a pair $(G, *)$ where G is a set and $* : G \times G \to G$ is a binary operation satisfying the *group axioms*: associativity, the existence of identity (denoted e), and the existence of inverse. We also directly say that G is a group, whenever the group operation is understood. Given a group $(G, *)$, a subset $H \subseteq G$ is a *subgroup*, denoted $H \leq G$, if $(H, *)$ is a group. The singleton $\{e\}$ is a subgroup of any group, called the *trivial group*. Given a group G and a subset $S \subseteq G$, the subgroup (of G) generated by S, denoted $\langle S \rangle$, is the smallest subgroup of G containing S; equivalently, $\langle S \rangle$ is the set of all finite products of elements in $S \cup S^{-1}$ where $S^{-1} := \{s^{-1} \mid s \in S\}$. The subgroup generated by a singleton $S = \{s\}$ is called a *cyclic group*; for simplicity, we also call it the subgroup generated by s (an element), denoted $\langle s \rangle$. Let G be a group and X be a set, then a *group action* of G on X (or G-action on X) is a function $\cdot : G \times X \to X$ that satisfies identity ($e \cdot x = x, \forall x \in X$) and compatibility ($g \cdot (h \cdot x) = (g * h) \cdot x, \forall g, h \in G, \forall x \in X$). In this paper, we adopt by default the multiplicative notation for group operations and actions, in which $\cdot$ or $*$ or both may be omitted. For any G-action on X, the *orbit* of x under G is the set $Gx := \{g \cdot x \mid g \in G\}$, and the *quotient* of X by G-action is the set consisting of all orbits $X/G := \{Gx \mid x \in X\}$.

A.4 For Section 3.4

Let G be a group. We use $\mathcal{H}_G^*$ to denote the collection of all subgroups of G. The binary relation "a subgroup of" on $\mathcal{H}_G^*$, denoted $\leq$, is a partial order. $(\mathcal{H}_G^*, \leq)$ is a lattice, called the *lattice of all subgroup* of G, or the *(complete) subgroup lattice* for G in short. For any pair of subgroups $A, B \in \mathcal{H}_G^*$, the join $A \vee B = \langle A \cup B \rangle$ is the smallest subgroup containing A and B; the meet $A \wedge B = A \cap B$ is the largest subgroup contained in A and B.

A.5 For Section 3.5

Let G be a group. We say that two elements $a, b \in G$ are *conjugate* to each other, if there exists a $g \in G$ such that $b = gag^{-1}$, and two subsets $A, B \subseteq G$ are *conjugate* to each other, if there exists a $g \in G$ such that $B = gAg^{-1}$. In either case, conjugacy is an equivalence relation on G (resp. 2^G, i.e. the power set of G), where the equivalence class of $a \in G$ (resp. $A \subseteq G$) is called the *conjugacy class* of a (resp. A). In particular, we can restrict the above equivalence relation on 2^G to the collection of all subgroups $\mathcal{H}_G^*$ which is a subset of 2^G.

A.6 For Section 4

Let $(G, *)$ and $(H, \cdot)$ be two groups. A function $\phi : G \to H$ is called a *homomorphsim* if $\phi(a * b) = \phi(a) \cdot \phi(b)$ for all $a, b \in G$. An *isomorphism* is a bijective homomorphism. We say two groups G and H are *homomorphic* if there exists a homomorphism $\phi : G \to H$, and say they are *isomorphic*, denoted $G \cong H$, if there exists an isomorphism $\phi : G \to H$. Let S be a subset of a group G; then $\mathrm{N}_G(S) := \{g \in G \mid gSg^{-1} = S\}$ is called the *normalizer* of S in G, which is a subgroup of G. We say a subset $T \subseteq G$ *normalizes* another subset $S \subseteq G$ if $T \subseteq \mathrm{N}_G(S)$. We say a subgroup N of a group G is a *normal subgroup* of G, denoted $N \trianglelefteq G$, if G normalizes N, i.e. $G = \mathrm{N}_G(N)$. Let G be a group, $N \trianglelefteq G$, $H \leq G$, $N \cap H = \{e\}$, and $G = NH$; then NH is the *inner semi-direct product* of N and H, and $N \rtimes H$ is the *outer semi-direct product* of N and H. The outer semi-direct product $N \rtimes H$ is the group of all ordered pairs $(n, h) \in N \times H$ with group operation defined by $(n, h)(n', h') = (nhn'h^{-1}, hh')$. The inner and outer semi-direct products are isomorphic, i.e. $NH \cong N \rtimes H$. The semi-direct product equation $G = NH$ gives a decomposition of G into "nearly non-overlapping" (i.e. with trivial intersection) subgroups; moreover, for any $g \in G$, these exist a unique $n \in N$ and $h \in H$ such that $g = nh$.

Appendix B. Mathematical Proofs

B.1 Theorem 4

Proof Let $X = \{1, 2, 3, 4\}$ and $h = (1234), g = (1324) \in S_4 = \mathsf{F}(X)$ be two transformations (also known as permutations, in the cycle notation) of X; consider the cyclic groups:

$$H = \langle h \rangle := \{h^n \mid n \in \mathbb{Z}\} = \{\mathrm{id}, h, h^2, h^3\} = \{\mathrm{id}, (1234), (13)(24), (1432)\};$$
$$G = \langle g \rangle := \{g^n \mid n \in \mathbb{Z}\} = \{\mathrm{id}, g, g^2, g^3\} = \{\mathrm{id}, (1324), (12)(34), (1423)\}.$$

It is clear that $H \neq G$ but $\pi(H) = \pi(G) = \{\{1, 2, 3, 4\}\}$, the coarsest partition of X. ■

B.2 Theorem 5

Proof For any $a, b \in X$, let $f_{a,b} : X \to X$ be the bijective function of the form

$$f_{a,b}(x) = \begin{cases} a & x = b, \\ b & x = a, \\ x & \textit{otherwise.} \end{cases}$$

Pick any partition $\mathcal{P} \in \mathfrak{P}_X^*$. For any cell $P \in \mathcal{P}$, define

$$S_P := \{f_{a,b} \mid a, b \in P, a \neq b\}, \qquad \text{and let } H := \left\langle \bigcup_{P \in \mathcal{P}} S_P \right\rangle.$$

