Communication-Optimal Loop Nests

Nick Knight

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-185
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-185.html

August 12, 2015

Communication-Optimal Loop Nests

by

Nicholas Sullender Knight

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

and the Designated Emphasis

in

Computational and Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor James Demmel, Chair
Professor Michael Christ
Professor Katherine Yelick

Summer 2015

Abstract

Communication-Optimal Loop Nests

by

Nicholas Sullender Knight

Doctor of Philosophy in Computer Science

Designated Emphasis in Computational and Data Science and Engineering

University of California, Berkeley

Professor James Demmel, Chair

Communication (data movement) often dominates a computation's runtime and energy costs, motivating organizing an algorithm's operations to minimize communication. We study communication costs of a class of algorithms including many-body and matrix/tensor computations and, more generally, loop nests operating on array variables subscripted by linear functions of the loop iteration vector. We use this algebraic relationship between variables and operations to derive communication lower bounds for these algorithms. We also discuss communication-optimal implementations that attain these bounds.

Contents

Acknowledgements **i**

1 Introduction **1**

2 Modeling Computation and Communication **5**

3 Modeling Loop Nests **19**

4 Hölder-Brascamp-Lieb-type Inequalities **27**

5 Communication Bounds for Loop Nests **67**

6 Conclusion **83**

References **85**

Acknowledgements

My doctoral studies (August 15, 2009 through August 14, 2015) were supported financially by the Par Lab, funded by Microsoft (024263), Intel (024894), UC Discovery (DIG07-10227), National Instruments, Nokia, NVIDIA, Oracle, and Samsung; by ASPIRE, funded by DARPA (HR0011-12-2-0016), C-FAR (a STARnet member, funded by the Semiconductor Research Corporation and DARPA), Intel, Google, Huawei, LG, Nokia, NVIDIA, Oracle, and Samsung; by the MathWorks; by the DOE (DE-SC0003959, DE-SC0004938, DE-SC0005136, DE-SC0008700, DE-AC02-05CH11231, DE-SC0010200); and by the NSF (ACI-1339676).

This dissertation focuses on an ongoing research project with collaborators Michael Christ, James Demmel, Thomas Scanlon, and Katherine Yelick, who I joined in January 2012. This project has benefitted from helpful technical discussions with Grey Ballard, Erin Carson, Benjamin Lipshitz, Oded Schwartz, Harsha Vardhan Simhadri, and Bernd Sturmfels.

Chapter 1

Introduction

Computers take time and energy to execute programs. The costs of moving data (*communication*) frequently dominate the costs of performing operations on that data (*computation*). This work studies organizing computations to minimize communication.

1.1 Avoiding Communication

Computers are ubiquitous in our lives, and have had a profound impact on the scientific method. Computer-based simulation enables scientists to study phenomena that are infeasible for lab experimentation, as well as collect and process data beyond human capabilities. However, computers have physical limitations too, and their users will only wait so long (or pay for so much electricity) before they expect a result. Thus it is important to ask how much computational problems cost and to develop algorithms and implementations to minimize costs.

In practice, communication costs more than computation (in terms of both time and energy) for technological reasons: improvements in processing speed and efficiency consistently exceed improvements in accessing memory, leading to an exponentially growing performance gap. So, even if the costs of a computer program are not dominated by communication on current hardware, they will likely be in the future. Additionally, as a program's working set grows, it must move data further, incurring a greater communication cost. Reducing communication costs will continue to be an important part of improving algorithmic performance.

1.2 Communication Lower Bounds

In this work, we derive *communication lower bounds* for algorithms that apply to all implementations, i.e., "any implementation of the given algorithm must move at least this much data". This means that if an implementation attains a communication lower bound, then to further reduce communication, we must seek a new algorithm. On the other hand, if there is a gap between the best known implementation and the lower bound, then it may be worth trying to find a better implementation. Communication lower bounds have practical applications in program optimization and can give useful insights into algorithm design.

The seminal work of Hong-Kung [21] provided a framework for reasoning about the communication within a sequential machine, in particular, the data movement between a fast memory with bounded capacity and a slow memory with unbounded capacity. Hong-Kung modeled an algorithm by the *directed acyclic graph (DAG)* of dependences between the operations performed in any execution of that algorithm, and modeled an implementation by playing a *pebble game* on the DAG's vertices, wherein placing/removing a pebble represented writing/erasing a memory cell. Hong-Kung's key result [21, Theorem 3.1] takes an algorithm and returns a lower bound on the communication in any implementation of that algorithm. Of practical importance, this lower bound is computable by solving a graph-partitioning problem, despite the fact that there are infinitely many possible implementations.

Hong-Kung's work inspired a large literature, including the present work. Our main connection with Hong-Kung's work is through the papers of Irony-Toledo-Tiskin [22] and Ballard-Demmel-Holtz-Schwartz [5], who specialized Hong-Kung's algorithm model to target a class of matrix computations whose associated DAGs have a three-dimensional geometric structure.

For example, consider the following program to multiply N-by-N matrices, $C = A \cdot B$, assuming C is zero-initialized:

$$\begin{aligned} &\textbf{for } i = 1 : N, \quad \textbf{for } j = 1 : N, \quad \textbf{for } k = 1 : N, \\ &\quad C(i,j) = C(i,j) + A(i,k) * B(k,j). \end{aligned}$$

The inner-loop statement is performed N^3 times with different i, j, k, each time accessing array variables $A(i,k)$, $B(k,j)$, and $C(i,j)$. The associated DAG can be embedded in three-dimensional Euclidean space, identifying each inner-loop statement as a point (i,j,k) and identifying its operands (entries of A, B, C) by its projections $(i,k), (k,j), (i,j)$ onto the three coordinate planes.

Irony-Toledo-Tiskin used this geometric interpretation of matrix multiplication to apply an inequality of Loomis-Whitney [24] to bound above the number of inner-loop statements doable with a fixed number of operands, enabling a simplification of Hong-Kung's graph-partitioning problem. Ballard-Demmel-Holtz-Schwartz extended these lower bounds to a more general class of matrix computations like LU-factorization that can be reduced to matrix multiplication. Of practical importance, these communication lower bounds indicated that a number of algorithms in numerical linear algebra had suboptimal communication costs, motivating a number of efforts to develop new algorithms and implementations (see [3] for a survey).

Our main contribution is to model a more general class of nested-loop programs than the ones considered by Ballard-Demmel-Holtz-Schwartz. In particular, we will allow any number of nested loops (instead of three) and any number of arrays (instead of three) each of any dimension (instead of two), and we will allow the arrays to be indexed by arbitrary linear combinations of the loop indices (instead of projections onto coordinate hyperplanes). The key to this extension is replacing Loomis-Whitney's inequality with a generalization proposed by Brascamp-Lieb [12] and analyzed by Bennett-Carbery-Christ-Tao [9].

1.3 Contributions

The main contributions of this work are as follows:

- New shared-memory parallel variant of Hong-Kung's sequential communication lower bound approach [21] — see Section 2.3;
- New communication lower bounds for a class of algorithms whose dependence structures model nested-loop programs with linear array index expressions, generalizing the 'matrix-multiplication-like' class studied by Irony-Toledo-Tiskin [22] and Ballard-Demmel-Holtz-Schwartz [5] — see Section 3.2;
- Sharper constants in *Hölder-Brascamp-Lieb-type inequalities*, extending work by Bennett-Carbery-Christ-Tao [9] and enabling our new communication lower bounds — see Section 4.2;
- New algorithm to compute constraints for Hölder-Brascamp-Lieb-type inequalities, extending work by Valdimarsson [34] and showing that communication lower bounds are computable — see Section 4.2; and
- New communication-optimal implementations, which attain communication lower bounds by *tiling*, in many cases of practical interest — see Section 5.2.

1.4 Outline

A chapter-by-chapter outline of this work is as follows.

In the remainder of Chapter 1 we review the mathematical notation used subsequently.

In Chapter 2 we model the execution of an idealized shared-memory parallel computer, where each processor has a private local memory and all processors share a global memory, and we define communication as data movement between local and global memory. We use *input-* and *output-paths*, structural properties of algorithms, to derive lower bounds on the amount of communication in any implementation of a given algorithm.

In Chapter 3 we define *Hölder-Brascamp-Lieb (HBL) interpretations* of algorithms which provide an intuitive way to model a class of nested-loop programs with certain input-/output-path properties. Recall the earlier example of multiplying N-by-N matrices, $C = A \cdot B$,

$$\begin{aligned} &\textbf{for } i = 1 : N, \quad \textbf{for } j = 1 : N, \quad \textbf{for } k = 1 : N, \\ &\quad C(i,j) = C(i,j) + A(i,k) * B(k,j); \end{aligned}$$

modeling this program as a DAG, we will see that any inner-loop statement (i, j, k) has incoming paths from input array variables $A(i, j)$ and $B(k, j)$ and an outgoing path to output array variable $C(i, j)$. These input-/output-paths model data dependences, which we use to derive lower bounds on data movement (communication).

In Chapter 4, we study HBL-type inequalities, for later use analyzing algorithms with HBL interpretations. We extend previous work [9] on HBL-type inequalities by demonstrating sharper constants, as well as extend previous work [34] on computability issues related to these inequalities.

Finally, in Chapter 5, we apply the HBL theory to complete our communication lower bound analysis of algorithms with HBL interpretations. We also show, in some important special cases, that these communication lower bounds are asymptotically attainable via *tiling*.

Conclusions are drawn and future work is reviewed in Chapter 6.

1.5 Notation

We employ standard mathematical notations with a few non-standard extensions.

The set of natural numbers is denoted by $\mathbb{N} = \{0, 1, 2, \ldots\}$; for any $n \in \mathbb{N}$, we let $\mathbb{N}_{\geq n} = \{n, n+1, \ldots, \}$, so $\mathbb{N}_{\geq 0} = \mathbb{N}$, and we let $[n] = \{1, \ldots, n\}$, so $[0] = \emptyset$.

For any set X, its cardinality $|X| \in \mathbb{N} \cup \{\infty\}$. For any $n \in \mathbb{N}$, X^n denotes the n-th Cartesian power of X; more generally, if Y is another set, then X^Y denotes the set of all functions from Y to X.

An Abelian group is a set G that is closed under addition $+$, an associative and commutative binary operation, with identity element 0 and an inverse element $-x$ for each $x \in G$. A group homomorphism is a function between Abelian groups that is compatible with the group structure. The rank of an Abelian group is the maximum cardinality of any linearly independent subset of G.

Chapter 2

Modeling Computation and Communication

In this chapter we formalize our computation model in order to reason mathematically about the communication costs of algorithms. A section-by-section outline is as follows.

- In Section 2.1, we model a computation as an execution of an idealized parallel machine with a two-level memory hierarchy, where each processor has a private local memory and all processors share a global memory. Communication is data movement between the local memories and the global memory.
- In Section 2.2, we introduce our notion of *algorithms*, which correspond to sets of machine executions called *implementations*.
- Then in Section 2.3, we use *input-* and *output-paths*, structural properties of algorithms, to derive lower bounds on the amount of communication in any implementation of a given algorithm.
- We discuss related work in Section 2.4: our model and communication lower bounds approach was inspired by Hong-Kung's seminal work [21].

2.1 Model Preliminaries

Machine Model A *machine* has a nonempty finite set of *processors*, each connected to a *local memory* and all connected to a *global memory* (see Figure 2.1). A memory contains storage elements called *cells*, each of which stores a single *value*; local memories have finitely many cells, while global memory can have infinitely many. Local memory is *private*, i.e., each processor can access its own, and only its own. On the other hand, any processor can access global memory, i.e., global memory is *shared*. Processors can only perform (arithmetic/logical) *operations* on local data, and it is often necessary to *communicate* (move data) between local and global memory. For example, communication is necessary when a processor's working set becomes sufficiently large relative to its (bounded) local memory size — our main results concern this particular computation/communication relationship.

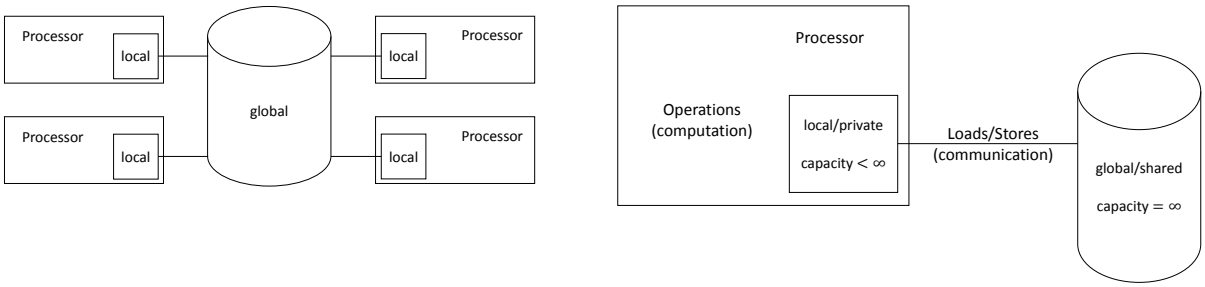


Figure 2.1: Model machine.

While values and operations will remain tangential to our analysis, it helps to have concrete definitions of these terms: a value is an element of some given nonempty set D and an operation is a function $f\colon D^m \to D^n$ for some $m, n \in \mathbb{N}_{\geq 1}$, an element of some given nonempty set B of such functions. Our machine model is thus implicitly parameterized by D and B.

Execution Model An *execution* comprises a finite ordered set of *instructions* and an *assignment* of (a function from) instructions to processors. Each processor can be instructed to load a value from global to local memory (*Load*), to store a value from local to global memory (*Store*), and to perform an operation on values in local memory (*Operation*). (An execution gives an a posteriori perspective, so there is no need to model branches, only operations and data movement.)

The *execution order*, a partial order, defines a temporal precedence relation: if two instructions are comparable then they were not issued concurrently. In particular, the execution order enforces two model assumptions regarding parallel execution:

1. "Sequential processors": each processor performs one instruction at a time in a given total order. That is, each processor's instructions compose a chain (totally ordered subset) in the execution order — see Figure 2.2 (left).

2. "Concurrent reads, exclusive writes, and no data races": each memory cell is associated with a total order that contains all pairs of reads and writes in which at least one element in the pair is a write. That is, each global cell's associated Stores compose a chain in the execution order which remains a chain when including any one of that cell's associated Loads — see Figure 2.2 (right).

Note that assumption 2 neglects local memory accesses because assumption 1 implies the stronger constraint that local reads (in addition to local writes) are exclusive.

A *schedule* of an execution defines *start* and *end times* (real numbers) for each instruction such that each instruction's start time follows its predecessors' end times. Thus, an execution

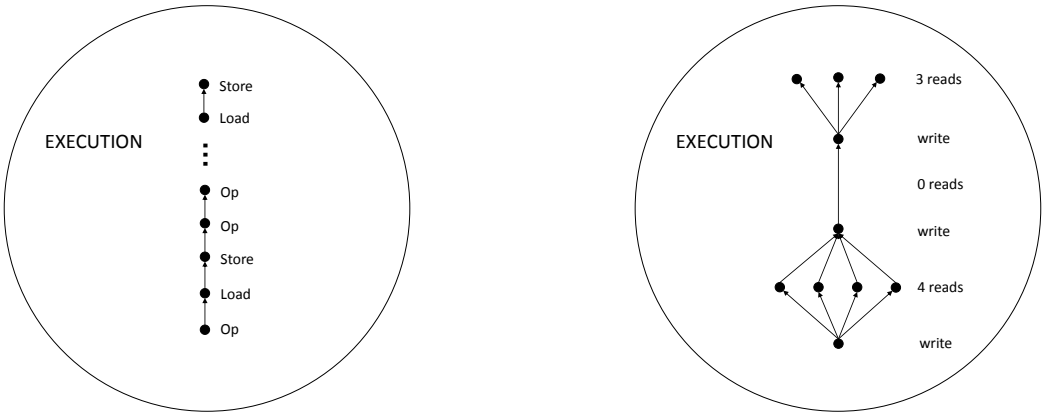


Figure 2.2: Execution assumptions. Left: some processor's instructions ("sequential processors"); right: some cell's reads/writes ("concurrent reads, exclusive writes, and no data races").

order enforces the two preceding model assumptions by constraining the set of permissible schedules (notably, an execution itself has no notion of time).