We claim $\pi(H) = \mathcal{P}$. To see this, for any distinct $x, x' \in X$ that are in the same cell in $\mathcal{P}$, $f_{x,x'} \in S_P$ for some $P \in \mathcal{P}$, so $f_{x,x'} \in H$. This implies that x and x' are in the same orbit in $\pi(H)$, since $x' = f_{x,x'}(x)$. Therefore, $\pi(H) \preceq \mathcal{P}$. Conversely, for any distinct $x, x' \in X$ that are in the same orbit in $\pi(H)$, there exists an $h \in H$ such that $x' = h(x)$. By definition, $h = h_k \circ \cdots \circ h_1$ for some finite integer $k > 0$ where $h_k, \ldots, h_1 \in \cup_{P \in \mathcal{P}} S_P$. Suppose $P' \in \mathcal{P}$ is the cell that x is in, i.e. $x \in P'$, then $h_1(x) \in P'$, since $h_1(x) \in P'$ if $h_1 \in S_{P'}$ and $h_1(x) = x$ otherwise. Likewise, we have $h_2 \circ h_1(x), h_3 \circ h_2 \circ h_1(x), \ldots, h_k \circ \cdots \circ h_1(x) \in P'$. This implies that $x' = h(x) = h_k \circ \cdots \circ h_1(x) \in P'$, i.e. x and x' are in the same cell in $\mathcal{P}$. Therefore, $\mathcal{P} \preceq \pi(H)$. Combining both directions yields $\pi(H) = \mathcal{P}$, so π is surjective. ■

B.3 Theorem 6

Proof (Partial-order reversal) Pick any $A, B \in \mathcal{H}_G^*$ and $A \leq B$. For any $x, x' \in X$ that are in the same cell in partition $\pi(A) = X/A = \{Ax \mid x \in X\}$, $x' \in Ax = \{a(x) \mid a \in A\}$. Since $A \leq B$, then $Ax \subseteq Bx$, which further implies that $x' \in Bx$. So, x and x' are in the same cell in partition $\pi(B)$. Therefore, $\pi(A) \succeq \pi(B)$.

(Strong duality) Pick any $A, B \in \mathcal{H}_G^*$. By the definition of join, $A, B \leq A \vee B$, so from what we have shown at the beginning, $\pi(A), \pi(B) \succeq \pi(A \vee B)$, i.e. $\pi(A \vee B)$ is a common coarsening of $\pi(A)$ and $\pi(B)$. Since $\pi(A) \wedge \pi(B)$ is the finest common coarsening of $\pi(A)$ and $\pi(B)$, then $\pi(A \vee B) \preceq \pi(A) \wedge \pi(B)$. Conversely, for any $x, x' \in X$ that are in the same cell in partition $\pi(A \vee B) = \pi(\langle A \cup B \rangle) = X/\langle A \cup B \rangle = \{\langle A \cup B \rangle x \mid x \in X\}$, x and x' must be in the same orbit under $\langle A \cup B \rangle$-action on X, i.e. $x' \in \langle A \cup B \rangle x$ which means $x' = f_k \circ \cdots \circ f_1(x)$ for some finite integer k where $f_1, \ldots, f_k \in A \cup B$ (note: the fact that A, B are both subgroups ensures that $A \cup B$ is closed under inverses). This implies that x and $f_1(x)$ are either in the same cell in partition $\pi(A)$ or in the same cell in partition $\pi(B)$ depending on whether $f_1 \in A$ or $f_1 \in B$, but in either event, x and $f_1(x)$ must be in the same cell in any common coarsening of $\pi(A)$ and $\pi(B)$. Note that $\pi(A) \wedge \pi(B)$ is a common coarsening of $\pi(A)$ and $\pi(B)$ (regardless of the fact that it is the finest), so x and $f_1(x)$ are in the same cell in partition $\pi(A) \wedge \pi(B)$. Likewise, $f_1(x)$ and $f_2 \circ f_1(x)$, $f_3 \circ f_2 \circ f_1(x)$ and $f_2 \circ f_1(x)$, …, $f_{k-1} \circ \cdots \circ f_1(x)$ and x' are all in the same cell in partition $\pi(A) \wedge \pi(B)$. Therefore, x and x' are in the same cell in partition $\pi(A) \wedge \pi(B)$. So, $\pi(A \vee B) \succeq \pi(A) \wedge \pi(B)$. Combining both directions yields $\pi(A \vee B) = \pi(A) \wedge \pi(B)$.

(Weak duality) Pick any $A, B \in \mathcal{H}_G^*$. By the definition of meet, $A, B \geq A \wedge B$, so from what have shown at the beginning, $\pi(A), \pi(B) \preceq \pi(A \wedge B)$, i.e. $\pi(A \wedge B)$ is a common refinement of $\pi(A)$ and $\pi(B)$. Since $\pi(A) \vee \pi(B)$ is the coarsest common refinement of $\pi(A)$ and $\pi(B)$, then $\pi(A \wedge B) \succeq \pi(A) \vee \pi(B)$. We cannot obtain equality in general. For example, let $X = \mathbb{Z}$ and $A = \{r : \mathbb{Z} \to \mathbb{Z} \mid r(x) = kx, k \in \{-1, 1\}\}$, $B = \{t : \mathbb{Z} \to \mathbb{Z} \mid t(x) = x + k, k \in \mathbb{Z}\}$. It is clear that $A, B \leq \mathsf{F}(X)$ and $A \wedge B = A \cap B = \{\mathrm{id}\}$, so $\pi(A \wedge B) = X/\{\mathrm{id}\} = \{\{x\} \mid x \in \mathbb{Z}\}$, i.e. the finest partition of $\mathbb{Z}$. However, $\pi(A) = \{\{x, -x\} \mid x \in \mathbb{Z}\}$ and $\pi(B) = \{\mathbb{Z}\}$, i.e. the coarsest partition of $\mathbb{Z}$, so $\pi(A) \vee \pi(B) = \pi(A) = \{\{x, -x\} \mid x \in \mathbb{Z}\}$. In this example, we see that $\pi(A \wedge B) \succeq \pi(A) \vee \pi(B)$ but $\pi(A \wedge B) \neq \pi(A) \vee \pi(B)$. ■

B.4 Theorem 8

Proof Pick any $g \in G$ and $Y \in 2^X$. For any $x \in g \cdot Y$, we have $x = g \cdot y$ for some $y \in Y$. Since $Y \in 2^X$, i.e. $Y \subseteq X$, then $y \in X$. This implies that $x = g \cdot y \in X$. Therefore, $g \cdot Y \subseteq X$, i.e. $g \cdot Y \in 2^X$. To see the corresponding function $\cdot : G \times 2^X \to 2^X$ is a G-action on 2^X, we first check that the identity element $e \in G$ satisfies $e \cdot Y = \{e \cdot y \mid y \in Y\} = \{y \mid y \in Y\} = Y$; then check that for any $g, h \in G$, $g \cdot (h \cdot Y) = \{g \cdot z \mid z \in \{h \cdot y \mid y \in Y\}\} = \{g \cdot (h \cdot y) \mid y \in Y\} = \{(gh) \cdot y \mid y \in Y\} = (gh) \cdot Y$.