Cost Model Suppose each instruction in an execution is assigned a nonnegative *cost*. We then define the *cost of a chain* in the execution as the sum of its instructions' costs; as a special case, we define the *cost of a processor* as the cost of that processor's chain. We also define the *critical-path cost* as the maximum cost over all chains (not necessarily a single processor's) in the execution order. Lastly, we define the *total cost* as the sum of all instructions' costs.

In this work we consider simple 0/1-cost assignments to count different types of instructions: for each chain in the execution order, we define its *computation cost* to be its number of Operations and its *communication cost* to be its number of Loads and Stores. The computation/communication tradeoffs mentioned above will be quantified in Section 2.3 in terms of these two costs.

We detour to show how three physical costs — time, energy, and power — can also modeled in this framework.

First, suppose that the instructions' costs are their *time-costs* (durations): we can model runtime with the critical-path time-cost. To see this connection, consider the *greedy schedule*, constructed by recursively assigning each instruction a start time equal to the maximum of its predecessors' end times (zero if none exist) and an end time equal to its start time plus its time-cost. This approach may introduce *idle time* — e.g., if processor p stores a value and processor q subsequently loads it, then q may be kept idle while the Store completes — but there always exists a critical path with no idle time. For any time-cost assignment of an execution, according to its greedy schedule, the length of the interval between zero and the maximum end time equals the critical-path time-cost.

Second, suppose that the instructions' costs are their *energy-costs* (amounts of energy consumed): we can model total energy consumption (or some processor p's energy consump-

tion) by the total energy-cost (or p's energy-cost).

Third, if we are given both time- and energy-cost assignments, then we can model average power of the machine (or processor p) by dividing the total energy-cost (or p's energy-cost) by the critical-path time-cost.

The point of this digression was to demonstrate that our simple cost model, counting instructions by type, can in some cases yield quantitative insights into physical costs: e.g., if Operations have a fixed time-cost and Loads/Stores have a fixed time-cost, then (modeled) runtime is bounded above/below by the sum/maximum of the critical-path computation and communication costs (this lower bound assumes complete overlap of computation and communication).

2.2 Analyzing Executions

We now introduce some notation and bookkeeping tools that will aid our subsequent analyses. Our analytical approach is independent of the particular values arising during execution, as well as the particular operations performed on them. Rather, our analysis depends on the particular cells each instruction accesses (reads from and writes to), and the order in which they do so.

Instruction Encoding As mentioned, for each instruction i in a given execution, we need only observe the cells it accesses. Each instruction i reads values from its *input cells* $a_1, \ldots, a_m$ (not necessarily distinct) and writes values to its *output cells* $b_1, \ldots, b_n$ (necessarily distinct). We define $\mathrm{In}(i) = \{a_1, \ldots, a_m\}$ and $\mathrm{Out}(i) = \{b_1, \ldots, b_n\}$. If i is a Load (resp., Store), then $m = n = 1$, and a_1 is a global (resp., local) cell and b_1 is a local (resp., global) cell. If i is an Operation, then $m, n \in \mathbb{N}_{\geq 1}$ and all cells are local.

Data Dependence An execution implies the manner in which individual values propagate: instructions depend on values created/copied by their predecessors in the execution order. Since new values can only be introduced by Operations — Loads and Stores only make copies of existing values — a value's *lifetime* either begins with the Operation that created it, or otherwise the value resided in memory since the start of execution.

We now formalize the notion of a lifetime as a chain in the execution order; keep in mind that this definition will depend only on the cells associated with the instructions and not the particular values they store: for example, the same lifetimes are constructed even if only one distinct value arises in the execution. In this sense, data dependence is a data-independent notion.

Consider any instruction i: we first define the lifetimes of i's output values and second the lifetimes of i's input values.

- The lifetime of the value written to some (local or global) cell $c \in \mathrm{Out}(i)$ by some instruction i is a chain in the execution order that ends with i. If i is an Operation, then we return the singleton chain (i). Otherwise, i was a Load or Store, thus has a unique $b \in \mathrm{In}(i)$; we recursively compute the lifetime $(i_1, \ldots, i_n)$ of the value read from

$b \in \text{In}(i)$ by i according to the procedure in the following bullet, and then return the chain $(i_1, \ldots, i_n, i)$.

- The lifetime of the value read from some (local or global) cell $c \in \text{In}(i)$ by some instruction i is a chain in the execution order, possibly empty. There are two possibilities: either some ancestor j of i wrote this value to c (i.e., $c \in \text{Out}(j)$), or no such ancestor j of i exists. In the latter case, the value initially stored in c was never altered prior to being read by i, which we indicate by returning the empty chain $()$. In the former case, due to the "concurrent reads, exclusive writes, and no data races" assumption defined in Section 2.1, we can identify the unique maximal (most recent) ancestor j of i such that $c \in \text{Out}(j)$. If j is an Operation, then we return the singleton chain (j). Otherwise, j was a Load or Store, thus has a unique $b \in \text{In}(j)$; we recursively compute the lifetime $(j_1, \ldots, j_n)$ of the value read from $b \in \text{In}(j)$ by j according to the procedure in the preceding bullet, and then return the chain $(j_1, \ldots, j_n, j)$.

Since there are finitely many instructions in the execution, this (downwards) recursion terminates with a finite chain, possibly empty, consisting entirely of Loads and Stores except possibly for its initial element. In this manner, we can trace the lifetime of every value that is read or written by an instruction.

When additionally given a schedule of the execution, we can extend this procedure to define the lifetime of the value stored in any cell c at any particular point t in time, provided that c is not being (over-) written at time t. That is, since our model implies that c stores a consistent value between the (totally ordered) instructions that write to c, we can apply the procedure to the most recent (before t) instruction i with $c \in \text{Out}(i)$, or returning $()$ if none exists. This observation also emphasizes that lifetimes are independent of the schedule.

Algorithms Each execution on a given machine *implements* an *algorithm*, an abstraction that ignores mechanical details to focus on the dependences between values. In Section 2.3, we will derive lower bounds on communication cost in terms of algorithms: these lower bounds apply to all executions which implement that algorithm. We will first define algorithms without reference to executions and then demonstrate how executions implement algorithms. Note that the following (non-standard) definition of 'algorithm' is more abstract than the standard definition (a computable function, Turing machine, etc.). We use the standard definition in Chapter 4 in the context of decision problems (see Theorems 4.3 and 4.4), but use the non-standard definition exclusively in all other chapters.

An algorithm is a finite set of *(function) evaluations*, symbolic expressions of the form

$$(y_1, \ldots, y_n) = f(x_1, \ldots, x_m)$$

for various $m, n \in \mathbb{N}_{\geq 1}$. The symbol f represents an operation with input arity m and output arity n; we suppose that f is distinct in each function evaluation and let F be the set of these symbols. The tuples $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_n)$ contain symbols called *variables*; we let V be the set of these symbols and suppose that $V \cap F = \emptyset$. Additionally, a variable can only appear once as some output y_j ("single assignment form"), but can appear more than once as an input $x_{i_1}, x_{i_2}, \ldots$. To formalize this constraint, we introduce f-subscripts to

disambiguate between evaluations, $x_{f,1}, \ldots, x_{f,m_f}$ and $y_{f,1}, \ldots, y_{f,n_f}$, i.e., we define the two functions

$$x\colon \{(f,i) \mid f \in F \wedge i \in [m_f]\} \to V$$
$$y\colon \{(f,j) \mid f \in F \wedge j \in [n_f]\} \to V.$$

With this notation, the 'single assignment form' assumption means that y is an injection while x need not be. The elements of $\mathrm{dom}(x)$ and $\mathrm{dom}(y)$ are called *input* and *output arguments*, resp., and the elements of $\mathrm{im}(x)$ and $\mathrm{im}(y)$ are called *input* and *output argument variables*, resp. (by definition, $V = \mathrm{im}(x) \cup \mathrm{im}(y)$). By injectivity of y (and not of x), each $v \in V$ is an output argument variable $y_{f,j}$ for at most one output argument (f,j), but can be an input argument variable $x_{f_1,i_1}, x_{f_2,i_2}, \ldots$ for multiple input arguments $(f_1,i_1),(f_2,i_2),\ldots$.

Besides V, F, x, y, an algorithm has designated *input* and *output variables* $I, O \subseteq V$, where $I = \mathrm{im}(x) \setminus \mathrm{im}(y)$ and $\mathrm{im}(y) \setminus \mathrm{im}(x) \subseteq O \subseteq \mathrm{im}(y)$. Here we observe an asymmetry in our model concerning inputs and outputs, i.e., that I is completely specified by x, y but O is not necessarily; by convention, we assume $O = \mathrm{im}(y) \setminus \mathrm{im}(x)$ when O is unspecified.

The functions x and y imply a partial order on the set of evaluations. The *algorithm DAG* (V,E) has the variables as vertices and has edge set $E = \bigcup_{f \in F} \{x_{f,1}, \ldots, x_{f,m_f}\} \times \{y_{f,1}, \ldots, y_{f,n_f}\}$; the input variables I are the indegree-0 vertices and the output variables O include the outdegree-0 vertices.

Implementations Next we show how to construct an algorithm from an execution that implements it. Before doing so, we explain how this approach deviates from Hong-Kung's [21].

Hong-Kung start with algorithms modeled by DAGs (Hong-Kung's DAGs are equivalent to our algorithm DAGs in the special case $n_f = 1$ for all $f \in F$, except for the definition of input variables I) and then define an algorithm's implementations as the different gameplays of the 'red/blue pebble game' on that DAG. (In the red/blue pebble game, defined in [21, Section 1], red and blue pebbles, which model local and global memory cells, resp., are placed on the vertices of the DAG, modeling storing the associated values in the associated cells.) Our approach starts with executions and then defines the mapping from executions to the algorithms they implement: in Hong-Kung's framework, this is like starting with a gameplay and reconstructing the DAG by studying the gameplay's moves. Our approach was motivated by complications we encountered while extending Hong-Kung's approach to the parallel setting (Hong-Kung modeled sequential machines). We note that our approach is more general than Hong-Kung's by allowing certain 'wasteful' implementations, e.g., those including Operations whose outputs are subsequently overwritten — however, we have not seen any theoretical benefit from this generality.

Roughly speaking, we construct an algorithm from an execution by identifying Operations and values in the execution with function evaluations and variables in the algorithm.

We consider each cell c (local and global) in turn. Compute the lifetime of the value stored in c at the end of execution: that is, identify the last instruction i in the execution order with $c \in \mathrm{Out}(i)$ and compute the lifetime of the value i wrote to c, or otherwise if no instructions wrote to c, return the empty chain () as the lifetime. Now examine the lifetime $(i_1, \ldots, i_t)$ just computed and consider two cases; we encapsulate our processing of

these cases in a subprocedure which will be reentered recursively with additional parameters denoted g and h.

- If the lifetime is the empty chain or if i_1 is a Load or Store, then we do nothing unless this is a recursive call. If this is a recursive call, we augment V with the variable c (identified with the cell c) and define $x_{g,h} = c$.

- Otherwise, i_1 is an Operation. If i_1 has been encountered already, let $f \in F$ denote its index; otherwise, augment F with a new symbol $f \in F$ that indexes i_1. Let m_f and n_f denote the number of i_1's input and output cells. We can identify exactly one $k \in [n_f]$ such that the lifetime began when i_1 wrote to its k-th output cell $b_k \in \mathrm{Out}(i_1)$. If the variable (f, k) does not yet exist in V, augment V with the variable (f, k) and define $y_{f,k} = (f, k)$. Additionally, if this is a recursive call, define $x_{g,h} = (f, k)$; otherwise, if it is not, designate (f, k) as an output variable, $(f, k) \in O$.

  Now, for each $i \in [m_f]$, trace the lifetime $(j_1, \ldots, j_q)$ of the value that i_1 read from its i-th input cell $a_i \in \mathrm{In}(i_1)$ and recursively evaluate this subprocedure substituting a_i for c and $(j_1, \ldots, j_q)$ for $(i_1, \ldots, i_t)$, and setting the parameters $g = f$ and $h = k$. Each of these recursive calls must terminate because the execution is finite.

We then proceed with the next cell c.

After all (potentially infinitely many) cells have been processed in this manner, we have constructed the objects V, F, x, y, O, where V, F are finite; recall that the input variables are completely defined by x, y: $I = \mathrm{im}(x) \setminus \mathrm{im}(y)$. The constructed objects V, F, x, y, I, O may fail to define an algorithm for a subtle reason: some function evaluations may not have had variables defined for each of their output arguments. Suppose, for example, that function evaluation $f \in F$ has an undefined variable $y_{f,k}$ for its k-th output argument (f, k): this means that the associated Operation's k-th output cell b_k was subsequently overwritten by another instruction without any intervening reads. Due to the top-down recursive approach taken, the fact that $f \in F$ implies that at least one of f's output argument variables must be defined — Operations whose outputs are all overwritten are never added to the algorithm — so $n_f \geq 2$. Thus we can simply delete f's k-th output argument, reindex the output argument $(f, j+1) \mapsto (f, j)$ and output argument variables $y_{f,j+1} \mapsto y_{f,j}$ for each $j \in \{k, \ldots, n_f - 1\}$, and redefine $n_f \mapsto n_f - 1 \geq 1$ (this alters F, y but not V, x, I, O).

After performing the above postprocessing step, we obtain an algorithm which we say is the one implemented by the given execution; every execution implements an algorithm, while an algorithm has many implementations. This construction also emphasizes an earlier remark that our model does not capture branching: if a conditional branch caused a different control path to be taken, then the corresponding executions would be mapped to different (branch-free) algorithms.

2.3 Communication Lower Bounds

We now explore a connection between an algorithm's DAG and its implementations' communication costs. (Communication costs were defined in Section 2.1 and algorithms/implementations were defined in Section 2.2.) A similar connection was observed by Hong-Kung [21] — we

review related work in Section 2.4. In this section, we assume a fixed algorithm, specified by the parameters V, F, x, y, I, O and inducing the algorithm DAG $G = (V, E)$, as defined in Section 2.2.

We first define two graph-theoretic properties called *input-* and *output-path cutsize*. For any $K \subseteq F$, let $\Pi_{\text{in}}(K)$ denote the set of all paths in G from I to $\{x_{f,1}, \ldots, x_{f,m_f} \mid f \in K\}$, called *input-paths*, and let $\Pi_{\text{out}}(K)$ denote the set of all paths in G from $\{y_{f,1}, \ldots, y_{f,n_f} \mid f \in K\}$ to O, called *output-paths*. (An input or output path can be a single element of I or O.) Let $Q_{\text{in}}(K)$ and $Q_{\text{out}}(K)$ denote the maximum cardinalities of subsets of $\Pi_{\text{in}}(K)$ and $\Pi_{\text{out}}(K)$ comprising only disjoint paths: we call $Q_{\text{in}}(K)$ and $Q_{\text{out}}(K)$ the *input-* and *output-path cutsizes* of K.

We observe two immediate properties of $\Pi_{\text{in}}, \Pi_{\text{out}}, Q_{\text{in}}, Q_{\text{out}}$. First, if $K = \emptyset$, then $Q_{\text{in}}(K) = Q_{\text{out}}(K) = 0$; otherwise, if $\emptyset \neq K \subseteq F$, $1 \leq Q_{\text{in}}(K) \leq \sum_{f \in K} m_f$ and $1 \leq Q_{\text{out}}(K) \leq \sum_{f \in K} n_f$. Second, for any $E_1, E_2 \subseteq F$, $E_1 \subseteq E_2$ implies that $\Pi_{\text{in}}(E_1) \subseteq \Pi_{\text{in}}(E_2)$ and $\Pi_{\text{out}}(E_1) \subseteq \Pi_{\text{out}}(E_2)$, therefore $Q_{\text{in}}(E_1) \leq Q_{\text{in}}(E_2)$ and $Q_{\text{out}}(E_1) \leq Q_{\text{out}}(E_2)$.

Now consider any implementation of the algorithm fixed above. Let P denote the set of processors; for each processor $p \in P$, let $M_p \in \mathbb{N}_{\geq 1}$ be processor p's (finite) local memory size. (These machine parameters are implied by the implementation.) For each $p \in P$, let $F_p \subseteq F$ denote the set of evaluations (in the algorithm) whose corresponding Operations (in the execution) are assigned to processor p. Continuing notation, we have the following connection between input-/output-path cutsize and communication.

Lemma 2.1. *For each $p \in P$ and each nonempty $K \subseteq F_p$, processor p performs at least $Q_{\text{in}}(K) - M_p$ Loads and at least $Q_{\text{out}}(K) - M_p$ Stores between the first and last Operations corresponding to K.*

Proof. See Section 2.3.1.

The conclusion of Lemma 2.1 is a set of lower bounds on W_p, processor p's communication cost — the total number of Loads and Stores it performs — for each $p \in P$. Any of these per-processor communication costs implies a lower bound on W, the critical-path communication cost; in particular,

$$W \geq \max_{p \in P} W_p.$$

When applying Lemma 2.1 to derive lower bounds on W_p for any $p \in P$, by monotonicity of $Q_{\text{in}}, Q_{\text{out}}$, there is no loss of generality to consider only the case $K = F_p$, i.e.,

$$\begin{aligned} W_p &\geq \max_{K \subseteq F_p} \Big(\max(0, Q_{\text{in}}(K) - M_p) + \max(0, Q_{\text{out}}(K) - M_p) \Big) \\ &= \max(0, Q_{\text{in}}(F_p) - M_p) + \max(0, Q_{\text{out}}(F_p) - M_p). \end{aligned}$$

Additionally, we can tighten this lower bound given more information about the contents of the processor p's local memory: for example, following [21] and requiring that all input/output values reside in global memory before/after execution, we can show that $W_p \geq Q_{\text{in}}(F_p) + Q_{\text{out}}(F_p)$ for all $p \in P$. Lower bounds of this form have been called memory-independent (see, e.g., [4]). Our goal is to derive lower bounds on each W_p that are memory-dependent, i.e., parameterized by M_p.

The following result, the main result of Section 2.3, applies Lemma 2.1 over a sequence of subsets of F_p to obtain a lower bound on W_p. We also adopt a stronger hypothesis regarding the input- and output-path cutsizes of these subsets.

Theorem 2.1. *For any $p \in P$, if there exists $K \subseteq F_p$ such that $N = |K| > 0$ and a nondecreasing $q\colon [N] \to \mathbb{R}$ such that $q(|J|) \leq Q_{\text{in}}(J) + Q_{\text{out}}(J)$ for all nonempty $J \subseteq K$, then processor p performs*

$$W_p \geq \max_{l \in \mathbb{N}_{\geq 1}} l \cdot \left(\left\lceil \frac{N}{\max\left\{ i \in [N] \,\middle|\, \lceil q(i) \rceil \leq 2M_p + l \right\}} \right\rceil - 1 \right) \tag{2.1}$$

Loads and Stores between the first and last Operations corresponding to K.

Proof. See Section 2.3.1.

In particular, the two maxima in (2.1) are well-defined. We show after proving Theorem 2.1 how its proof can be adjusted to exploit individual lower bounds on $Q_{\text{in}}, Q_{\text{out}}$ to derive similar conclusions for Loads and Stores independently (as in Lemma 2.1).

In this work, we will use the following specialization and simplification of Theorem 2.1.

Corollary 2.1. *If the hypotheses of Theorem 2.1 are satisfied by $q(i) = i^{1/\sigma}$ for some $\sigma \in [1, \infty)$, then the conclusion of Theorem 2.1 can be simplified: processor p performs at least*

$$W_p \geq \max_{l \in \mathbb{N}_{\geq 1}} l \left(\left\lceil \frac{N}{\lfloor (2M_p + l)^\sigma \rfloor} \right\rceil - 1 \right) \geq \begin{cases} \frac{1}{2 \cdot 3^\sigma} \frac{N}{M_p^{\sigma-1}} & \text{if } N \geq (4M_p)^\sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Loads and Stores between the first and last Operations corresponding to K.

Proof. See Section 2.3.1.

Corollary 2.1 addresses implementations of algorithms where, if processor p performs sufficiently many Operations N with respect to their local memory size M_p, then they must communicate as indicated by (2.2). Moreover, in this regime, there exists a lower-bound tradeoff between communication and memory, which we express in asymptotic notation,

$$W_p = \Omega\left(\frac{N}{M_p^{\sigma-1}} \right) \qquad \Rightarrow \qquad W_p \cdot M_p^{\sigma-1} = \Omega(N).$$

In Chapter 3, we will define and give examples of a class of algorithms to which Corollary 2.1 applies (the proof of this follows later, in Section 5.1). Then, in Chapter 5, we determine the parameter σ for the example algorithms in Chapter 3 as well as study the attainability of the communication lower bounds.

2.3.1 Proofs

Before proving Lemma 2.1, we give an informal alternative argument which ultimately fails for reasons related to an obstacle called 'R2/D2 arguments' by Ballard-Demmel-Holtz-Schwartz [5], also called 'paired Allocate/Frees statements of array variables' in our earlier work [15]. A contribution of the present work is a graph-theoretic reinterpretation of techniques both prior works employed to overcome this obstacle.

We continue the notation introduced earlier in Section 2.3: in particular, we consider some evaluations K assigned to some processor p. Let $X_K = \{x_{f,1}, \ldots, x_{f,m_f} \mid f \in K\}$ and $Y_K = \{y_{f,1}, \ldots, y_{f,n_f} \mid f \in K\}$ denote the associated input and output argument variables. Before the first and after the last element of its subchain K of assigned instructions, processor p can have at most M_p values in local memory. Since an implementation of an algorithm, by definition, treats variables as if their values were distinct, $|X_K| - M_p$ and $|Y_K| - M_p$ are perhaps intuitive candidates for lower bounds on the numbers of Loads and Stores processor p must perform. This intuition fails: some values associated with both X_K and Y_K could be temporary, meaning that processor p computes and discards (overwrites) them without incurring Loads and Stores, thus beating the 'lower bounds'. In addition to being quite common, this scenario is also desirable from the standpoint of minimizing communication (the practical motivation of our work).

To derive a lower bound on Loads and Stores in terms of numbers of input and output argument variables, we need to account for temporaries. Our accounting technique, codified in the definitions of $Q_{\text{in}}, Q_{\text{out}}$, is closely related to Ballard-Demmel-Holtz-Schwartz's usage of 'accumulators' [5, Section 2] and to 'surrogates', a generalization we introduced later in [15, Section 4.2]. These ideas, like many in the present work, have their roots in Hong-Kung's seminal paper [21], which we discuss in Section 2.4. Lastly, we point out a complementary accounting technique, a reduction approach called imposing reads and writes [5, Section 3.4] (see also [5, Corollary 5.1], [2, Section 4.2], and [15, Section 4.4]); developing a graph-theoretic analogue of this tool for use in the present model remains future work.

Proof of Lemma 2.1. Recall the notation defined in the paragraphs preceding the statement of Lemma 2.1. We consider any processor $p \in P$ with $F_p \neq \emptyset$ and any nonempty $K \subseteq F_p$. We let s and t denote the first and last instructions (both Operations) corresponding to K. Let $X_K = \{x_{f,1}, \ldots, x_{f,m_f} \mid f \in K\}$ and $Y_K = \{y_{f,1}, \ldots, y_{f,n_f} \mid f \in K\}$.

We first show that any path in the algorithm DAG from some $u \in I$ to some $w \in X_K$ contains at least one vertex v whose associated value either resided in processor p's local memory immediately before s or was loaded by processor p between s and t. By definition, $w = x_{f,i}$ for some some evaluation (and Operation) $f \in F_p$ and $i \in [m_f]$. There are three possibilities for the value that f read from its input cell $a_i \in \text{In}(f)$, not necessarily mutually exclusive: it resided in processor p's local memory immediately before s, it was Loaded by processor p after s and before f, or it was written by processor p during some Operation $g \in F_p$ such that $s \leq g < f$. In the first two cases, we conclude with $v = w$. Otherwise, w's predecessor along the given path equals $x_{g,k}$ for some $k \in [m_g]$, and we repeat the same reasoning, replacing $x_{f,i}$ by $x_{g,k}$. We have made progress along the given path, and since the algorithm is finite, this recursion must terminate, and it does so having determined the desired vertex v.

We second show that any path in the algorithm DAG from some $u \in Y_K$ to some $w \in O$ contains at least one vertex v whose associated value either resided in processor p's local memory immediately after t or was stored by processor p between s and t. By definition, $u = y_{f,j}$ for some evaluation (and Operation) $f \in F_p$ and $j \in [1, \ldots, n_f]$. There are three possibilities for the value that f wrote to its output cell $b_j \in \text{Out}(f)$, not necessarily mutually exclusive: it resided in processor p's local memory immediately after t, it was Stored by processor p after f and before t, or it was read by processor p during some Operation $g \in F_p$ such that $f < g \leq t$. (It is impossible that this value was simply overwritten without intervening reads due to the definition of an implementation.) In the first two cases, we conclude with $v = u$. Otherwise, u's successor along the given path equals $y_{g,k}$ for some $k \in [n_g]$, and we repeat the same reasoning, replacing $y_{f,j}$ by $y_{g,k}$. We have made progress along the given path, and since the algorithm is finite, this recursion must terminate, and it does so having determined the desired vertex v.

Consider any $\Pi'_{\text{in}} \subseteq \Pi_{\text{in}}(K)$: each element of Π'_{in} contributes a variable w whose value either resides in processor p's local memory immediately before s or is Loaded by processor p between s and t, and at most M_p variables can satisfy the former case. If Π'_{in} is a set of disjoint paths, then these variables are distinct, so the number of processor p's Loads between s and t must be at least $\max(0, |\Pi'_{\text{in}}| - M_p\}$. Similarly, consider any $\Pi'_{\text{out}} \subseteq \Pi_{\text{out}}(K)$: each element of Π'_{out} contributes a variable w whose value either resides in processor p's local memory immediately after t or is Stored by processor p between s and t, and at most M_p variables can satisfy the former case. If Π'_{out} is a set of disjoint paths, then these variables are distinct, so the number of processor p's Stores between s and t must be at least $\max(0, |\Pi'_{\text{out}}| - M_p\}$. The desired conclusion follows from the fact that there exists a $\Pi'_{\text{in}} \subseteq \Pi_{\text{in}}(K)$ comprising $Q_{\text{in}}(K)$ disjoint paths and a $\Pi'_{\text{out}} \subseteq \Pi_{\text{out}}(K)$ comprising $Q_{\text{out}}(K)$ disjoint paths. □

Now it is time to prove our main lower bound result, Theorem 2.1. We use a simple counting argument which we call segmentation, in which we apply Lemma 2.1 to subchains of the chain K. A similar argument was introduced by Hong-Kung in their the proof of [21, Lemma 3.1] and can be seen in many works building on Hong-Kung's. A distinction between our segmentation argument and many prior ones is that we allow the segment size to vary, rather than fixing it equal to the local memory size (however, we later make a similar choice to obtain the simpler result in Corollary 2.1). This generalized segmentation argument can improve the constant factors in the lower bound: for example, recent work of Langou-Lowery [25] exploited this flexibility to improve the constants in the communication lower bounds for matrix multiplication derived previously in [22, 5].

Proof of Theorem 2.1. Consider the hypothesized $p \in P$ and $K \subseteq F_p$. Let's introduce some additional notation. For any subchain J of processor p's chain of instructions, let W_J denote the number of (processor p's) Loads and Stores in J, and let $\overline{J}$ denote all of processor p's assigned instructions between the first and last instruction of J, inclusive; with this notation, let $L = \overline{K}$. We will prove a stronger result, that the right-hand side of (2.1) is a lower bound on W_L, which in turn is the desired lower bound on W_p, processor p's communication cost.

For each $l \in \mathbb{N}_{\geq 1}$, there exists a unique $k_l \in \mathbb{N}_{\geq 1}$ such that L is the concatenation $L = L_1 \cdots L_{k_l}$ of k_l subchains L_j such that

$$l = W_{L_1} = W_{L_2} = \cdots = W_{L_{k_l - 1}} \geq W_{L_{k_l}} > 0.$$

Since $W_L = \sum_{j=1}^{k_l} W_{L_j}$, we have that $l(k_l - 1) \le W_L \le lk_l$, attaining equality in the upper bound when l divides W_L. Also note that $l(k_l - 1) = 0$ if and only if $k_l = 1$ if and only if $l \ge W_L$; since the execution is finite, $W_L < \infty$, and thus $l(k_l - 1)$ is positive for at most finitely many $l \in \mathbb{N}_{\ge 1}$. Therefore, considering all $l \in \mathbb{N}_{\ge 1}$,

$$W_L \ge \max_{l \in \mathbb{N}_{\ge 1}} l(k_l - 1). \tag{2.3}$$

We will now derive a lower bound on k_l. Fix any $l \in \mathbb{N}_{\ge 1}$ and consider the concatenation $L = L_1 \cdots L_{k_l}$ as defined above; this defines the concatenation $K = K_1 \cdots K_{k_l}$, where each $K_j = K \cap L_j$. Since $0 < N = |K| = \sum_{j=1}^{k_l} |K_j| \le k_l \max_{j=1}^{k_l} |K_j|$, and since $k_l \in \mathbb{Z}$,

$$k_l \ge \left\lceil \frac{N}{\max_{j=1}^{k_l} |K_j|} \right\rceil. \tag{2.4}$$

We now weaken this inequality by bounding $\max_{j=1}^{k_l} |K_j|$ in terms of the function q.

Consider any $j \in [k_l]$ such that $|K_j| > 0$ (there is at least one since $N > 0$): applying Lemma 2.1, we have that

$$W_{\overline{K_j}} \ge \max(0, Q_{\text{in}}(K_j) - M_p) + \max(0, Q_{\text{out}}(K_j) - M_p) \ge Q_{\text{in}}(K_j) + Q_{\text{out}}(K_j) - 2M_p;$$

since $\overline{K_j} \subseteq L_j$, $W_{\overline{K_j}} \le W_{L_j} \le l$, and it follows that $Q_{\text{in}}(K_j) + Q_{\text{out}}(K_j) \le 2M_p + l$. By hypothesis, for any nonempty $J \subseteq K$, $\lceil q(|J|) \rceil \le Q_{\text{in}}(J) + Q_{\text{out}}(J)$ (the ceiling function is implied in the hypothesis); therefore, setting $J = K_j$, $\lceil q(|K_j|) \rceil \le 2M_p + l$, and

$$\max_{j=1}^{k_l} |K_j| \le \max \left\{ i \in [N] \right\} \Big| \lceil q(i) \rceil \le 2M_p + l \Big\}, \tag{2.5}$$

where the maximum on the right-hand side is well defined since $|K_j| \le N$ and since q is nondecreasing.

We obtain the lower bound (2.1) by substituting (2.5) into (2.4) into (2.3) and noting that $W_p \ge W_L$. □

Now consider for the moment the two weaker hypotheses: $q(|J|) \le Q_{\text{in}}(J)$ for all $\emptyset \ne J \subseteq K$, or $q(|J|) \le Q_{\text{out}}(J)$ for all $\emptyset \ne J \subseteq K$. In either case, we can repeat the preceding argument (with a few adjustments), redefining W_J to count only Loads or Stores, and obtain the same lower bound as (2.1) except with $M_p + l$ appearing in place of $2M_p + l$. These individual bounds on Loads and Stores may be useful for the analysis of machines with asymmetric Load/Store costs: for example, Stores are more expensive on some memory devices, and some algorithms can account for this [13]. However, we will not distinguish Load/Store costs further in the present work.

Proof of Corollary 2.1. Continue the notation introduced at the beginning of the proof of Theorem 2.1. We first simplify the maximum in the denominator of the fractional term in the right-hand side of (2.1). For any $l \in \mathbb{N}_{\ge 1}$ and any $i \in [N]$,

$$\lceil i^{1/\sigma} \rceil \le 2M_p + l \quad \Leftrightarrow \quad i^{1/\sigma} \le 2M_p + l \quad \Leftrightarrow \quad i \le (2M_p + l)^\sigma \quad \Leftrightarrow \quad i \le \lfloor (2M_p + l)^\sigma \rfloor,$$

where the three forward implications are immediate from properties of the floor function, of exponentiation, and the fact that $i \in \mathbb{Z}$; to see that the fourth inequality implies the first, substitute $i = \lfloor (2M_p+l)^\sigma \rfloor$ into the first inequality: $\lceil \lfloor (2M_p+l)^\sigma \rfloor^{1/\sigma} \rceil \leq \lceil 2M_p+l \rceil = 2M_p+l$. Therefore,

$$\max \left\{ i \in [N] \} \,\middle|\, \lceil q(i) \rceil \leq 2M_p + l \right\} = \min(|K|, \lfloor (2M_p + l)^\sigma \rfloor),$$

and substituting into the right-hand side of (2.1),

$$\max_{l \in \mathbb{N}_{\geq 1}} l \left(\left\lceil \frac{N}{\max \left\{ i \in [N] \,\middle|\, \lceil q(i) \rceil \leq 2M_p + l \right\}} \right\rceil - 1 \right)$$
$$= \max_{l \in \mathbb{N}_{\geq 1}} l \left(\left\lceil \frac{N}{\min(|K|, \lfloor (2M_p + l)^\sigma \rfloor)} \right\rceil - 1 \right) = \max_{l \in \mathbb{N}_{\geq 1}} l \left(\left\lceil \frac{N}{\lfloor (2M_p + l)^\sigma \rfloor} \right\rceil - 1 \right);$$

this is the first (tighter) lower bound in (2.2).