Pick any $g \in G$ and $\mathcal{P} \in \mathfrak{P}_X^*$. For any distinct elements $Q, Q' \in g \cdot \mathcal{P}$, we have $Q = g \cdot P$ and $Q' = g \cdot P'$ for some distinct $P, P' \in \mathcal{P}$, respectively. Since P, P' are two distinct cells in partition $\mathcal{P}$, $P \cap P' = \emptyset$. We claim that $Q \cap Q' = \emptyset$. Assume otherwise, then there exists a $q \in Q \cap Q'$ and $q = g \cdot p = g \cdot p'$ for some $p \in P, p' \in P'$. This implies that $p = (g^{-1}g) \cdot p = g^{-1} \cdot (g \cdot p) = g^{-1} \cdot (g \cdot p') = (g^{-1}g) \cdot p' = p' \in P \cap P'$, which contradicts the fact that $P \cap P' = \emptyset$. For any $x \in X$, $g^{-1} \cdot x \in X$, then there exists a cell $P \in \mathcal{P}$ such that $g^{-1} \cdot x \in P$. This implies that $x = (gg^{-1})x = g \cdot (g^{-1} \cdot x) \in g \cdot P$ which is an element in $g \cdot \mathcal{P}$. Therefore, the union of all elements in $g \cdot \mathcal{P}$ covers X, or more precisely, equals X, since every element in $g \cdot \mathcal{P}$ is a subset of X. Hence, $g \cdot \mathcal{P}$ is indeed a partition of X, i.e. $g \cdot \mathcal{P} \in \mathfrak{P}_X^*$. To see the corresponding function $\cdot : G \times \mathfrak{P}_X^* \to \mathfrak{P}_X^*$ is a G-action on $\mathfrak{P}_X^*$, we first check that the identity element $e \in G$ satisfies $e \cdot \mathcal{P} = \{e \cdot P \mid P \in \mathcal{P}\} = \{P \mid P \in \mathcal{P}\} = \mathcal{P}$; then check that for any $g, h \in G$, $g \cdot (h \cdot \mathcal{P}) = \{g \cdot Q \mid Q \in \{h \cdot P \mid P \in \mathcal{P}\}\} = \{g \cdot (h \cdot P) \mid P \in \mathcal{P}\} = \{(gh) \cdot P \mid P \in \mathcal{P}\} = (gh) \cdot \mathcal{P}$. ■

B.5 Theorem 9

Proof For any $Y \in \pi(g \circ H \circ g^{-1})$, Y is an orbit in X under $g \circ H \circ g^{-1}$, then $Y = (g \circ H \circ g^{-1})x = \{(g \circ h \circ g^{-1}) \cdot x \mid h \in H\} = \{g \circ h \circ g^{-1}(x) \mid h \in H\} = \{(g \circ h)(g^{-1}(x)) \mid h \in H\} = \{(gh) \cdot g^{-1}(x) \mid h \in H\} = \{g \cdot (h \cdot g^{-1}(x)) \mid h \in H\} = \{g \cdot y \mid y \in Hg^{-1}(x)\} = g \cdot Hg^{-1}(x)$ for some $x \in X$. Note that in the above derivation, $g^{-1}(x) \in X$ since $g \in \mathsf{F}(X)$. So, $Hg^{-1}(x)$ is the orbit of $g^{-1}(x)$ under H, i.e. $Hg^{-1}(x) \in \pi(H)$. This implies that $Y \in g \cdot \pi(H)$. Therefore, $\pi(g \circ H \circ g^{-1}) \subseteq g \cdot \pi(H)$.

Conversely, for any $Y \in g \cdot \pi(H)$, $Y = g \cdot P$ for some $P \in \pi(H)$. Note that P is an orbit in X under H, i.e. $P = Hx = \{h \cdot x \mid h \in H\}$ for some $x \in X$, then $Y = g \cdot P = \{g \cdot y \mid y \in P\} = \{g \cdot (h \cdot x) \mid h \in H\} = \{(gh) \cdot x \mid h \in H\} = \{g \circ h(x) \mid h \in H\} = \{g \circ h \circ g^{-1} \circ g(x) \mid h \in H\} = \{(g \circ h \circ g^{-1})(g(x)) \mid h \in H\} = \{(g \circ h \circ g^{-1}) \cdot g(x) \mid h \in H\} = (g \circ H \circ g^{-1})g(x)$ for some $x \in X$. Note that in the above derivation, $g(x) \in X$ since $g \in \mathsf{F}(X)$. Therefore, $(g \circ H \circ g^{-1})g(x)$ is the orbit of $g(x)$ under $g \circ H \circ g^{-1}$, i.e. $(g \circ H \circ g^{-1})g(x) \in \pi(g \circ H \circ g^{-1})$. This implies that $Y \in \pi(g \circ H \circ g^{-1})$. So, $g \cdot \pi(H) \subseteq \pi(g \circ H \circ g^{-1})$. ■

B.6 Theorem 12

Proof It is straightforward to check that

$$T \circ h = T \circ h' \iff h' \circ h^{-1} \in T \iff \ell(h' \circ h^{-1}) = I \iff \ell(h') = \ell(h).$$

The last if-and-only-if condition holds because $\ell(h' \circ h^{-1}) = \ell(h')\ell(h)^{-1}$ by Lemma 11. ■

B.7 Theorem 13

Proof Let $h, h' \in H$ be any two affine transformations in the same coset in H/T, then this means $T \circ h = T \circ h'$, or equivalently $h' \circ h^{-1} \in T$. By Equation (1), we have

$$\tau(h' \circ h^{-1}) = \tau(h') + \ell(h')\tau(h^{-1}) = \tau(h') + \ell(h')(-\ell(h)^{-1}\tau(h)) = \tau(h') - \tau(h),$$

where the last equality holds by Theorem 12. Therefore, $\tau(h') - \tau(h) \in \tau(T)$. ■

B.8 Theorem 15

Proof It is clear that $\ell(H) \leq \mathsf{GL}_n(\mathbb{R})$, since $H \leq \mathsf{AFF}(\mathbb{R}^n)$ and ℓ is a homomorphism (Lemma 11) which preserves subgroups. Let $\bar{\ell} : H/T \to \ell(H)$ be the function of the form $\bar{\ell}(T \circ h) = \ell(h)$, we claim that $\bar{\ell}$ is an isomorphism. To see this, for any $T \circ h, T \circ h' \in H/T$,

$$\bar{\ell}((T \circ h)(T \circ h')) = \bar{\ell}(T \circ (h \circ h')) = \ell(h \circ h') = \ell(h)\ell(h') = \bar{\ell}(T \circ h)\bar{\ell}(T \circ h'),$$

which implies $\bar{\ell}$ is a homomorphism. Further, for any $T \circ h, T \circ h' \in H/T$, if $\bar{\ell}(T \circ h) = \bar{\ell}(T \circ h')$, then $\ell(h) = \ell(h')$. By Theorem 12, this implies that $T \circ h = T \circ h'$, so $\bar{\ell}$ is injective. Lastly, for any $A \in \ell(H)$, there exists an $h \in H$ such that $\ell(h) = A$. For this particular h, $T \circ h \in H/T$, and $\bar{\ell}(T \circ h) = \ell(h) = A$. This implies that $\bar{\ell}$ is surjective. ■