To simplify further, we will weaken this lower bound by fixing $l = M_p \geq 1$:

$$\max_{l \in \mathbb{N}_{\geq 1}} l \left(\left\lceil \frac{N}{\lfloor (2M_p + l)^\sigma \rfloor} \right\rceil - 1 \right) \geq M_p \left(\left\lceil \frac{N}{\lfloor (3M_p)^\sigma \rfloor} \right\rceil - 1 \right);$$

supposing that $N > \lfloor (3M_p)^\sigma \rfloor$,

$$M_p \left(\left\lceil \frac{N}{\lfloor (3M_p)^\sigma \rfloor} \right\rceil - 1 \right) \geq M_p \cdot \frac{N}{2\lfloor (3M_p)^\sigma \rfloor} \geq \frac{1}{2 \cdot 3^\sigma} \cdot \frac{N}{M_p^{\sigma-1}}.$$

Note that the second (looser) lower bound in (2.2) assumes the simpler, stronger condition $N \geq (4M_p)^\sigma > \lfloor (3M_p)^\sigma \rfloor$. □

It is possible to strengthen the second conclusion of Corollary 2.1 by assuming stronger hypotheses regarding N and M_p, or by taking more care approximating the maximum argument l, rather than simply setting $l = M_p$.

2.4 Related Work

As mentioned previously, our work was influenced in several ways by Hong-Kung's paper [21]. (Note that the following observations about our model and Hong-Kung's are meant to be informal.) Hong-Kung's machine model is like a special case of ours with only one processor: roughly speaking, Hong-Kung's 'red/blue pebble game' model can be extended to ours by introducing pebbles in different shades of red, modeling different processors' local memories. However, to model a parallel computation, we must address contention for shared resources, like data races when accessing shared memory, and these issues did not arise in Hong-Kung's (sequential) model. Even in the sequential (one processor) case, there are a few technical distinctions between the models that complicate comparisons.

For example, in our model, Operations' output values can overwrite input values (sometimes called 'sliding pebbles'), and we allow values to reside in local memory before and after execution. Both of these extensions to Hong-Kung's model were considered by Savage [31], who also developed a stronger lower bound argument that synthesized the theories of space and communication costs.

Another substantial distinction between our model and Hong-Kung's is that ours disallows recomputation — each evaluation is associated with exactly one Operation in any implementation. Generally speaking, implementations with recomputation can potentially reduce communication cost, and it is future work to address recomputation within our model. We note that several practical applications of this work, like matrix multiplication, have the special property that each evaluation has a single successor in the algorithm DAG, and since there is no reason to recompute a value more times than it is read by its successors (in particular, there is no reason to recompute a vertex with outdegree-0), recomputation is unhelpful for reducing communication in these algorithms. Disallowing recomputation has become a popular assumption in the related literature; Elango-Rastello-Pouchet-Ramanujam-Sadayappan [18] recently showed that Hong-Kung's lower bound framework can be tightened to exploit this assumption.

Another improvement to Hong-Kung's approach, proposed in earlier conference talks by us [16] and by Langou-Lowery [25], is changing the segment size, the parameter l in Theorem 2.1, which we already noted can improve the lower bounds versus the typical choice of setting l to equal the local memory size. In the case of matrix multiplication, Langou-Lowery showed that this can sharpen the lower bound by a factor of 2.

Chapter 3

Modeling Loop Nests

In this chapter we use algorithms (defined in Section 2.2) to model loop nests that access arrays by linear functions of the loop iteration vector. A section-by-section outline is as follows.

- In Section 3.1, we give motivating examples of algorithms to develop intuition for how loop nests can be modeled by algorithms.
- In Section 3.2, we define *Hölder-Brascamp-Lieb (HBL) interpretations* of algorithms, which model the linear mapping between operations and variables in many loop nests; this is the main contribution of this chapter.
- Then in Section 3.3, we review related work, in particular the papers of Irony-Toledo-Tiskin [22] and of Ballard-Demmel-Holtz-Schwartz [5], which studied a special class of algorithms with HBL interpretations (matrix multiplication and related algorithms).

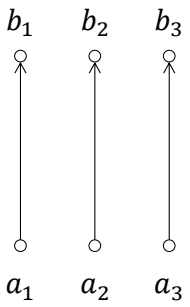
3.1 Motivating Examples

A practical motivation of this work is transforming computer programs, especially nested-loop programs. For example, multiplying N-by-N matrices, $C = A \cdot B$, can be programmed with triply nested loops (assuming C is zero-initialized),

$$\begin{aligned} &\textbf{for } i = 1 : N, \quad \textbf{for } j = 1 : N, \quad \textbf{for } k = 1 : N, \\ &\quad C(i,j) = C(i,j) + A(i,k) * B(k,j). \end{aligned}$$

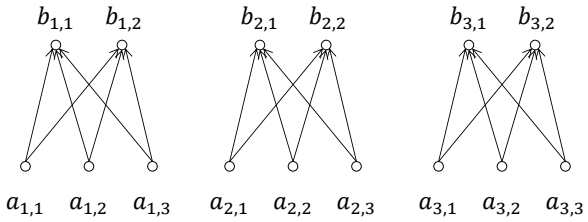
This program executes the inner-loop statement "$C(i,j) = C(i,j) + A(i,k) * B(k,j)$" exactly N^3 times, with different indices i, j, k — however, there is freedom to execute these statements in a different order besides the prescribed (lexicographical) order on $(i,j,k) \in [N]^3$ — and we would like to reorder the N^3 statements to expose data reuse. Our communication lower bounds apply to all reorderings, so if our implementation attains the lower bound, then there is no hope in trying other reorderings to further reduce communication — instead, we must seek a new algorithm.

To be clear, the preceding pseudocode programming language is meant to be informal, and we will not define a conversion between pseudocode and algorithms (as defined in Section 2.2); moreover, terms like "program", "loop", "statement", etc., are undefined in our



$$\left\{ b_i = f_i(a_i) \,\middle|\, i \in [N] \right\}$$

Figure 3.1: Example 1 (see Section 3.1.1). The DAG depicts the case $N = 3$.



$$\left\{ (b_{i,1}, \ldots, b_{i,n}) = f_i(a_{i,1}, \ldots, a_{i,m}) \,\middle|\, i \in [N] \right\}$$

Figure 3.2: Example 2 (see Section 3.1.2). The DAG depicts the case $N = 3$, $m = 3$, $n = 2$.

model. Ultimately, to apply this work to program transformations, we will need to formalize the conversion between some given program representation and an algorithm in our model. Elango-Rastello-Pouchet-Ramanujam-Sadayappan recently demonstrated such a conversion [18], paving the way for future applications of communication lower bounds to program transformations. We discuss program transformations abstractly (in terms of algorithms) in Chapter 5.

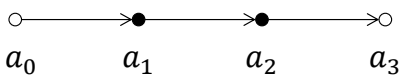
We begin with some motivating examples of algorithms with particular input- and output-path properties that we will characterize in a more general fashion in Section 3.2. In this examples, we will denote variables by subscripted expressions a_i, $b_{i,j}$, etc.; variables denoted in this manner are assumed to be distinct unless specified otherwise. We will also assume a given parameter $N \in \mathbb{N}_{\geq 1}$.

3.1.1 Example 1

Our first example is N independent unary operations — see Figure 3.1. The input- and output-paths in this algorithm's DAG are the singletons $\{a_i\}$ and $\{b_i\}$ for each $i \in [N]$, which are always disjoint. Thus, for any $K \subseteq F$, its input-path cutsize $Q_{\text{in}}(K)$ and output-path cutsize $Q_{\text{out}}(K)$ (defined in Section 2.3) both equal $|K|$.

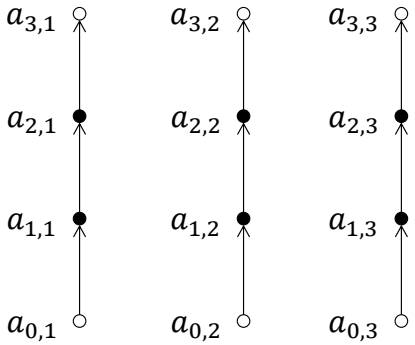
3.1.2 Example 2

For the second example, a variant on the first, we suppose each operation has input arity $m \in \mathbb{N}_{\geq 1}$ and output arity $n \in \mathbb{N}_{\geq 1}$ — see Figure 3.2. The input- and output-paths in



$$\left\{ a_i = f_i(a_{i-1}) \,\middle|\, i \in [N] \right\}$$

Figure 3.3: Example 3 (see Section 3.1.3). The DAG depicts the case $N = 3$.



$$\left\{ a_{i,j} = f_{i,j}(a_{i-1,j}) \,\middle|\, (i,j) \in [N]^2 \right\}$$

Figure 3.4: Example 4 (see Section 3.1.4). The DAG depicts the case $N = 3$.

this algorithm's DAG are still (disjoint) singletons, except that there are now more: for any $K \subseteq F$, $Q_{\text{in}}(K) = m|K|$ and $Q_{\text{out}}(K) = n|K|$.

3.1.3 Example 3

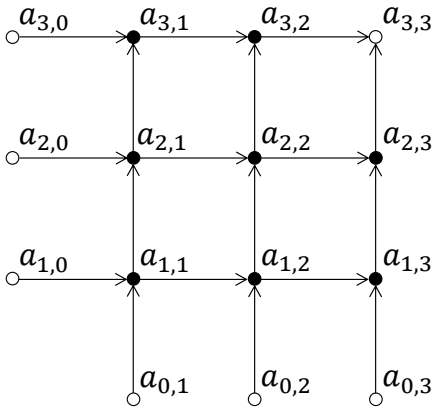
The third example is N completely dependent unary operations, whose induced (algorithm) DAG is a directed path — see Figure 3.3. Since there is only one input a_0 and one output a_N, there can be at most one disjoint input-path and output-path. Therefore, for any nonempty $K \subseteq F$, $Q_{\text{in}}(K) = Q_{\text{out}}(K) = 1$.

3.1.4 Example 4

The fourth example synthesizes the first and third examples, with N independent sets of N completely dependent unary operations — see Figure 3.4. In this case, each of the N paths cut (intersected) by K contributes exactly one to both the input- and output-path cutsize. That is, for any nonempty $K \subseteq F$, $Q_{\text{in}}(K) = Q_{\text{out}}(K) = |\{j \mid f_{i,j} \in K\}|$. For example, this number is one when $K = \{f_{1,1}, f_{2,1}, f_{3,1}\}$, but three when $K = \{f_{1,1}, f_{1,2}, f_{1,3}\}$.

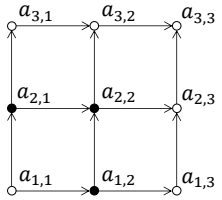
3.1.5 Example 5

The fifth example is a essentially a product of the third example with itself, defining a "diamond" DAG structure — see Figure 3.5. There is only one output variable: O was unspecified, so by convention $O = \text{im}(y) \setminus \text{im}(x) = \{a_{N,N}\}$. Thus there can be at most one disjoint output-path. On the other hand, the $2N$ input variables $I = \{a_{0,k}, a_{k,0} \mid k \in [N]\}$

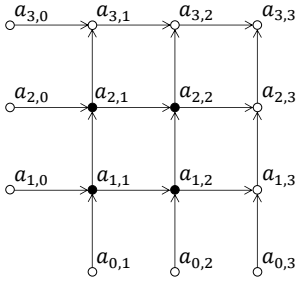


$$\left\{ a_{i,j} = f_{i,j}(a_{i-1,j}, a_{i,j-1}) \,\middle|\, (i,j) \in [N]^2 \right\}$$

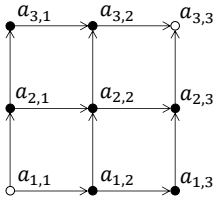
Figure 3.5: Example 5 (see Section 3.1.5). The DAG depicts the case $N = 3$.



Variant 1



Variant 2



Variant 3

Figure 3.6: Example 5 variants, DAGs for the case $N = 3$ (see Section 3.1.5).

are a great source of disjoint input-paths. Indeed, for any nonempty $K \subseteq F$,

$$Q_{\text{in}}(K) = |\{i \mid f_{i,j} \in K\}| + |\{j \mid f_{i,j} \in K\}|, \qquad Q_{\text{out}}(K) = 1.$$

The diamond DAG is a good example to demonstrate the dependence of our approach on input- and and output-paths. Consider the three variants of this example in Figure 3.6. Variant 1 flips the original (variant 0), with one input variable $I = \{a_{1,1}\}$ but $2N-1$ output variables $O = \{a_{N,k}, a_{k,N} \mid k \in [N]\}$: for any nonempty $K \subseteq F$,

$$Q_{\text{in}}(K) = 1, \qquad Q_{\text{out}}(K) = |\{i \mid f_{i,j} \in K\}| + |\{j \mid f_{i,j} \in K\}| - |\{k \mid f_{k,k} \in K\}|.$$

Variant 2 combines variants 0 and 1, with $I = \{a_{0,k}, a_{k,0} \mid k \in [N]\}$ and $O = \{a_{N,k}, a_{k,N} \mid k \in [N]\}$, so

$$\begin{aligned} Q_{\text{in}}(K) &= |\{i \mid f_{i,j} \in K\}| + |\{j \mid f_{i,j} \in K\}|, \\ Q_{\text{out}}(K) &= |\{i \mid f_{i,j} \in K\}| + |\{j \mid f_{i,j} \in K\}| - |\{k \mid f_{k,k} \in K\}|. \end{aligned}$$

Variant 3 has $I = \{a_{1,1}\}$ and $O = \{a_{N,N}\}$, so

$$Q_{\text{in}}(K) = Q_{\text{out}}(K) = 1.$$

We will see in Chapter 5 that our communication lower bounds are asymptotically attainable for variants 0, 1, and 2, but not for variant 3. For variant 3, our communication lower bound is zero, but it is possible to show that a nonzero lower bound exists when N is sufficiently large (see, e.g., [11]). Our approach inherits this weakness from Hong-Kung's approach: their lower bound [21, Theorem 3.1] is also trivial for this example (note that variant 0 is treated in [21, Section 7]). Savage [31] strengthened this aspect of Hong-Kung's approach by showing that Hong-Kung's hypothesis regarding dominator set size (analogous to Q_{in}) can be relaxed. While Savage's approach generally yields tighter lower bounds than Hong-Kung's for examples like variant 3 with few inputs and outputs, due to technical distinctions between Hong-Kung's and Savage's models — Savage allows an instruction's output value to overwrite one of its input values ("sliding pebbles"; see also [19]) and allows inputs/outputs to reside in local memory before/after execution — Savage's lower bounds can also be smaller than Hong-Kung's. Hong-Kung's and Savage's approaches were compared and extended by Bilardi-Pietracaprina-D'Alberto [11]; it is future work to incorporate these refinements in our approach.

These diamond DAGs generalize from two dimensions to $r \in \mathbb{N}_{\geq 2}$ dimensions: e.g., the original (variant 0) becomes

$$\left\{ a_{i_1,\ldots,i_r} = f_{i_1,\ldots,i_r}(a_{i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_r} \mid j \in [r]) \,\middle|\, (i_1,\ldots,i_r) \in [N]^r \right\}.$$

There is still only one output variable, $O = \{a_{N,\ldots,N}\}$, but there are now rN^{r-1} input variables $I = \{a_{i_1,\ldots,i_r} \mid (\exists k \in [r])\, i_k = 0\}$. For any nonempty $K \subseteq F$, $Q_{\text{out}}(K) = 1$ but

$$Q_{\text{in}}(K) = \sum_{j=1}^{r} \left| \left\{ (i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_r) \,\middle|\, f_{i_1,\ldots,i_r} \in K \right\} \right|.$$

Each summand (indexed by j) is the cardinality of a subset of $\mathbb{Z}^{r-1}$ obtained from a subset of $\mathbb{Z}^r$ by 'forgetting' the latter's elements' j-th components. These r projections, functions from $\mathbb{Z}^r$ to $\mathbb{Z}^{r-1}$, are special examples of group homomorphisms from the Abelian group $\mathbb{Z}^r$ to the Abelian group $\mathbb{Z}^{r-1}$. We will revisit this example in Section 5.5.5.

3.1.6 Example 6

The next example, called a *reduction*, means any algorithm whose DAG has a single output — see Figure 3.7. For any nonempty $K \subseteq F$, $1 \leq Q_{\text{in}}(K) \leq N$ and $Q_{\text{out}}(K) = 1$. In Figure 3.7, we also show two special reductions which attain the lower and upper bounds, respectively, on $Q_{\text{in}}(K)$, for any nonempty $K \subseteq F$.

Similarly to reductions we have *expansions* — see Figure 3.8 For any nonempty $K \subseteq F$, $Q_{\text{in}}(K) = 1$ and $1 \leq Q_{\text{out}}(K) \leq N$. In Figure 3.8, we show two special expansions which attain the lower and upper bounds, respectively, on $Q_{\text{out}}(K)$, for any nonempty $K \subseteq F$.

3.2 HBL Interpretations

In this section we define a group-theoretic interpretation for an algorithm called an *HBL (Hölder-Brascamp-Lieb) interpretation*, which will describe a broad range of algorithms, and

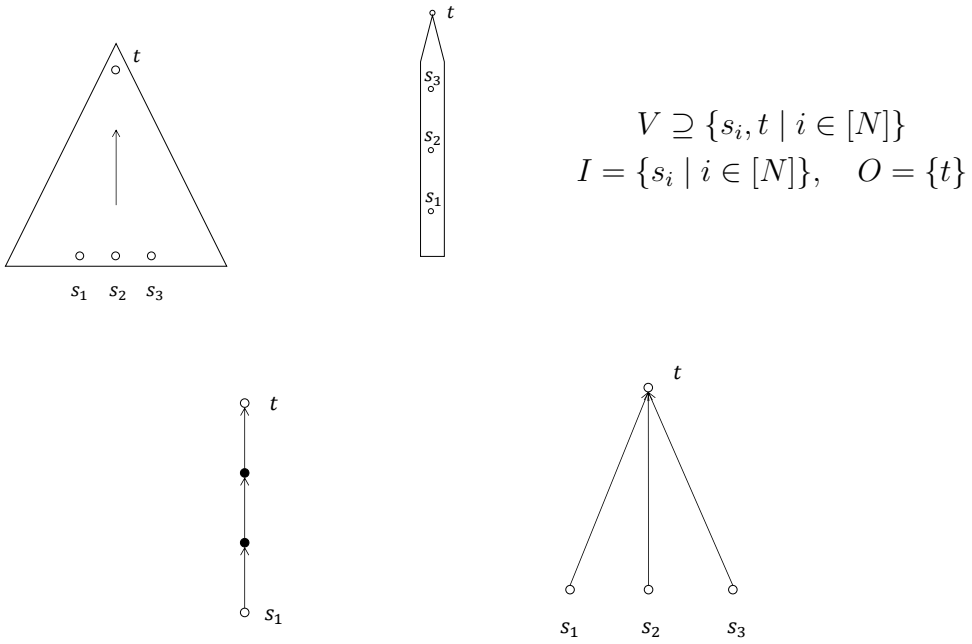


Figure 3.7: Example 6, reduction (see Section 3.1.6). The top row gives two cartoon characterizations of the corresponding class of DAGs (when $N = 3$) which are meant to be equivalent: the 'pencil' version is convenient in later examples (see Sections 5.5.2 and 5.5.3) with multiple reductions intersecting at different angles. The bottom row gives two concrete special cases (when $N = 3$) mentioned in Section 3.1.6.

let us systematically derive communication lower bounds and (sometimes) optimal implementations using techniques in Chapters 4 and 5.

Computational data is commonly structured as arrays (scalars, vectors, matrices, etc.). Given some $r \in \mathbb{N}$ and $Z \subseteq \mathbb{Z}^r$, an *array (of variables)* is an injective function $A\colon Z \to V$. This is an informal analogy for array data structures commonly found in programming languages, as well as more general data structures with indirection like "$B(\text{pointer}(i))$", provided that pointer$(\cdot)$ is injective; due to our abstract model, our arrays have no assumptions on rectangularity, type homogeneity, data-layout regularity, etc. When we consider multiple arrays, e.g., A_1, A_2, A_3, their elements may *alias*, e.g., $A_1(i) = A_2(i, j) = A_3(i, j, k)$ may all denote the same variable. We can disallow this possibility by *assuming no aliasing*, meaning that the images of all arrays are disjoint. However, as we will explain in Chapter 5, our communication lower bound analysis assumes maximal aliasing, meaning that the constants in our lower bounds could be sharpened in the absence of aliasing.

Consider any algorithm with parameters V, F, x, y, I, O and its DAG (V, E). Consider any

- K, nonempty subset of F,
- $m, r \in \{1, 2, \ldots\}$, $r_1, \ldots, r_m \in \{0, 1, \ldots\}$,
- Z, injection from K to $\mathbb{Z}^r$,

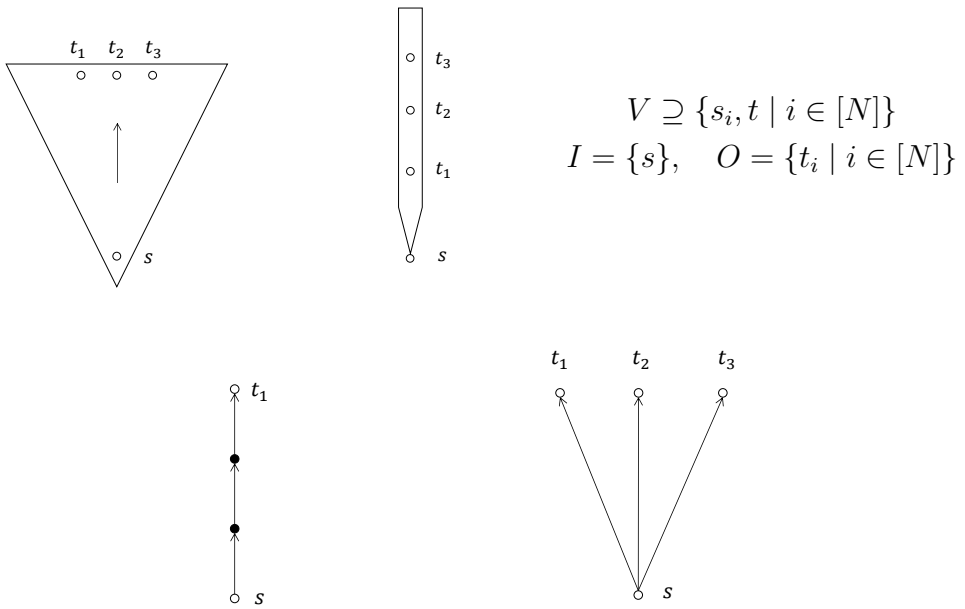


Figure 3.8: Example 6, expansion (Section 3.1.6). The top row gives two cartoon characterizations of the corresponding class of DAGs (when $N = 3$) which are meant to be equivalent: the 'pencil' version is convenient in later examples (see Sections 5.5.2 and 5.5.3) with multiple expansions intersecting at different angles. The bottom row gives two concrete special cases (when $N = 3$) mentioned in Section 3.1.6.

- $\phi_1, \ldots, \phi_m$, group homomorphisms from $\mathbb{Z}^r$ to $\mathbb{Z}^{r_1}, \ldots, \mathbb{Z}^{r_m}$, and
- $A_1, \ldots, A_m$, injections from $\phi_1(\text{im}(Z)), \ldots, \phi_m(\text{im}(Z))$ to $I \cup O$.

For each $\emptyset \neq J \subseteq K$, letting $A(J) = \bigcup_{j=1}^{m} A_j(\phi_j(Z(J)))$, suppose there are disjoint paths in (V, E)

- from each $u \in A(J) \cap I$ to distinct $v \in \bigcup_{f \in J} \{x_{f,1}, \ldots, x_{f,m_f}\}$ and
- to each $v \in A(J) \cap O$ from distinct $u \in \bigcup_{f \in J} \{y_{f,1}, \ldots, y_{f,n_f}\}$.

We say that the given algorithm admits an *HBL interpretation* (with respect to these parameters).

For example, we see that the diamond DAG in Section 3.1.5, in particular, the variant in Figure 3.5, admits an HBL interpretation with

$$K = \{f_{i,j} \mid (i,j) \in [N]^2\}, \quad m = r = 2, \quad r_1 = r_2 = 1 \quad , Z(f_{i,j}) = (i,j),$$
$$\phi_1(i,j) = i, \quad \phi_2(i,j) = j, \quad A_1(i) = a_{i,0}, \quad A_2(j) = a_{0,j}.$$

For any $\emptyset \neq J \subseteq K$, we can identify the desired set of disjoint paths by tracing each of the $2N$ inputs (positioned along the one of the coordinate axes in the plane) in a straight line until intersecting J (or ignore those inputs if J isn't intersected).

Having identified a set of disjoint input- and output-paths for each $\emptyset \neq J \subseteq K$, with each element of $A(J)$ included in a distinct path, we conclude that $Q_{\text{in}}(J) + Q_{\text{out}}(J) \geq |A(J)|$, the key property for proving the following result.

Theorem 3.1. *If* $\bigcap_{j=1}^{m} \ker(\phi_j) = \{0\}$*, then there exists* $\sigma \in [1, m]$ *such that*

$$|J|^{1/\sigma} \leq Q_{\text{in}}(J) + Q_{\text{out}}(J) \qquad \text{for all } \emptyset \neq J \subseteq K. \tag{3.1}$$

Proof. See Section 5.1.

Theorem 3.1 yields communication lower bounds via Corollary 2.1. That is, given an algorithm and HBL interpretation (continuing notation from earlier in Section 3.2), consider any implementation of that algorithm where some processor p, with M_p local memory cells, is assigned the $N = |K|$ Operations corresponding to the evaluations $K \subseteq F$. By Corollary 2.1, if $N \geq (4M_p)^{\sigma}$, then processor p performs at least

$$\frac{1}{2 \cdot 3^{\sigma}} \cdot \frac{N}{M_p^{\sigma-1}} = \Omega\left(\frac{N}{M_p^{\sigma-1}}\right)$$

Loads and Stores between their first and last Operation corresponding to K.

As we will see in Section 5.1, σ can be computed by solving the linear program,

$$\sigma = \min\left\{ \sum_{j=1}^{m} s_j \;\middle|\; s \in [0,\infty)^m \wedge (\forall H \leq \mathbb{Z}^r)\, \text{rank}(H) \leq \sum_{j=1}^{m} s_j \,\text{rank}(\phi_j(H)) \right\}. \tag{3.2}$$

3.3 Related Work

The class of algorithms studied in this chapter builds on earlier works by Irony-Toledo-Tiskin [22] and Ballard-Demmel-Holtz-Schwartz [5], which studied a special class of HBL interpretations, based on the Loomis-Whitney inequality — a special case of Hölder-Brascamp-Lieb-type inequalities, which we define next in Chapter 4 — in order to target matrix multiplication and related algorithms. While there are many similarities in the class of algorithms we consider, our models of computation have some key distinctions. In particular, our algorithm model is in some ways more constrained, avoiding the complication of 'R2/D2 operands' (mentioned in Section 2.3) at the cost of having to develop the input- and output-path-based analysis. We also address the "no interleaving" constraint in our earlier work [15] by allowing operations with multiple outputs (we are not aware of an earlier model that did this), thus enforcing atomicity of inner-loop statements.

Chapter 4

Hölder-Brascamp-Lieb-type Inequalities

Our communication lower bound analysis in Section 2.3 formalized the intuitive argument that a bounded number of operands implies a bounded number of operations can be performed. The key property of our target class of computations which enables this intuitive argument is that their operations and operands can be identified by tuples of integers, and the mapping from an operation to its operands can be modeled by linear functions between integer tuples. We exploit this algebraic relationship via the following result.

Theorem 4.1 ([15, Theorem 3.2])**.** *Suppose we are given a positive integer m, nonnegative integers $r, r_1, \ldots, r_m$, and Abelian group homomorphisms $\phi_1, \ldots, \phi_m$ from $\mathbb{Z}^r$ to $\mathbb{Z}^{r_1}, \ldots, \mathbb{Z}^{r_m}$, respectively. Let $s_1, \ldots, s_m$ be nonnegative real numbers.*

$$\textit{If} \qquad \operatorname{rank}(H) \le \sum_j s_j \operatorname{rank}(\phi_j(H)) \qquad \textit{for all subgroups } H \textit{ of } \mathbb{Z}^r, \tag{4.1}$$

$$\textit{then} \qquad |E| \le \prod_j |\phi_j(E)|^{s_j} \qquad \textit{for all nonempty finite subsets } E \textit{ of } \mathbb{Z}^r. \tag{4.2}$$

Proof. See Section 4.6.

Each $x \in \mathbb{Z}^r$ identifies an operation whose operands are identified by $\phi_1(x), \ldots, \phi_m(x)$; thus, Theorem 4.1 gives an upper bound on the number $|E|$ of operations that can be performed when certain numbers $|\phi_1(E)|, \ldots, |\phi_m(E)|$ of operands are available.

We will actually prove a more general version of Theorem 4.1, stated as Theorem 4.2 in Section 4.2. While Theorem 4.1 suffices to analyze our target class of computations, we believe there are applications for the more general result in Theorem 4.2, in particular regarding groups with torsion.

This chapter is largely based on material that appeared in our earlier report [15, Sections 3 and 5]. A section-by-section outline of this chapter is as follows.

- In Section 4.1, we define *Hölder-Brascamp-Lieb-type (HBL-type) inequalities*, which generalize the inequalities (4.2) in the conclusion of Theorem 4.1 that we use to bound the number of doable operations given a bounded number of operands.

- In Section 4.2, we state our main results: Theorem 4.2 is a generalization of Theorem 4.1, and Theorems 4.3 and 4.4 concern the decidability of the hypotheses (4.1) of Theorem 4.1.
- In Section 4.3, we review related work on HBL-type inequalities, in particular, the paper by Bennett-Carbery-Christ-Tao [9] that inspired this line of inquiry, as well as our previous report [15].
- In Section 4.4, we study the hypothesis of Theorem 4.2, a generalization of (4.1).
- In Section 4.5, we study the conclusion of Theorem 4.2, a generalization of (4.2).
- In Section 4.6, we synthesize the results developed in Sections 4.4 and 4.5 to conclude the proof of Theorem 4.2, and then finally of Theorem 4.1, as a special case.
- Lastly, in Section 4.7, we study the decidability of the hypothesis of Theorem 4.2, proving Theorems 4.3 and 4.4.

4.1 Definitions and Notation

We define Hölder-Brascamp-Lieb-type inequalities using some measure-theoretic notation (definitions can be found in standard textbooks, e.g., [30]). Let (X, Σ, μ) be a measure space and let $f\colon X \to \mathbb{C} \cup \{\infty\}$ be a Σ-measurable function. For each $p \in (0, \infty]$, we define

$$\|f\|_{L^p(X,\Sigma,\mu)} = \begin{cases} \left(\int\limits_{(X,\Sigma)} |f|^p \, \mathrm{d}\mu \right)^{1/p} & p < \infty \\ \operatorname*{ess\,sup}\limits_{(X,\Sigma,\mu)} |f| & p = \infty. \end{cases}$$

Our work considers the special case of discrete measure spaces, where $\Sigma = 2^X$ is the discrete σ-algebra and $\mu = |\cdot|$ is counting measure; we specialize the preceding notation for this discrete setting, defining

$$\|f\|_{\ell^p(X)} = \|f\|_{L^p(X,2^X,|\cdot|)} = \begin{cases} \left(\sum\limits_{x \in X} |f(x)|^p \right)^{1/p} & p < \infty \\ \sup\limits_{x \in X} |f(x)| & p = \infty. \end{cases}$$

Additionally, it will be more convenient for us to work with the reciprocals $s = 1/p \in [0, \infty)$ of the exponents $p \in (0, \infty]$; in case $p = \infty$, we define $L^{1/0} = L^\infty$ and $\ell^{1/0} = \ell^\infty$.