B.9 Theorem 17

Proof For any $A \in \ell(H)$ and $v \in \tau(T)$, there exists an $f_{A,u} \in H$ and an $f_{I,v} \in T$. Since $T \trianglelefteq H$, then $f_{A,u} \circ f_{I,v} \circ f_{A,u}^{-1} \in T$. By Equation (1) we have that $f_{A,u} \circ f_{I,v} \circ f_{A,u}^{-1} = f_{I,Av}$, so $f_{I,Av} \in T$, i.e. $Av = \tau(f_{I,Av}) \in \tau(T)$. To see $\cdot : \ell(H) \times \tau(T) \to \tau(T)$ defines a group action of $\ell(H)$ on $\tau(T)$ is then easy, since it is a matrix-vector multiplication. A quick check shows that for any $v \in \tau(T)$, $I \cdot v = v$; for any $v \in \tau(T)$ and $A, B \in \ell(H)$, $A \cdot (B \cdot v) = (AB) \cdot v$. ■

B.10 Lemma 19

Proof For any $A \in L$, $\xi(A) = \xi(IA) = \xi(I) + I\xi(A) = \xi(I) + \xi(A)$. Note that $\xi(A), \xi(I) \in \mathbb{R}^n/V$, so $\xi(A) = V + a$ and $\xi(I) = V + b$ for some $a, b \in \mathbb{R}^n$. Thus,

$$\xi(A) = \xi(I) + \xi(A) \implies V + a = V + (b + a).$$

This further implies that $b \in V$ and $\xi(I) = V + b = V$.

For any $A \in L$, $V = \xi(A^{-1}A) = \xi(A^{-1}) + A^{-1}\xi(A)$. Note that $\xi(A), \xi(A^{-1}) \in \mathbb{R}^n/V$, so $\xi(A) = V + a$ and $\xi(A^{-1}) = V + c$ for some $a, c \in \mathbb{R}^n$. Thus,

$$V = \xi(A^{-1}) + A^{-1}\xi(A) \implies V = V + (c + A^{-1}a).$$

This further implies that $c + A^{-1}a \in V$, or equivalently, $c \in V + (-A^{-1}a)$. Therefore, c and $-A^{-1}a$ are in the same coset and $\xi(A^{-1}) = V + c = V + (-A^{-1}a) = -A^{-1}\xi(A)$. ■

B.11 Theorem 20

Proof Let $\Psi : \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)} \to \Sigma$ be the function defined by

$$\Psi(H) := (\ell(H),\ \tau(T),\ \xi_H) \quad \text{for any } H \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)},$$

where $T := \mathsf{T}(\mathbb{R}^n) \cap H$, and $\xi_H : \ell(H) \to \mathbb{R}^n/\tau(T)$ is given by $\xi_H(A) = \tau(\bar{\ell}^{-1}(A))$ with $\bar{\ell} : H/T \to \ell(H)$ being the isomorphism defined in the proof of Theorem 15. We first show Ψ is well-defined, and then show it is bijective. The entire proof is divided into four parts.

1. Check that ξ_H is well-defined. More specifically, we want to show that

$$\xi_H(A) \in \mathbb{R}^n/\tau(T) \quad \text{for any } H \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)} \text{ and } A \in \ell(H).$$

For any $A \in \ell(H)$, $\bar{\ell}^{-1}(A)$ is the coset $T \circ h$ in H/T such that $\ell(h) = A$. Pick any $h \in \bar{\ell}^{-1}(A)$ which is possible since as a coset $\bar{\ell}^{-1}(A) \neq \emptyset$. For any $h' \in \bar{\ell}^{-1}(A)$, by Theorem 13, $\tau(h') - \tau(h) \in \tau(T)$, i.e. $\tau(h') \in \tau(T) + \tau(h)$, so $\tau(\bar{\ell}^{-1}(A)) \subseteq \tau(T) + \tau(h)$. Conversely, for any $w \in \tau(T) + \tau(h)$, there exists a $v \in \tau(T)$ such that $w = v + \tau(h)$. Note that the pure translation $t_v \in T \leq H$ and $h \in \bar{\ell}^{-1}(A) \subseteq H$, so their composition $t_v \circ h \in H$. Further, it is an easy check that $\ell(t_v \circ h) = A$ and $\tau(t_v \circ h) = v + \tau(h) = w$. This implies that we have found $h' := t_v \circ h \in \bar{\ell}^{-1}(A)$ and $\tau(h') = w$, thus, $w \in \tau(\bar{\ell}^{-1}(A))$. This finally yields that $\tau(T) + \tau(h) \subseteq \tau(\bar{\ell}^{-1}(A))$. Combining the two directions, we have $\tau(\bar{\ell}^{-1}(A)) = \tau(T) + \tau(h)$; so, $\xi_H(A) = \tau(\bar{\ell}^{-1}(A)) \in \mathbb{R}^n/\tau(T)$. This implies ξ_H is well-defined.

2. Check that Ψ is well-defined. More specifically, we want to show that

$$\Psi(H) \in \Sigma \quad \text{for any } H \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}.$$

For any $H \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$, it is clear that $\ell(H) \leq \mathsf{GL}_n(\mathbb{R})$, $\tau(T) \leq \mathbb{R}^n$, and they are compatible (Theorem 17); therefore, it suffices to show that $\xi_H \in \Xi_{\ell(H),\tau(T)}$. Note that, for any $A, A' \in \ell(H)$, the product of two cosets $\bar{\ell}^{-1}(A)\bar{\ell}^{-1}(A') = (T \circ f_{A,u})(T \circ f_{A',u'}) = T \circ (f_{A,u} \circ f_{A',u'}) = T \circ f_{AA',u+Au'}$, for some $f_{A,u}, f_{A',u'} \in H$. Therefore,

$$\xi_H(AA') = \tau(\bar{\ell}^{-1}(AA')) = \tau(\bar{\ell}^{-1}(A)\bar{\ell}^{-1}(A')) = \tau(T \circ f_{AA',u+Au'}) = \tau(T) + u + Au'.$$

On the other hand,

$$\xi_H(A) + A\xi_H(A') = (\tau(T) + u) + A(\tau(T) + u') = \tau(T) + u + Au'.$$

Therefore, $\xi_H(AA') = \xi_H(A) + A\xi_H(A')$ and $\xi_H \in \Xi_{\ell(H),\tau(T)}$. This implies that for any $H \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$, $\Psi(H) \in \Sigma$, so Ψ is well-defined.