A Hölder-Brascamp-Lieb-type inequality is defined by the following parameters:

- a positive integer $m \in \{1, 2, \ldots\}$, and dedicating j to range over $\{1, \ldots, m\}$,
- $m + 1$ measure spaces $(G, \Sigma, \mu), (G_j, \Sigma_j, \mu_j)$ where G, G_j are Abelian groups,
- m Abelian group homomorphisms $\phi_j\colon G \to G_j$,

- m Σ_j-measurable functions $f_j \colon G_j \to [0,\infty]$,
- m nonnegative real numbers $s_j \in [0,\infty)$, and
- an extended-real number $C \in \mathbb{R} \cup \{\pm\infty\}$.

Together these parameters define an inequality,

$$\int_{(G,\Sigma)} \prod_j f_j \circ \phi_j \, \mathrm{d}\mu \le C \prod_j \|f_j\|_{L^{1/s_j}(G_j,\Sigma_j,\mu_j)}, \tag{4.3}$$

where we adopt the convention that $0 \cdot \pm\infty = 0$.

We will study families of Hölder-Brascamp-Lieb-type inequalities, where some parameters are fixed, while others are treated as variables. In particular, we will frequently fix the number m, the measure spaces $(G,\Sigma,\mu), (G_j,\Sigma_j,\mu_j)$, and the homomorphisms ϕ_j, while varying the functions f_j, numbers s_j, and number C.

Since this work targets discrete measure spaces, our notation suppresses the discrete σ-algebras Σ, Σ_j and counting measures μ, μ_j to simplify in three ways: (1) we collect the fixed parameters as the triple $\mathcal{G} = (G, (G_j)_j, (\phi_j)_j)$ called an HBL datum, where m is implicit, (2) we use the shorthand notation $\|\cdot\|_{\ell^p(X)} = \|\cdot\|_{L^p(X,2^X,|\cdot|)}$ defined above, and (3) we avoid mentioning the (trivial) Σ_j-measurability of the functions f_j. We also collect the variable parameters f_j, s_j as m-tuples called m-functions $f = (f_j)_j \in \bigtimes_j (G_j \to [0,\infty])$ and m-exponents $s = (s_j)_j \in [0,\infty)^m$.

4.2 Main Results

Consider an HBL datum $\mathcal{G} = (G, (G_j)_j, (\phi_j)_j)$. Let $\mathcal{P}(\mathcal{G})$ denote the (convex) set of all m-exponents $s \in [0,\infty)^m$, such that

$$\operatorname{rank}(H) \le \sum_j s_j \operatorname{rank}(\phi_j(H)) \qquad \text{for all finite rank subgroups } H \text{ of } G; \tag{4.4}$$

additionally, for each $s \in [0,\infty)^m$, let $A(\mathcal{G},s), B(\mathcal{G},s), C(\mathcal{G},s) \in \mathbb{R} \cup \{\pm\infty\}$ denote the infima of $A, B, C \in \mathbb{R} \cup \{\pm\infty\}$ such that

$$|H| \le A \prod_j |\phi_j(H)|^{s_j} \qquad \text{for all finite subgroups } H \text{ of } G, \tag{4.5}$$

$$|E| \le B \prod_j |\phi_j(E)|^{s_j} \qquad \text{for all nonempty finite subsets } E \text{ of } G, \tag{4.6}$$

$$\sum_{x \in G} \prod_j f_j(\phi_j(x)) \le C \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} \qquad \begin{array}{c}\text{for all } m\text{-functions} \\ f \in \bigtimes_j (G_j \to [0,\infty]);\end{array} \tag{4.7}$$

We will prove Theorem 4.1 as a special case of the following more general result, which relates the four objects $\mathcal{P}(\mathcal{G}), A(\mathcal{G},\cdot), B(\mathcal{G},\cdot), C(\mathcal{G},\cdot)$ defined by $\mathcal{G}$.

Theorem 4.2 ([14, Theorem 1.2])**.** *Consider any $s \in [0,\infty)^m$ and let $r = (\min(1,s_j))_j$. If $s \in \mathcal{P}(\mathcal{G})$, then $B(\mathcal{G},s) = C(\mathcal{G},s) = A(\mathcal{G},r)$. Conversely, if $B(\mathcal{G},s) < \infty$, then $s \in \mathcal{P}(\mathcal{G})$.*

Proof. See Section 4.6.

A supplemental argument (see Lemmas 4.27 and 4.39) shows that there is no loss of generality to apply Theorem 4.2 only to m-exponents $s \in [0,1]^m$, enabling a simpler formulation (see [14, Theorem 1.2]). While we ignore this simplification here, we will exploit it at many points in our analysis.

A practical question is whether the hypothesis $s \in \mathcal{P}(\mathcal{G})$ of Theorem 4.2 can be verified computationally for any m-exponent s. While the general case is open, we found an affirmative answer in a special case, when G has finite rank: in this case, since ranks are natural numbers, and since G's rank is an upper bound on all its subgroups' ranks, only finitely many distinct inequalities can appear in (4.4). Our algorithm that proves the following result depends crucially on this fact.

Theorem 4.3 ([15, Theorem 5.1])**.** *If* $\mathrm{rank}(G) < \infty$*, then there exists an algorithm that computes a finite subset of the inequalities in* (4.4) *which define the same set* $\mathcal{P}(\mathcal{G})$*.*

Proof. See Section 4.7; there are additional assumptions on how $\mathcal{G}$ is represented.

While this result suffices to verify whether a given m-exponent $s \in \mathcal{P}(\mathcal{G})$, it does not address the computability of the set of inequalities in (4.4). That is, one can in principle query the existence of a particular inequality in (4.4) by asking whether a subgroup H of G exists with certain values of $\mathrm{rank}(H), \mathrm{rank}(\phi_j(H))$.

Theorem 4.4 ([15, Theorem 5.9])**.** *If* $\mathrm{rank}(G) < \infty$*, then there exists an algorithm to compute the set of all inequalities in* (4.4) *if and only if there exists an algorithm to determine whether an arbitrary system of polynomial equations over* $\mathbb{Q}$ *is solvable over* $\mathbb{Q}$*.*

Proof. See Section 4.7.1; there are additional assumptions on how $\mathcal{G}$ is represented.

The question of whether the latter algorithm exists is known as Hilbert's Tenth Problem over $\mathbb{Q}$, and this question remains open (see, e.g., [28]).

4.3 Background and Related Work

Hölder-Brascamp-Lieb-type inequalities include those studied by Rogers-Hölder [29, 20] (see also [26]), Young [36] (see also [7, 12]), Loomis-Whitney [24], and Brascamp-Lieb [12]. Typically, studies of Hölder-Brascamp-Lieb-type inequalities have considered the continuum setting, where the measure spaces $(G, \Sigma, \mu), (G_j, \Sigma_j, \mu_j)$ are finite-dimensional $\mathbb{R}$-vector spaces with Lebesgue measure and the homomorphisms ϕ_j are $\mathbb{R}$-linear maps (see [6] for references). A pair of papers [10, 9] by Bennett-Carbery-Christ-Tao derived necessary and sufficient conditions for Hölder-Brascamp-Lieb-type inequalities to hold with $C < \infty$, both in the continuum setting and, in [9], in the case of discrete Abelian groups. In particular, [9, Theorem 2.4] is the starting point for Theorem 4.2; we now restate this result in the present notation, assuming a given HBL datum $\mathcal{G}$.

Proposition 4.1 ([9, Theorem 2.4])**.** *Suppose* G, G_j *are finitely generated and consider any* $s \in [0, \infty)$*. If* $s \in \mathcal{P}(\mathcal{G})$*, then* $B(\mathcal{G}, s) \leq C(\mathcal{G}, s) < \infty$*. Conversely, if* $B(\mathcal{G}, s) < \infty$*, then* $s \in \mathcal{P}(\mathcal{G})$*.*

In their proof, Bennett-Carbery-Christ-Tao observed a link between the values $B(\mathcal{G}, s)$ and $C(\mathcal{G}, s)$ and the structure of the torsion subgroup $T(G)$ of G. We pursued this link in a paper [15], strengthening Proposition 4.1 with sharper bounds on these constants.

Proposition 4.2 ([15, Theorem 3.12])**.** *Suppose G, G_j are finitely generated and consider any $s \in [0, \infty)$. If $s \in \mathcal{P}(\mathcal{G})$, then $B(\mathcal{G}, s) \leq C(\mathcal{G}, s) \leq |T(G)|$. Conversely, if $B(\mathcal{G}, s) < \infty$, then $s \in \mathcal{P}(\mathcal{G})$.*

The inequality in (4.6) for $E = T(G) \cap \bigcap_j \ker(\phi_j)$ shows that $B(\mathcal{G}, s) \geq |E| \geq 1$ for all m-exponents s. Therefore, $B(\mathcal{G}, s) = C(\mathcal{G}, s) = |T(G)|$ whenever $T(G) \leq \bigcap_j \ker(\phi_j)$, e.g., when G is torsion-free, or when all G_j are torsion-free. In the general case, however, Proposition 4.2 leaves a gap of up to a factor of $|T(G)|$ between $B(\mathcal{G}, s), C(\mathcal{G}, s)$ and their upper bounds. This gap was closed by Christ [14], who precisely determined $B(\mathcal{G}, s), C(\mathcal{G}, s)$ in the more general setting where G, G_j are not necessarily finitely generated. Christ's main result [14, Theorem 1.2], stated above as Theorem 4.2, thus completed the study of the optimal constants for discrete Abelian groups.

In [15], we also studied the computability of the hypothesis $s \in \mathcal{P}(\mathcal{G})$; we already summarized our findings in Theorems 4.3 and 4.4. We mention here that our algorithm for Theorem 4.3 is similar to an algorithm of Valdimarsson [35, Theorem 1.8], in the continuum setting, for computing the hypothesis of [9, Theorem 2.1].

4.4 Analysis of $\mathcal{P}$

Now we study the set $\mathcal{P}(\mathcal{G})$, defined by an HBL datum $\mathcal{G}$ via (4.4), in more depth. We let $\mathcal{H}^*(G), \mathcal{H}^*_{\mathbb{N}}(G), \mathcal{H}^*_{\mathrm{fg}}(G), \mathcal{H}^*_0(G), \mathcal{H}^*_{\mathrm{f}}(G)$ denote the sets of all, all finite-rank, all finitely generated, all rank-zero, and all finite subgroups of G, respectively (definitions can be found in standard texts, e.g., [23]); these sets are partially ordered by containment,

$$\mathcal{H}^*(G) \supseteq \mathcal{H}^*_{\mathbb{N}}(G) \supseteq \mathcal{H}^*_{\mathrm{fg}}(G) \cup \mathcal{H}^*_0(G) \supseteq \mathcal{H}^*_{\mathrm{fg}}(G) \cap \mathcal{H}^*_0(G) = \mathcal{H}^*_{\mathrm{f}}(G).$$

Most results in this section are stated assuming a general HBL datum $\mathcal{G}$, although most will only be applied in the case where G is finitely generated, i.e., where

$$\mathcal{H}^*(G) = \mathcal{H}^*_{\mathbb{N}}(G) = \mathcal{H}^*_{\mathrm{fg}}(G) \supseteq \mathcal{H}^*_0(G) = \mathcal{H}^*_{\mathrm{f}}(G).$$

Given $\mathcal{G}$ and any $\mathcal{H} \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$, we define

$$\mathcal{P}(\mathcal{G}, \mathcal{H}) = \Big\{ s \in [0, \infty)^m \;\Big|\; (\forall H \in \mathcal{H}) \operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H)) \Big\}. \tag{4.8}$$

That is, $\mathcal{P}(\mathcal{G}, \mathcal{H})$ is defined by a nonempty, countable set of $\mathbb{Z}$-linear inequalities constraining $s \in \mathbb{R}^m$: there are m inequalities of the form $s_j \geq 0$ and countably many of the form $\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H))$, even if $\mathcal{H}$ is uncountable, since finite ranks are integers. Since $\mathcal{P}(\mathcal{G}, \mathcal{H})$ equals the intersection of closed half-spaces, it is closed and convex.

4.4.1 Group-Theoretic Tools

For the following three elementary results, let G be any Abelian group with finite rank, not necessarily associated with an HBL datum.

Lemma 4.1. *For any $H \leq G$, there exists $K \leq G$ such that $H \leq K$, $\operatorname{rank}(H) = \operatorname{rank}(K)$, and G/K is torsion-free.*

Proof. Define $K = \{x \mid (\exists a \in \mathbb{Z})ax \in H\}$; it follows that $H \leq K \leq G$. If $x + K \in G/K$ such that $a(x + K) = 0 + K$ for some $a \in \mathbb{Z}$, then $y = ax \in K$; however, by definition of K, there exists $b \in \mathbb{Z}$ such that $by \in H$, i.e., $bax \in H$, thus $x \in K$, so $x + K = 0 + K$, i.e., G/K is torsion-free. Since each element of K/H is a torsion element, K/H is a torsion group, so $\operatorname{rank}(K/H) = 0$, i.e., $\operatorname{rank}(H) = \operatorname{rank}(K)$. □

Lemma 4.2. *For any $H \leq G$, there exists $K \leq G$ such that $\operatorname{rank}(G) = \operatorname{rank}(H) + \operatorname{rank}(K)$ and $H \cap K = \{0\}$.*

Proof. By Lemma 4.1, we can find $H \leq F \leq G$ such that G/F is torsion-free and $\operatorname{rank}(H) = \operatorname{rank}(F)$. Since G/F has finite rank, let U be the subgroup of G/F generated by any $\operatorname{rank}(G/F)$ independent elements of G/F: U is finitely generated with $\operatorname{rank}(U) = \operatorname{rank}(G/F)$, and we can write $U = L/F$ for some $F \leq L \leq G$ such that $\operatorname{rank}(L) = \operatorname{rank}(G)$. Since L/F is a torsion-free finitely generated Abelian group, it is projective (moreover, free) as a $\mathbb{Z}$-module, meaning that for any surjective Abelian group homomorphism ψ with domain V and codomain L/F, there exists $W \leq V$ such that W and $\ker(\psi)$ are complementary subgroups of V. Letting $V = L$, we have that $\ker(\psi) = F$, and so taking $K = W \leq L$, we have that $L = F + K$, so $\operatorname{rank}(G) = \operatorname{rank}(L) = \operatorname{rank}(F) + \operatorname{rank}(K) = \operatorname{rank}(H) + \operatorname{rank}(K)$; moreover, $H \cap K \leq F \cap K = \{0\}$, i.e., $H \cap K = \{0\}$. □

Lemma 4.3. *If ϕ is an Abelian group homomorphism with domain G and if there exist $H, K \leq G$ such that $H \leq K \leq G$ and $\operatorname{rank}(H) = \operatorname{rank}(K)$, then $\operatorname{rank}(\phi(H)) = \operatorname{rank}(\phi(K))$.*

Proof. $0 = \operatorname{rank}(K/H) \geq \operatorname{rank}(\phi(K)/\phi(H)) \geq 0$, so $0 = \operatorname{rank}(\phi(K)) - \operatorname{rank}(\phi(H))$. □

4.4.2 Embedding $\mathbb{Z}$ in $\mathbb{Q}$

We digress briefly to mention an arguably simpler approach to studying $\mathcal{P}(\mathcal{G}, \cdot)$ which we will exploit later to prove Theorem 4.4.

Consider any HBL datum $\mathcal{G} = (G, (G_j)_j, (\phi_j)_j)$, but treat G, G_j as $\mathbb{Z}$-modules (defining multiplication in terms of addition) and ϕ_j as $\mathbb{Z}$-module homomorphisms. Let f be a ring homomorphism from $\mathbb{Z}$ to an integral domain $\mathbb{D}$. Define $G^{\mathbb{D}}, G_j^{\mathbb{D}}, \phi_j^{\mathbb{D}}$ via extension of scalars: that is, regarding $\mathbb{D}$ as a $\mathbb{Z}$-module via f, $G^{\mathbb{D}} = G \otimes_{\mathbb{Z}} \mathbb{D}$ and $G_j^{\mathbb{D}} = G_j \otimes_{\mathbb{Z}} \mathbb{D}$ are $\mathbb{D}$-modules and $\phi_j^{\mathbb{D}} = \phi_j \otimes_{\mathbb{Z}} \mathrm{id}_{\mathbb{D}}$ are $\mathbb{D}$-module homomorphisms ($\otimes_{\mathbb{Z}}$ denotes the tensor product of $\mathbb{Z}$-modules or $\mathbb{Z}$-module homomorphisms). We also introduce the notation $U \leq_{\mathbb{D}} V$, asserting that U is a sub(-$\mathbb{D}$-)module of a $\mathbb{D}$-module V, and $\operatorname{rank}_{\mathbb{D}}(V)$, denoting the rank (over $\mathbb{D}$) of a $\mathbb{D}$-module V, but often omit the subscript-$\mathbb{D}$s in the case $\mathbb{D} = \mathbb{Z}$ since ambiguity in this case causes no harm.

Lemma 4.4. *Consider any $h, h_j \in \mathbb{N}$. If there exists $H \leq G$ such that $\operatorname{rank}(H) = h$ and each $\operatorname{rank}(\phi_j(H)) = h_j$, then there exists $V \leq_{\mathbb{D}} G^{\mathbb{D}}$ such that $\operatorname{rank}_{\mathbb{D}}(V) = h$ and each $\operatorname{rank}_{\mathbb{D}}(\phi_j^{\mathbb{D}}(V)) = h_j$. The converse also holds if $\mathbb{D} = \mathbb{Q}$.*

Proof. Letting $V = H^{\mathbb{D}} = H \otimes_{\mathbb{Z}} \mathbb{D} \leq_{\mathbb{D}} G^{\mathbb{D}}$, the rank properties follow as consequences of the $\mathbb{Z}$-linearity of $\otimes_{\mathbb{Z}}$, i.e., if elements $e_i \in G$ are $\mathbb{Z}$-linearly independent then elements $e_i \otimes 1 \in G^{\mathbb{D}}$ are $\mathbb{D}$-linearly independent. To show the converse, note that $\mathbb{Q}$ is the fraction field of $\mathbb{Z}$. Therefore, for any $V \leq_{\mathbb{Q}} G^{\mathbb{Q}}$, any $\mathbb{Q}$-basis of V (at least one exists) can be converted to a (same cardinality) $\mathbb{Z}$-basis of some $V_{\mathbb{Z}} \leq G$ such that $V = (V_{\mathbb{Z}})^{\mathbb{Q}}$. □

Therefore, it is possible to analyze $\mathcal{P}(\mathcal{G}, \cdot)$ via $\mathcal{P}_{\mathbb{Q}}(\mathcal{G}^{\mathbb{Q}}, \cdot)$, where $\mathcal{G}^{\mathbb{Q}} = (G^{\mathbb{Q}}, (G_j^{\mathbb{Q}})_j, (\phi_j^{\mathbb{Q}})_j)$ and $\mathcal{P}_{\mathbb{Q}}$ is obtained from $\mathcal{P}$ by replacing Abelian group ranks with $\mathbb{Q}$-module ranks in (4.8). The primary advantage of this approach is that $\mathbb{Q}$-modules are free, so some arguments simplify: in particular, submodules always have complements, avoiding the need for Lemmas 4.1 and 4.2, and proper submodules always have strictly smaller rank, avoiding the need for Lemma 4.3.

4.4.3 Properties of $\mathcal{P}(\mathcal{G}, \mathcal{H})$

We now derive some basic results regarding $\mathcal{P}(\mathcal{G}, \mathcal{H})$ for any fixed $\mathcal{G}$ and $\mathcal{H} \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$.

Lemma 4.5. *If $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$, then $t \in \mathcal{P}(\mathcal{G}, \mathcal{H})$ for all $s \leq t \in [0, \infty)^m$.*

Proof. If the inequality induced by any $H \in \mathcal{H}$ is satisfied by s then it is by t as well. □

Lemma 4.6. *If $\mathcal{H}' \subseteq \mathcal{H}$ then $\mathcal{P}(\mathcal{G}, \mathcal{H}') \supseteq \mathcal{P}(\mathcal{G}, \mathcal{H})$.*

Proof. $\mathcal{P}(\mathcal{G}, \mathcal{H}')$ is defined by a subset of the inequalities that define $\mathcal{P}(\mathcal{G}, \mathcal{H})$. □

There is no loss of generality to suppose each ϕ_j is surjective, i.e., $G_j = \phi_j(G)$.

Lemma 4.7. $\mathcal{P}(\mathcal{G}, \mathcal{H}) = \mathcal{P}((G, (\phi_j(G))_j, (\chi_j)_j), \mathcal{H})$, *where* $\chi_j \colon G \to \phi_j(G) \colon x \mapsto \phi_j(x)$.

Proof. For each $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ and each j, $\operatorname{rank}(\chi_j(H)) = \operatorname{rank}(\psi_j(H))$. □

Additionally, there is no loss of generality to ignore the rank-0 subgroups of G.

Lemma 4.8. $\mathcal{P}(\mathcal{G}, \mathcal{H}) = \mathcal{P}(\mathcal{G}, \mathcal{H} \setminus \mathcal{H}^*_0(G))$.

Proof. Each $H \in \mathcal{H}^*_0(G)$ induces the inequality $0 \leq \sum_j s_j \cdot 0$, satisfied by all $s \in \mathbb{R}^m$. □

Note that if $\mathcal{H} \subseteq \mathcal{H}^*_0(G)$, e.g., $\mathcal{H} = \emptyset$ or $G \in \mathcal{H}^*_0(G)$, then $\mathcal{P}(\mathcal{G}, \mathcal{H}) = [0, \infty)^m$.

Lemma 4.9. *If $r = \sup_{H \in \mathcal{H}} \operatorname{rank}(H) < \infty$, then $\mathcal{P}(\mathcal{G}, \mathcal{H}') = \mathcal{P}(\mathcal{G}, \mathcal{H})$ for some finite $\mathcal{H}' \subseteq \mathcal{H}$.*

Proof. The ranks of H and its homomorphic images $\phi_j(H)$ are integers between 0 and $r \in \mathbb{N}$; therefore, the set $\mathcal{H}$, possibly infinite, generates at most $(r+1)^{m+1} < \infty$ distinct inequalities. The conclusion follows by choosing any subset $\mathcal{H}'$ of $\mathcal{H}$ that induces these inequalities. □

Lemma 4.10. *If $\operatorname{rank}(H \cap \bigcap_j \ker(\phi_j)) = 0$ for all $H \in \mathcal{H}$, then $[1, \infty)^m \subseteq \mathcal{P}(\mathcal{G}, \mathcal{H})$.*

Proof. If $\operatorname{rank}(G) = 0$, then $\mathcal{P}(\mathcal{G}, \mathcal{H}) = [0, \infty)^m$ by Lemma 4.8, so the conclusion is immediate. If $\operatorname{rank}(G) > 0$, then consider any $H \in \mathcal{H}$ and the corresponding homomorphism $\Phi\colon H \to \bigoplus_j \phi_j(H)\colon x \mapsto (\phi_j(x))_j$; let $K = \ker(\Phi) = H \cap \bigcap_j \ker(\phi_j) \in \mathcal{H}_0^*(G)$, so by the first isomorphism theorem ($\Phi(H) \cong H/K$), the fact that $\operatorname{rank}(\Phi(H)) \leq \operatorname{rank}(\bigoplus_j \phi_j(H)) = \sum_j \operatorname{rank}(\phi_j(H))$, and the hypothesis,

$$0 = \operatorname{rank}(K) = \operatorname{rank}(H) - \operatorname{rank}(\Phi(H)) \geq \operatorname{rank}(H) - \sum_j \operatorname{rank}(\phi_j(H)),$$

so for all $H \in \mathcal{H}$, $\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H))$ holds with $s_j = 1$ for each j. Thus by Lemma 4.5, $t \in \mathcal{P}(\mathcal{G}, \mathcal{H})$ for all $t \geq (1, \ldots, 1)$. □

Lemma 4.11. *If* $\operatorname{rank}(H \cap \bigcap_j \ker(\phi_j)) > 0$ *for some* $H \in \mathcal{H}$*, then* $\mathcal{P}(\mathcal{G}, \mathcal{H}) = \emptyset$*.*

Proof. Observe that $\sum_j s_j \operatorname{rank}(\phi_j(H)) = 0 < \operatorname{rank}(H)$ for all $s \in \mathbb{R}^m$. □

Lemma 4.12. *If* $H \in \mathcal{H} \setminus \mathcal{H}_0^*(G)$*, then* $\sum_j s_j \geq 1$ *for all* $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$*.*

Proof. Rearranging H's induced inequality, $1 \leq \sum_j s_j \operatorname{rank}(\phi_j(H))/\operatorname{rank}(H) \leq \sum_j s_j$. □

Lemma 4.13. *Suppose* $m \geq 2$ *and consider any* $s \in [0, \infty)^m$ *with some component* $s_k = 0$*. Then,* $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$ *if and only if* $(s_j)_{j \neq k} \in \mathcal{P}((G, (G_j)_{j \neq k}(\phi_j)_{j \neq k}), \mathcal{H})$*.*

Proof. For any $H \in \mathcal{H}_{\mathbb{N}}^*(G)$, $\sum_j s_j \operatorname{rank}(\phi_j(H)) = \sum_{j \neq k} s_j \operatorname{rank}(\phi_j(H))$. □

Lemma 4.14. *Suppose that there exist* $k, l \in \{1, \ldots, m\}$ *such that* $k < l$ *and* $\operatorname{rank}(\ker(\phi_k) \cap \ker(\phi_l)) = \operatorname{rank}(\ker(\phi_k)) = \operatorname{rank}(\ker(\phi_l)) < \infty$*. Let* $\mathcal{G}' = (G, (G_j)_{j \neq l}, (\phi_j)_{j \neq l})$*. If* $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$*, then* $t \in \mathcal{P}(\mathcal{G}', \mathcal{H})$ *where*

$$t = (s_1, \ldots, s_{k-1}, s_k + s_l, s_{k+1}, \ldots, s_{l-1}, s_{l+1}, \ldots, s_m);$$

conversely, if $t \in \mathcal{P}(\mathcal{G}')$*, then for all* $\epsilon \in [0, t_k]$*,* $s(\epsilon) \in \mathcal{P}(\mathcal{G})$ *where*

$$s(\epsilon) = (t_1, \ldots, t_{k-1}, \epsilon, t_{k+1}, \ldots, t_{l-1}, t_k - \epsilon, t_l, \ldots, t_{m-1}).$$

Proof. The inequality induced by any $H \leq G$ can be rewritten,

$$\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H)) = (s_k + s_l) \operatorname{rank}(\phi_k(H)) + \sum_{j \neq k,l} s_j \operatorname{rank}(\phi_j(H)).$$

□

Lemma 4.15. *Suppose that there exists* $k \in \{1, \ldots, m\}$ *such that* $\operatorname{rank}(\phi_k(G)) = 0$*. Let* $\mathcal{G}' = (G, (G_j)_{j \neq k}, (\phi_j)_{j \neq k})$*. If* $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$*, then* $t = (s_j)_{j \neq k} \in \mathcal{P}(\mathcal{G}', \mathcal{H})$*; conversely, if* $t \in \mathcal{P}(\mathcal{G}', \mathcal{H})$*, then for all* $\epsilon \in [0, \infty)$*,* $s(\epsilon) \in \mathcal{P}(\mathcal{G}, \mathcal{H})$ *where*

$$s(\epsilon) = (t_1, \ldots, t_{k-1}, \epsilon, t_k, \ldots, t_{m-1}).$$

Proof. The inequality induced by any $H \leq G$ can be rewritten,

$$\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H)) = \sum_{j \neq k} s_j \operatorname{rank}(\phi_j(H)).$$

□

Lemma 4.16. *If there exists a $k \in \{1, \dots, m\}$ such that $\operatorname{rank}(\phi_k(G)) = \operatorname{rank}(G) < \infty$ and if there exists an $H \in \mathcal{H}$ such that $\operatorname{rank}(H) > 0$, then $e_k \in \mathcal{P}(\mathcal{G}, \mathcal{H})$.*

Proof. For any $K \leq G$, $\operatorname{rank}(\phi_k(K)) = \operatorname{rank}(K)$, so the the inequality induced by K can be rewritten,

$$\operatorname{rank}(K) \leq \sum_j s_j \operatorname{rank}(\phi_j(K)) = s_k \operatorname{rank}(K) + \sum_{j \neq k} s_j \operatorname{rank}(\phi_j(K)).$$

If $s_k \geq 1$, the preceding inequality holds for any $(s_j)_{j \neq k} \in [0, \infty)^{m-1}$. □

4.4.4 Extreme Points of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$

We continue studying arbitrary $\mathcal{G}$ and $\mathcal{H} \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$, which remain fixed across the following results. As mentioned earlier, $\mathcal{P}(\mathcal{G}, \mathcal{H})$ is a closed convex subset of $\mathbb{R}^m$. The following results consider intersecting $\mathcal{P}(\mathcal{G}, \mathcal{H})$ with the unit m-cube $[0, 1]^m$. We will motivate the choice $[0, 1]^m$ later in Lemma 4.27; for now, we only exploit the fact that this intersection results in a bounded set. Therefore, as a bounded and closed convex subset of $\mathbb{R}^m$, $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$ equals the convex hull of its extreme points (in particular, such points exist).

Lemma 4.17. *Suppose $m \geq 2$ and consider any extreme point s of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$. Either some component $s_k \in \{0, 1\}$ or there exists $H \in \mathcal{H}$ such that $0 < \operatorname{rank}(H) < \operatorname{rank}(G)$ and $\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H))$.*

Proof. If $\operatorname{rank}(G) = 0$ then by Lemma 4.8, $\mathcal{P}(\mathcal{G}, \mathcal{H}) = [0, \infty)^m$, so the first case applies. Otherwise, suppose $\operatorname{rank}(G) > 0$ and $s \in (0, 1)^m$. As an extreme point of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$, s must lie at the intersection of m $\mathbb{Z}$-linearly independent hyperplanes that support $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$, and since $s \in (0, 1)^m$, these hyperplanes support $\mathcal{P}(\mathcal{G}, \mathcal{H})$ but not $[0, 1]^m$. In other words, we can find $H_1, H_2 \in \mathcal{H}$ such that $0 < \operatorname{rank}(H_i) = \sum_j s_j \operatorname{rank}(\phi_j(H_i))$ for both $i \in \{1, 2\}$. Moreover, the $\mathbb{Z}$-linear independence of the corresponding hyperplanes means that for some j, $\operatorname{rank}(\phi_j(H_1)) \neq \operatorname{rank}(\phi_j(H_2))$. Therefore, the ranks of H_1 and H_2 cannot both equal $\operatorname{rank}(G)$: if $\operatorname{rank}(G) < \infty$, this follows from Lemma 4.3, and if $\operatorname{rank}(G) = \infty$, this follows since $H_1, H_2 \in \mathcal{H}^*_{\mathbb{N}}(G)$. Therefore there exists $H_i \in \mathcal{H}$ such that $0 < \operatorname{rank}(H_i) = \sum_j s_j \operatorname{rank}(\phi_j(H_i)) < \operatorname{rank}(G)$. □

Lemma 4.18. *If $\sum_j \ker(\phi_j) \in \mathcal{H}^*_0(G)$ and s is an extreme point of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$, then $s \in \{0, 1\}^m$.*

Proof. By the first hypothesis, $\operatorname{rank}(H) = \operatorname{rank}(\phi_j(H))$ for all $H \in \mathcal{H}$, and every positive rank subgroup induces the same inequality $1 \leq \sum_j s_j$. This inequality corresponds to a hyperplane in $\mathbb{R}^m$ with normal $(1, \dots, 1)$, which intersects $[0, 1]^m$ at $e_1, \dots, e_m$ (the standard basis vectors in $\mathbb{R}^m$), thus all extreme points in $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$ are in $\{0, 1\}^m$. □

Lemma 4.19. *If s is an extreme point of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$ with some component $s_k \in (0, 1)$, then $\operatorname{rank}(\phi_k(H)) > 0$ for all $H \in \mathcal{H}$ with $\operatorname{rank}(H) = \operatorname{rank}(G)$.*