3. Check that Ψ is injective. Pick any $H, H' \in \mathcal{H}^*_{\mathsf{AFF}(\mathbb{R}^n)}$ and suppose $\Psi(H) = \Psi(H')$, i.e. $(\ell(H), \tau(T), \xi_H) = (\ell(H'), \tau(T'), \xi_{H'})$, where $T := \mathsf{T}(\mathbb{R}^n) \cap H$ and $T' := \mathsf{T}(\mathbb{R}^n) \cap H'$. For any $f_{A,u} \in H$, $A = \ell(f_{A,u}) \in \ell(H) = \ell(H')$; thus, there exists some $f_{A,u'} \in H'$. Let $\bar{\ell} : H/T \to \ell(H)$ and $\bar{\ell}' : H'/T' \to \ell(H')$ be the isomorphisms similarly defined as in Theorem 15. As proved earlier, we have

$$\xi_H(A) = \tau(\bar{\ell}^{-1}(A)) = \tau(T) + \tau(f_{A,u}) = \tau(T) + u,$$
$$\xi_{H'}(A) = \tau(\bar{\ell}'^{-1}(A)) = \tau(T') + \tau(f_{A,u'}) = \tau(T) + u'.$$

Therefore, $\tau(T) + u = \tau(T) + u'$. This implies that $\tau(f_{A,u}) = u \in \tau(T) + u' = \tau(\bar{\ell}'^{-1}(A))$. So, $f_{A,u} \in \bar{\ell}'^{-1}(A) \subseteq H'$, and $H \subseteq H'$. By a completely symmetrical process, $H' \subseteq H$. Therefore, $H = H'$, which implies that Ψ is injective.

4. Check that Ψ is surjective. Pick any $(L, V, \xi) \in \Sigma$ and let

$$H := \{f_{A,u} \in \mathsf{AFF}(\mathbb{R}^n) \mid A \in L, u \in \xi(A)\}.$$

We first show that $H \leq \mathsf{AFF}(\mathbb{R}^n)$ by a subgroup test. It is clear $H \subseteq \mathsf{AFF}(\mathbb{R}^n)$. The identity matrix $I \in L$ and $\mathbf{0} \in V = \xi(I)$, so the identity transformation $\mathrm{id} = f_{I,\mathbf{0}} \in H$. For any $f_{A,u}, f_{A',u'} \in H$, we have $A, A' \in L$ and $u \in \xi(A), u' \in \xi(A')$, which respectively implies that $AA' \in L$ and $u + Au' \in \xi(A) + A\xi(A') = \xi(AA')$. So, $f_{A,u} \circ f_{A',u'} = f_{AA',u+Au'} \in H$. For any $f_{A,u} \in H$, we have $A \in L$ and $u \in \xi(A)$, which respectively implies that $A^{-1} \in L$ and $-A^{-1}u \in -A^{-1}\xi(A) = \xi(A^{-1})$. So, $f^{-1}_{A,u} = f_{A^{-1},-A^{-1}u} \in H$. Therefore, $H \leq \mathsf{AFF}(\mathbb{R}^n)$. Now we show that $\Psi(H) = (\ell(H),\ \tau(T),\ \xi_H) = (L, V, \xi)$. First, for any $A \in \ell(H)$, there exists an $f_{A,u} \in H$, so $A \in L$ which implies that $\ell(H) \subseteq L$. Conversely, for any $A \in L$, $\xi(A)$ is a coset in $\mathbb{R}^n/V$, so $\xi(A) \neq \emptyset$. Pick any $u \in \xi(A)$, then $f_{A,u} \in H$, so $A = \ell(f_{A,u}) \in \ell(H)$ which implies $L \subseteq \ell(H)$. Combining both directions yields $\ell(H) = L$. Second, note that $T = \mathsf{T}(\mathbb{R}^n) \cap H = \{f_{I,u} \in \mathsf{AFF}(\mathbb{R}^n) \mid u \in \xi(I) = V\}$, so $\tau(T) = \{u \mid u \in V\} = V$. Third, note that $\xi_H : \ell(H) \to \mathbb{R}^n/\tau(T)$ and $\xi : L \to \mathbb{R}^n/V$. We have shown that $\ell(H) = L$ and $\tau(T) = V$, so ξ_H and ξ have the same domain and codomain. Further, for any $A \in L$, $\xi_H(A) = \tau(\bar{\ell}^{-1}(A)) = \tau(\{f_{A,u} \in \mathsf{AFF}(\mathbb{R}^n) \mid u \in \xi(A)\}) = \{u \mid u \in \xi(A)\} = \xi(A)$. So, $\xi_H = \xi$. Now we have $\Psi(H) = (L, V, \xi)$. Therefore, Ψ is surjective. ∎

B.12 Theorem 26

Proof Pick any $f' \in \mathsf{F}(Y)$, and let $f : X \to X$ be the function given by

$$f(x) = \begin{cases} f'(x), & x \in Y; \\ x, & x \in X \backslash Y. \end{cases}$$

Then it is clear that $f(Y) = f'(Y) = Y$ and $f(X\backslash Y) = X\backslash Y$. For any $x, x' \in X$ and $f(x) = f(x')$: if $f(x) \in X\backslash Y$ then $x = x'$; otherwise $x, x' \in Y$ and $f'(x) = f'(x')$ which yields $x = x'$ since f' is injective. This implies that f is injective. f is also surjective, since $f(X) = f(Y \cup (X\backslash Y)) = f(Y) \cup f(X\backslash Y) = Y \cup (X\backslash Y) = X$. So $f \in \mathsf{F}(X)$. Further, the fact that $f(Y) = f'(Y) = Y$ implies that $f \in \mathsf{F}(X)_Y$ and $f' = f|_Y \in \mathsf{F}(X)_Y|_Y$. Therefore, $\mathsf{F}(Y) \subseteq \mathsf{F}(X)_Y|_Y$. Conversely, pick any $f|_Y \in \mathsf{F}(X)_Y|_Y$. $f|_Y$ is injective since $f \in \mathsf{F}(X)_Y \subseteq \mathsf{F}(X)$ is injective; $f|_Y$ is surjective since $f|_Y(Y) = f(Y) = Y$. So $f|_Y \in \mathsf{F}(Y)$. This implies that $\mathsf{F}(X)_Y|_Y \subseteq \mathsf{F}(Y)$. ■

B.13 Theorem 28

Proof Pick any $t'_u \in \mathsf{T}(\mathbb{Z}^n)$, then by definition, $u \in \mathbb{Z}^n$, and $t'_u(x) = x + u$, for any $x \in \mathbb{Z}^n$. Let $t : \mathbb{R}^n \to \mathbb{R}^n$ be the function given by $t(x) = x + u$. Since $u \in \mathbb{Z}^n$, then $u \in \mathbb{R}^n$, so $t \in \mathsf{T}(\mathbb{R}^n)$. Further, note that $t(\mathbb{Z}^n) = \mathbb{Z}^n$; therefore, $t \in \mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}$. It follows that $t'_u = t|_{\mathbb{Z}^n} \in \mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, so $\mathsf{T}(\mathbb{Z}^n) \subseteq \mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$.

Pick any $t' \in \mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, then by definition, there exists a $t_u \in \mathsf{T}(\mathbb{R}^n)$ where $u \in \mathbb{R}^n$ such that $t_u(\mathbb{Z}^n) = \mathbb{Z}^n$ and $t' = t_u|_{\mathbb{Z}^n}$, i.e. $t'(x) = x + u$, for any $x \in \mathbb{Z}^n$. The condition $t_u(\mathbb{Z}^n) = \mathbb{Z}^n$ implies in particular $t_u(\mathbf{0}) = u \in \mathbb{Z}^n$. It follows that $t' \in \mathsf{T}(\mathbb{Z}^n)$, so $\mathsf{T}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n} \subseteq \mathsf{T}(\mathbb{Z}^n)$.

Pick any $r'_A \in \mathsf{R}(\mathbb{Z}^n)$, then by definition, $A \in \mathsf{O}_n(\mathbb{Z})$, and $r'_A(x) = Ax$, for any $x \in \mathbb{Z}^n$. Let $r : \mathbb{R}^n \to \mathbb{R}^n$ be the function given by $r(x) = Ax$. Since $A \in \mathsf{O}_n(\mathbb{Z})$, then $A \in \mathsf{O}_n(\mathbb{R})$, so $r \in \mathsf{R}(\mathbb{R}^n)$. Further, note that $r(\mathbb{Z}^n) = \mathbb{Z}^n$; therefore, $r \in \mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}$. It follows that $r'_A = r|_{\mathbb{Z}^n} \in \mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, so $\mathsf{R}(\mathbb{Z}^n) \subseteq \mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$.

Pick any $r' \in \mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, then by definition, there exists a $r_A \in \mathsf{R}(\mathbb{R}^n)$ where $A \in \mathsf{O}_n(\mathbb{R})$ such that $r_A(\mathbb{Z}^n) = \mathbb{Z}^n$ and $r' = r_A|_{\mathbb{Z}^n}$, i.e. $r'(x) = Ax$, for any $x \in \mathbb{Z}^n$. The condition $r_A(\mathbb{Z}^n) = \mathbb{Z}^n$ implies in particular $A\boldsymbol{e_i} \in \mathbb{Z}^n$ for all i, i.e. the columns of A are from $\mathbb{Z}^n$. So $A \in \mathsf{O}_n(\mathbb{Z})$. It follows that $r' \in \mathsf{R}(\mathbb{Z}^n)$, so $\mathsf{R}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n} \subseteq \mathsf{R}(\mathbb{Z}^n)$. ■

B.14 Theorem 29

Proof Pick any $h' \in \mathsf{ISO}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$, then by definition, there exists an $h \in \mathsf{ISO}(\mathbb{R}^n)$ such that $h(\mathbb{Z}^n) = \mathbb{Z}^n$ and $h' = h|_{\mathbb{Z}^n}$. For any $x, y \in \mathbb{Z}^n$,

$$\mathrm{d}(h'(x), h'(y)) = \mathrm{d}(h|_{\mathbb{Z}^n}(x), h|_{\mathbb{Z}^n}(y)) = \mathrm{d}(h(x), h(y)) = \mathrm{d}(x, y).$$

This implies that $h' \in \mathsf{ISO}(\mathbb{Z}^n)$. So, $\mathsf{ISO}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n} \subseteq \mathsf{ISO}(\mathbb{Z}^n)$.

Conversely, pick any $h' \in \mathsf{ISO}(\mathbb{Z}^n)$ and let $h'_0 = h' - h'(\mathbf{0})$. Note that $h'_0 \in \mathsf{ISO}(\mathbb{Z}^n)$ and $h'_0(\mathbf{0}) = \mathbf{0}$. This implies that $\|h'_0(x)\|_2 = \mathrm{d}(h'_0(x), h'_0(\mathbf{0})) = \mathrm{d}(x, \mathbf{0}) = \|x\|_2$, for any $x \in \mathbb{Z}^n$.

Further, for any $x, y \in \mathbb{Z}^n$, expanding the distance-preserving equation $\|h'_0(x) - h'_0(y)\|_2^2 = \|x - y\|_2^2$ and cancelling equal terms (i.e. $\|h'_0(x)\|_2^2 = \|x\|_2^2$ and $\|h'_0(y)\|_2^2 = \|y\|_2^2$) yields

$$\langle h'_0(x), h'_0(y)\rangle = \langle x, y\rangle \quad \text{for all } x, y \in \mathbb{Z}^n.$$

Now let $h : \mathbb{R}^n \to \mathbb{R}^n$ be the function given by $h(x) = Ax+u$, where $A = [h'_0(\boldsymbol{e_1}), \cdots, h'_0(\boldsymbol{e_n})] \in \mathbb{Z}^{n\times n}$ and $u = h'(\mathbf{0}) \in \mathbb{Z}^n$. Moreover, $\langle h'_0(\boldsymbol{e_i}), h'_0(\boldsymbol{e_j})\rangle = \langle \boldsymbol{e_i}, \boldsymbol{e_j}\rangle = \delta_{ij}$. So, A is orthogonal, i.e. $A \in \mathsf{O}_n(\mathbb{Z}) \subseteq \mathsf{O}_n(\mathbb{R})$. This implies $h \in \mathsf{ISO}(\mathbb{R}^n)$. We claim $h(\mathbb{Z}^n) = \mathbb{Z}^n$ and $h' = h|_{\mathbb{Z}^n}$.

To see $h(\mathbb{Z}^n) = \mathbb{Z}^n$, first pick any $x \in h(\mathbb{Z}^n)$, then there exists $y \in \mathbb{Z}^n$ such that $x = h(y) = Ay + u$. Since $A \in \mathbb{Z}^{n\times n}$ and $u \in \mathbb{Z}^n$, $x \in \mathbb{Z}^n$ which implies $h(\mathbb{Z}^n) \subseteq \mathbb{Z}^n$. Conversely, pick any $x \in \mathbb{Z}^n$. Let $y = A^\top(x - u)$, then $y \in \mathbb{Z}^n$ and $h(y) = Ay + u = x$. So $x \in h(\mathbb{Z}^n)$ which implies $\mathbb{Z}^n \subseteq h(\mathbb{Z}^n)$.

To see $h' = h|_{\mathbb{Z}^n}$, pick any $x \in \mathbb{Z}^n$, then $\langle h'_0(\boldsymbol{e_i}), h'_0(x)\rangle = \langle \boldsymbol{e_i}, x\rangle = \langle A\boldsymbol{e_i}, Ax\rangle = \langle h'_0(\boldsymbol{e_i}), Ax\rangle$, for all $i = 1, \dots, n$. So, $\langle h'_0(\boldsymbol{e_i}), h'_0(x) - Ax\rangle = 0$, for all $i = 1, \dots, n$, that is

$$A^\top(h'_0(x) - Ax) = \mathbf{0}.$$

Multiplying both sides by A yields $h'_0(x) = Ax$, for all $x \in \mathbb{Z}^n$. Hence,

$$h(x) = Ax + u = h'_0(x) + h'(\mathbf{0}) = h'(x) \quad \text{for all } x \in \mathbb{Z}^n.$$

That is, $h' = h|_{\mathbb{Z}^n}$. It follows that $h' \in \mathsf{ISO}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$. So, $\mathsf{ISO}(\mathbb{Z}^n) \subseteq \mathsf{ISO}(\mathbb{R}^n)_{\mathbb{Z}^n}|_{\mathbb{Z}^n}$. ■

B.15 Theorem 32

Proof For any $A, B \subseteq S$, we have $\pi'(A \cup B) = \langle A \cup B\rangle = \langle\langle A\rangle \cup \langle B\rangle\rangle = \langle \pi'(A) \cup \pi'(B)\rangle = \pi'(A) \vee \pi'(B)$. Then for any $\pi'(A), \pi'(B) \in \pi'(2^S)$ where $A, B \subseteq S$, the join $\pi'(A) \vee \pi'(B) = \pi'(A \cup B) \in \pi'(2^S)$, since $A \cup B \subseteq S$. So, $(\pi'(2^S), \leq)$ is a join-semilattice.

We show $(\pi'(2^S), \leq)$ is not a meet-semilattice via an example. Let $X = \mathbb{R}^n$ and $S = \{t_{\boldsymbol{e_1}}, t_{\boldsymbol{e_2}}, t_{(3/2)\cdot\mathbf{1}}\}$ where $\boldsymbol{e_1} = (1, 0), \boldsymbol{e_2} = (0, 1), \mathbf{1} = (1, 1)$. Further, let $A = \{t_{\boldsymbol{e_1}}, t_{\boldsymbol{e_2}}\}$ and $B = \{t_{(3/2)\cdot\mathbf{1}}\}$. The meet $\pi'(A) \wedge \pi'(B) = \langle A\rangle \cap \langle B\rangle = \langle t_{3\cdot\mathbf{1}}\rangle \notin \pi'(2^S)$. ■

B.16 Theorem 36

Proof It is an exercise to check that all entities in the theorem are indeed groups.

Let $\phi_{\mathsf{P}} : \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n) \to \mathsf{P}_n$ be the function given by $\phi_{\mathsf{P}}(r'_P) = P$, for any $r'_P \in \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n)$. For any $r'_P, r'_Q \in \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n)$, if $\phi_{\mathsf{P}}(r'_P) = \phi_{\mathsf{P}}(r'_Q)$, i.e. $P = Q$, then $r'_P = r'_Q$, so ϕ_{P} is injective. For any $P \in \mathsf{P}_n$, $r'_P \in \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n)$ and $\phi_{\mathsf{P}}(r'_P) = P$, so ϕ_{P} is surjective. Further, for any $r'_P, r'_Q \in \mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n)$, $\phi_{\mathsf{P}}(r'_P \circ r'_Q) = \phi_{\mathsf{P}}(r'_{P\cdot Q}) = P \cdot Q = \phi_{\mathsf{P}}(r'_P) \cdot \phi_{\mathsf{P}}(r'_Q)$, so ϕ_{P} is a homomorphism. Now we see that ϕ_{P} is an isomorphism. So, $(\mathsf{R}_{\mathsf{P}}(\mathbb{Z}^n), \circ) \cong (\mathsf{P}_n, \cdot)$.

Let $\phi_S : S_n \to \mathsf{P}_n$ be the function given by $\sigma \mapsto P^\sigma$, where P^σ is an $n \times n$ permutation matrix obtained by permuting the rows of the identity matrix according to σ, i.e.

$$P^\sigma_{ij} = \begin{cases} 1 & i = \sigma(j) \\ 0 & i \neq \sigma(j) \end{cases} \quad \text{for any } i, j \in \{1, \dots, n\}.$$

For any $\sigma, \mu \in S_n$, if $\phi_S(\sigma) = \phi_S(\mu)$, i.e. $P^\sigma = P^\mu$, then $\sigma(j) = \mu(j)$ for all $j \in \{1, \ldots, n\}$, i.e. $\sigma = \mu$, so ϕ_S is injective. For any $P \in \mathsf{P}_n$, let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be the function given by $\sigma(j) \in \{i | P_{ij} = 1\}$, which is well-defined since $\{i | P_{ij} = 1\}$ is a singleton for all $j \in \{1, \ldots, n\}$ given that $P \in \mathsf{P}_n$ is a permutation matrix. It is clear that $\sigma \in S_n$, and $\phi_S(\sigma) = P$. So, ϕ_S is surjective. Further, for any $\sigma, \mu \in S_n$, $\phi_S(\sigma \circ \mu) = P^{\sigma \circ \mu} = P^\sigma \cdot P^\mu = \phi_S(\sigma) \cdot \phi_S(\mu)$ where the second equality holds because for all $i, j \in \{1, \ldots, n\}$,

$$(P^\sigma \cdot P^\mu)_{ij} = \sum_{k=1}^{n} P^\sigma_{ik} \cdot P^\mu_{kj} = P^\sigma_{i\mu(j)} \cdot 1 = \begin{cases} 1 & i = \sigma \circ \mu(j) \\ 0 & i \neq \sigma \circ \mu(j) \end{cases} = P^{\sigma \circ \mu}_{ij},$$

so ϕ_S is a homomorphism. Now we see that ϕ_S is an isomorphism. So, $(S_n, \circ) \cong (\mathsf{P}_n, \cdot)$.

Let $\phi_{\mathsf{N}} : \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n) \to \mathsf{N}_n$ be the function given by $\phi_{\mathsf{N}}(r'_N) = N$, for any $r'_N \in \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n)$. For any $r'_N, r'_M \in \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n)$, if $\phi_{\mathsf{N}}(r'_N) = \phi_{\mathsf{N}}(r'_M)$, i.e. $N = M$, then $r'_N = r'_M$, so ϕ_{N} is injective. For any $N \in \mathsf{N}_n$, $r'_N \in \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n)$ and $\phi_{\mathsf{N}}(r'_N) = N$, so ϕ_{N} is surjective. Further, for any $r'_N, r'_M \in \mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n)$, $\phi_{\mathsf{N}}(r'_N \circ r'_M) = \phi_{\mathsf{N}}(r'_{N \cdot M}) = N \cdot M = \phi_{\mathsf{N}}(r'_N) \cdot \phi_{\mathsf{N}}(r'_M)$, so ϕ_{N} is a homomorphism. Now we see that ϕ_{N} is an isomorphism. So, $(\mathsf{R}_{\mathsf{N}}(\mathbb{Z}^n), \circ) \cong (\mathsf{N}_n, \cdot)$. ■

B.17 Theorem 37

Proof We first show that $\mathsf{N}_n, \mathsf{P}_n \leq \mathsf{O}_n(\mathbb{Z})$. $(\mathsf{O}_n(\mathbb{Z}), \cdot)$ is a group since matrix multiplication $\cdot$ is associative, $I \in \mathsf{O}_n(\mathbb{Z})$ is the identity element, and for any $A \in \mathsf{O}_n(\mathbb{Z})$, $A^\top \in \mathsf{O}_n(\mathbb{Z})$ is its inverse. Pick any $N \in \mathsf{N}_n$, then $N \in \mathbb{Z}^{n \times n}$ and $N^\top N = NN = I$, so $N \in \mathsf{O}_n(\mathbb{Z})$, which implies that $\mathsf{N}_n \subseteq \mathsf{O}_n(\mathbb{Z})$. Pick any $P \in \mathsf{P}_n$, then $P = [\boldsymbol{e_{\sigma(1)}}, \cdots, \boldsymbol{e_{\sigma(n)}}] \in \mathbb{Z}^{n \times n}$ for some $\sigma \in S_n$ and $(P^\top P)_{ij} = \boldsymbol{e}^\top_{\boldsymbol{\sigma(i)}} \boldsymbol{e_{\sigma(j)}} = \delta_{ij}$, i.e. $P^\top P = I$, so $P \in \mathsf{O}_n(\mathbb{Z})$, which implies that $\mathsf{P}_n \subseteq \mathsf{O}_n(\mathbb{Z})$. Now we perform subgroup tests to show that $\mathsf{N}_n, \mathsf{P}_n \leq \mathsf{O}_n(\mathbb{Z})$. First, we check that 1) $I \in \mathsf{N}_n$, 2) for any $N, N' \in \mathsf{N}_n$, $NN' \in \mathsf{N}_n$, 3) for any $N \in \mathsf{N}_n$, $N^{-1} = N \in \mathsf{N}_n$; therefore, $\mathsf{N}_n \leq \mathsf{O}_n(\mathbb{Z})$. Second, we check that 1) $I \in \mathsf{P}_n$, 2) for any $P, P' \in \mathsf{P}_n$, $PP' \in \mathsf{P}_n$, 3) for any $P \in \mathsf{P}_n$, $P^{-1} = P^\top \in \mathsf{P}_n$; therefore, $\mathsf{P}_n \leq \mathsf{O}_n(\mathbb{Z})$.

Now we show that $\mathsf{N}_n \cap \mathsf{P}_n = \{I\}$. Pick any $N \in \mathsf{N}_n \backslash \{I\}$ and any $P \in \mathsf{P}_n$. It is clear that $N \neq P$ since N has at least one -1 entries while P has no -1 entries. This implies that $(\mathsf{N}_n \backslash \{I\}) \cap \mathsf{P}_n = \emptyset$. Further, $I \in \mathsf{N}_n \cap \mathsf{P}_n$. Therefore, $\mathsf{N}_n \cap \mathsf{P}_n = \{I\}$.

Lastly we show that $\mathsf{O}_n(\mathbb{Z}) = \langle \mathsf{N}_n \cup \mathsf{P}_n \rangle$. It is clear that $\langle \mathsf{N}_n \cup \mathsf{P}_n \rangle \subseteq \mathsf{O}_n(\mathbb{Z})$, since $\mathsf{N}_n, \mathsf{P}_n \leq \mathsf{O}_n(\mathbb{Z})$. Conversely, pick any $A = [\boldsymbol{a_1}, \cdots, \boldsymbol{a_n}] \in \mathsf{O}_n(\mathbb{Z})$ where $\boldsymbol{a_i}$ denotes the ith column of A. By definition, $A^\top A = I$, so $\langle \boldsymbol{a_i}, \boldsymbol{a_i} \rangle = \|\boldsymbol{a_i}\|_2^2 = 1$ for $i \in \{1, \ldots, n\}$, and $\langle \boldsymbol{a_i}, \boldsymbol{a_j} \rangle = 0$ for $i, j \in \{1, \ldots, n\}$ and $i \neq j$. On the one hand, given $A \in \mathbb{Z}^{n \times n}$, the unit-norm property $\|\boldsymbol{a_i}\|_2^2 = 1$ implies that $\boldsymbol{a_i}$ is a standard basis vector or its negation, i.e. $\boldsymbol{a_i} = \pm \boldsymbol{e_k}$ for some $k \in \{1, \ldots, n\}$. On the other hand, for $i \neq j$, the orthogonality property $\langle \boldsymbol{a_i}, \boldsymbol{a_j} \rangle = 0$ implies that for some $k \neq k'$, $\boldsymbol{a_i} = \pm \boldsymbol{e_k}$ and $\boldsymbol{a_j} = \pm \boldsymbol{e_{k'}}$. Thus, there exist some vector $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{1, -1\}^n$ and some permutation $\sigma \in S_n$ such that $A = [\alpha_1 \boldsymbol{e_{\sigma(1)}}, \cdots, \alpha_n \boldsymbol{e_{\sigma(n)}}] = \operatorname{diag}(\alpha)[\boldsymbol{e_{\sigma(1)}}, \cdots, \boldsymbol{e_{\sigma(n)}}] = NP$, where $N = \operatorname{diag}(\alpha) \in \mathsf{N}_n$ and $P = [\boldsymbol{e_{\sigma(1)}}, \cdots, \boldsymbol{e_{\sigma(n)}}] \in \mathsf{P}_n$. This implies $A \in \langle \mathsf{N}_n \cup \mathsf{P}_n \rangle$. So, $\mathsf{O}_n(\mathbb{Z}) \subseteq \langle \mathsf{N}_n \cup \mathsf{P}_n \rangle$. ■

B.18 Theorem 41

Proof Suppose for any $s \in S$, $s \notin \langle S \backslash \{s\} \rangle$. However, $s \in \langle S \rangle$; so, $\langle S \backslash \{s\} \rangle \neq \langle S \rangle$. By definition, S is a minimal generating set. On the other hand, suppose there exists an $s \in S$ such that $s \in \langle S \backslash \{s\} \rangle$, i.e. $s = s_k \circ \cdots \circ s_1$ for some k where $s_k, \ldots, s_1 \in (S \backslash \{s\}) \cup (S \backslash \{s\})^{-1}$. Pick any $s' \in \langle S \rangle$, $s' = s'_{k'} \circ \cdots \circ s'_1$ for some k' where $s'_{k'}, \ldots, s'_1 \in S \cup S^{-1}$. For any $i \in \{1, \ldots, k'\}$, if $s'_i = s$, replace it with $s_k \circ \cdots \circ s_1$; if $s'_i = s^{-1}$, replace it with $s_1^{-1} \circ \cdots \circ s_k^{-1}$; otherwise $s'_i \in (S \backslash \{s\}) \cup (S \backslash \{s\})^{-1}$, do nothing. This results in an expression of s' as the composition of finitely many elements in $(S \backslash \{s\}) \cup (S \backslash \{s\})^{-1}$, i.e. $s' \in \langle S \backslash \{s\} \rangle$. So, $\langle S \rangle \subseteq \langle S \backslash \{s\} \rangle$. It is trivial to see that $\langle S \backslash \{s\} \rangle \subseteq \langle S \rangle$ since $S \backslash \{s\} \subseteq S$. Therefore, $\langle S \backslash \{s\} \rangle = \langle S \rangle$. By definition, S is not a minimal generating set. ■