Proof. Suppose towards obtaining a contradiction that the hypotheses hold but $\operatorname{rank}(\phi_k(H)) = 0$ for some $H \in \mathcal{H}$ with $\operatorname{rank}(H) = \operatorname{rank}(G)$. By Lemma 4.3 and the definition of rank,

$\operatorname{rank}(\phi_k(K)) = 0$ for all $K \in \mathcal{H}$; therefore, $\sum_j t_j \operatorname{rank}(\phi_j(K)) = t_k \cdot 0 + \sum_{j \neq k} t_j \operatorname{rank}(\phi_j(K))$ for all $K \in \mathcal{H}$ and all $t \in \mathcal{P}(\mathcal{G}, \mathcal{H})$. Thus, both

$$t_- = (s_1, \ldots, s_{k-1}, 0, s_{k+1}, \ldots, s_m), \qquad t_+ = (s_1, \ldots, s_{k-1}, 1, s_{k+1}, \ldots, s_m)$$

are in $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$ and $s \neq t_\pm$ by hypothesis. Since s lies on an open line segment between $t_\pm \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$, it cannot be an extreme point, a contradiction. □

Lemma 4.20. *If s is an extreme point of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$, $\sup_{H \in \mathcal{H}} \operatorname{rank}(H) < \infty$, and $\operatorname{rank}(H) < \sum_j s_j \operatorname{rank}(\phi_j(H))$ for all $H \in \mathcal{H} \setminus \mathcal{H}_0^*(G)$, then $s \in \{0,1\}^m$.*

Proof. Suppose towards obtaining a contradiction that the hypotheses hold but some component $s_k \in (0,1)$. First, observe that

$$t_+ = (s_1, \ldots, s_{k-1}, 1, s_{k+1}, \ldots, s_m) \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m,$$

via Lemma 4.5, and that $s \neq t_+$. Second, under the present hypotheses that $s_k > 0$ and the inequality generated by each $H \in \mathcal{H}$ with $\operatorname{rank}(H) > 0$ is strict, it follows that for each such H we can find a positive ϵ such that $\operatorname{rank}(H) \leq \sum_j (s_j - \epsilon) \operatorname{rank}(\phi_j(H))$. From Lemma 4.9, it suffices to restrict our attention to the inequalities induced by some finite $\mathcal{H}' \subseteq \mathcal{H}$, in which case only finitely many ϵ arise and their minimum ϵ^* is well-defined, and, in particular, positive. Thus we can define

$$t_- = (s_1, \ldots, s_{k-1}, s_k - \epsilon^*, s_{k+1}, \ldots, s_m) \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m,$$

and $s \neq t_-$. Since s lies on an open line segment between $t_\pm \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$, it cannot be an extreme point, a contradiction. □

Lemma 4.21. *If s is an extreme point of $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$, $\sup_{H \in \mathcal{H}} \operatorname{rank}(H) < \infty$, and $\operatorname{rank}(H) < \sum_j s_j \operatorname{rank}(\phi_j(H))$ for all $H \in \mathcal{H} \setminus \mathcal{H}_0^*(G)$ such that $\operatorname{rank}(H) < \operatorname{rank}(G)$, then at most one component $s_k \notin \{0,1\}$.*

Proof. Suppose towards obtaining a contradiction that the hypotheses hold but $s_k, s_l \in (0,1)$ for two distinct indices k, l. By Lemma 4.19, $\operatorname{rank}(\phi_k(G)), \operatorname{rank}(\phi_l(G)) > 0$; in particular, $\operatorname{rank}(G) > 0$. Define $t(\epsilon) \in \mathbb{R}^m$ componentwise by

$$t_j(\epsilon) = \begin{cases} s_k + \epsilon \operatorname{rank}(\phi_l(G)) & j = k \\ s_l - \epsilon \operatorname{rank}(\phi_k(G)) & j = l \\ s_j & j \notin \{k, l\}. \end{cases}$$

We now determine constraints on ϵ so that $t(\epsilon) \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$. For any $H \in \mathcal{H}$, we will require that

$$\begin{aligned} \operatorname{rank}(H) &\leq \sum_j t_j(\epsilon) \operatorname{rank}(\phi_j(H)) \\ &= \epsilon(\operatorname{rank}(\phi_l(G)) \operatorname{rank}(\phi_k(H)) - \operatorname{rank}(\phi_k(G)) \operatorname{rank}(\phi_l(H))) + \sum_j s_j \operatorname{rank}(\phi_j(H)); \end{aligned}$$

defining

$$d(H) = \sum_j s_j \operatorname{rank}(\phi_j(H)) - \operatorname{rank}(H) \in [0, \infty),$$
$$r(H) = \operatorname{rank}(\phi_l(G)) \operatorname{rank}(\phi_k(H)) - \operatorname{rank}(\phi_k(G)) \operatorname{rank}(\phi_l(H)) \in \{-\operatorname{rank}(G)^2, \dots, \operatorname{rank}(G)^2\},$$

the constraint on ϵ that each $H \in \mathcal{H}$ induces is $-\epsilon r(H) \leq d(H)$, or more strictly, $|\epsilon||r(H)| \leq d(H)$. Actually, there is no constraint on ϵ in the case $r(H) = 0$, e.g., when $\operatorname{rank}(H) = \operatorname{rank}(G)$ or $\operatorname{rank}(H) = 0$. Thus it suffices to consider only the subset $\mathcal{H}' \subseteq \mathcal{H}$ comprising the $H \in \mathcal{H}$ such that $0 < \operatorname{rank}(H) < \operatorname{rank}(G)$. By the same argument used to prove Lemma 4.9, $d(H)$ takes on only finitely many distinct values as H varies over $\mathcal{H}'$, thus we can define $d^* = \min_{H \in \mathcal{H}'} d(H)$. It remains to enforce that $t \in [0, \infty)^m$, and, in particular, that $t \in [0, 1]^m$: $0 \leq s_k + \epsilon \operatorname{rank}(\phi_l(G)) \leq 1$ and $0 \leq s_l - \epsilon \operatorname{rank}(\phi_k(G)) \leq 1$. Combining these constraints on ϵ, replacing $d(H)$ by its minimum d^*, replacing $|r(H)|$, $\operatorname{rank}(\phi_k(G))$, and $\operatorname{rank}(\phi_l(G))$ by a common upper bound $\operatorname{rank}(G)^2 > 0$, and symmetrizing (tightening at most one of) the upper and lower bounds,

$$|\epsilon| \leq \frac{\min\{d^*, s_k, 1 - s_k, s_l, 1 - s_l\}}{\operatorname{rank}(G)^2};$$

by hypothesis, $d^* > 0$ and $s_k, s_l \in (0, 1)$, so the right-hand side is positive. Thus, there exists $\epsilon_\pm \in \mathbb{R}$ such that $\epsilon_- < 0 < \epsilon_+$, $t(\epsilon_\pm) \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$, and $s \neq t(\epsilon_\pm)$. Since s lies on an open line segment between $t(\epsilon_\pm) \in \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0, 1]^m$, it cannot be an extreme point, a contradiction. □

4.4.5 Properties of $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathbb{N}}(G))$

For any HBL datum $\mathcal{G}$, the set $\mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathbb{N}}(G))$ is of principal interest, so we introduce the notation $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathbb{N}}(G))$.

Before studying $\mathcal{P}(\mathcal{G})$, we mention that we could also define $\mathcal{P}(\mathcal{G})$ in terms of $\mathcal{H}^*_{\text{fg}}(G)$.

Lemma 4.22. $\mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathbb{N}}(G)) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\text{fg}}(G))$.

Proof. We have right-to-left inclusion from the fact that $\mathcal{H}^*_{\text{fg}}(G) \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$. To observe left-to-right inclusion, for each $H \in \mathcal{H}^*_{\mathbb{N}}(G)$, pick any $\operatorname{rank}(G)$ independent elements of G to generate $K \in \mathcal{H}^*_{\text{fg}}(G)$ such that $\operatorname{rank}(H) = \operatorname{rank}(K)$; by Lemma 4.3, for each j, $\operatorname{rank}(\phi_j(H)) = \operatorname{rank}(\phi_j(K))$. □

Lemma 4.23. *If* $G \in \mathcal{H}^*_{\mathbb{N}}(G)$, $H \in \mathcal{H}^*_{\text{fg}}(G)$, *and* $\operatorname{rank}(H) = \operatorname{rank}(G)$, *then* $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\text{fg}}(H))$.

Proof. We have right-to-left inclusion from the fact that $\mathcal{H}^*_{\text{fg}}(H) \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$. (Note that all subgroups of H are finitely generated.) To observe left-to-right inclusion, consider each $K \in \mathcal{H}^*_{\mathbb{N}}(G)$: letting $F = K \cap H \in \mathcal{H}^*_{\text{fg}}(H)$, it follows from the argument in Lemma 4.3 that $\operatorname{rank}(K) = \operatorname{rank}(F)$ and for each j, $\operatorname{rank}(\phi_j(K)) = \operatorname{rank}(\phi_j(F))$. □

A key property of $\mathcal{P}(\mathcal{G})$ is that it factors in the following sense. Given the HBL datum $\mathcal{G} = (G, (G_j), (\phi_j))$ and any $H \leq G$, we define two associated HBL data called the factor data: the restricted datum $\mathcal{G}|_H$ and the quotient datum $\mathcal{G}|^H$:

$$\begin{aligned}
\mathcal{G}|_H &= (H, (\phi_j(H))_j, (\chi_j)_j), & \chi_j &\colon H \to \phi_j(H) \colon x \mapsto \phi_j(x), \\
\mathcal{G}|^H &= (G/H, (G_j/\phi_j(H))_j, (\psi_j)_j), & \psi_j &\colon G/H \to G_j/\phi_j(H) \colon x + H \mapsto \phi_j(x) + \phi_j(H).
\end{aligned}$$

In our analysis, the symbols χ_j and ψ_j will always denote the homomorphisms associated with the factor data $\mathcal{G}|_H$ and $\mathcal{G}|^H$, i.e., χ_j and ψ_j are implied by the notations $\mathcal{G}|_H$ and $\mathcal{G}|^H$. Also note an asymmetry in these definitions: χ_j is always surjective, whereas ψ_j is surjective only if ϕ_j is; this property is exploited later in Lemma 4.43; however, the homomorphisms' codomains play no role in the analysis of $\mathcal{P}$.

Lemma 4.24. *If $H \in \mathcal{H}^*_{\mathbb{N}}(G)$, then $\mathcal{P}(\mathcal{G}|_H) \cap \mathcal{P}(\mathcal{G}|^H) \subseteq \mathcal{P}(\mathcal{G}) \subseteq \mathcal{P}(\mathcal{G}|_H)$.*

Proof. Consider any $K \in \mathcal{H}^*_{\mathbb{N}}(G)$: it follows that $K \cap H \in \mathcal{H}^*_{\mathbb{N}}(H)$ and $(K + H)/H \in \mathcal{H}^*_{\mathbb{N}}(G/H)$, even if $\operatorname{rank}(G) = \infty$. Now consider any $s \in \mathcal{P}(\mathcal{G}|_H) \cap \mathcal{P}(\mathcal{G}|^H)$: it follows that $\operatorname{rank}(K \cap H) \leq \sum_j s_j \operatorname{rank}(\chi_j(K \cap H))$ and $\operatorname{rank}((K+H)/H) \leq \sum_j s_j \operatorname{rank}(\psi_j((K+H)/H))$. Combining these results,

$$\begin{aligned}
\operatorname{rank}(K) &= \operatorname{rank}(K \cap H) + \operatorname{rank}((K + H)/H) \\
&\leq \sum_j s_j \operatorname{rank}(\chi_j(K \cap H)) + \sum_j s_j \operatorname{rank}(\psi_j((K + H)/H)) \\
&= \sum_j s_j \Big(\operatorname{rank}(\chi_j(K \cap H)) + \operatorname{rank}(\psi_j((K + H)/H))\Big) \\
&= \sum_j s_j \Big(\operatorname{rank}(\phi_j(K \cap H)) + \operatorname{rank}((\phi_j(K) + \phi_j(H))/\phi_j(H))\Big) \\
&= \sum_j s_j \operatorname{rank}(\phi_j(K)).
\end{aligned}$$

Repeating for all $K \in \mathcal{H}^*_{\mathbb{N}}(G)$, we conclude that $s \in \mathcal{P}(\mathcal{G})$; the first inclusion follows by repeating this argument for every $s \in \mathcal{P}(\mathcal{G}|_H) \cap \mathcal{P}(\mathcal{G}|^H)$.

To show the second inclusion, consider any $s \in \mathcal{P}(\mathcal{G})$. By definition, $\mathcal{H}^*_{\mathbb{N}}(H) \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$. Since $\phi_j(K) = \chi_j(K)$ for any $K \in \mathcal{H}^*_{\mathbb{N}}(H)$, $s \in \mathcal{P}(\mathcal{G}|_H)$. Considering every $s \in \mathcal{P}(\mathcal{G})$ we conclude $\mathcal{P}(\mathcal{G}) \subseteq \mathcal{P}(\mathcal{G}|_H)$. □

Lemma 4.25. *If $s \in \mathcal{P}(\mathcal{G})$, $H \in \mathcal{H}^*_{\mathbb{N}}(G)$, and $\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H))$, then $s \in \mathcal{P}(\mathcal{G}|_H) \cap \mathcal{P}(\mathcal{G}|^H)$.*

Proof. By Lemma 4.24, $s \in \mathcal{P}(\mathcal{G}) \subseteq \mathcal{P}(\mathcal{G}|_H)$, so it remains to show that $s \in \mathcal{P}(\mathcal{G}|^H)$. Each element of $\mathcal{H}^*_{\mathbb{N}}(G/H)$ can be written as $(K+H)/H$ for some $K \in \mathcal{H}^*_{\mathbb{N}}(G)$, even if $\operatorname{rank}(G) = \infty$. Additionally, $\operatorname{rank}((K+H)/H) = \operatorname{rank}(K+H) - \operatorname{rank}(H)$ and, for each j, $\operatorname{rank}(\psi_j((K+H)/H)) = \operatorname{rank}((\phi_j(K)+\phi_j(H))/\phi_j(H)) = \operatorname{rank}(\phi_j(K)+\phi_j(H)) - \operatorname{rank}(\phi_j(H))$. Combining

these identities and the hypotheses that $s \in \mathcal{P}(\mathcal{G})$ and $\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H))$,

$$\begin{aligned}\operatorname{rank}((K+H)/H) &= \operatorname{rank}(K+H) - \operatorname{rank}(H) \\ &\leq \sum_j s_j \operatorname{rank}(\phi_j(K+H)) - \sum_j s_j \operatorname{rank}(\phi_j(H)) \\ &= \sum_j s_j \Big(\operatorname{rank}(\phi_j(K+H)) - \operatorname{rank}(\phi_j(H))\Big) \\ &= \sum_j s_j \operatorname{rank}(\psi((K+H)/H)).\end{aligned}$$

Repeating for all $(K+H)/H \in \mathcal{H}^*_{\mathbb{N}}(G/H)$, we conclude that $s \in \mathcal{P}(\mathcal{G}|^H)$. □

Lemma 4.26. *Consider any $s \in [0,\infty)^m$ with some component $s_k \geq 1$ and let $K = \ker(\phi_k)$. Then $s \in \mathcal{P}(\mathcal{G})$ if and only if $(s_j)_{j\neq k} \in \mathcal{P}((G,(G_j)_{j\neq k},(\phi_j|_K)_{j\neq k})$.*

Proof. Sufficiency follows from the fact that, for any $H \in \mathcal{H}^*_{\mathbb{N}}(K)$, $\sum_j s_j \operatorname{rank}(\phi_j(H)) = \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(H))$. To show necessity, consider any $H \in \mathcal{H}^*_{\mathbb{N}}(G)$: let $F_1 = H \cap K$ and apply Lemma 4.2 to define $F_2 \leq H$ such that $\operatorname{rank}(H) = \operatorname{rank}(F_1) + \operatorname{rank}(F_2)$. It follows that both $F_1, F_2 \in \mathcal{H}^*_{\mathbb{N}}(G)$, so

$$\begin{aligned}&\sum_j s_j \operatorname{rank}(\phi_j(H)) \\ &\quad\geq \operatorname{rank}(\phi_k(H)) + \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(H)) \geq \operatorname{rank}(\phi_k(F_2)) + \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(F_1)) \\ &\quad\geq \operatorname{rank}(\phi_k(F_2)) + \operatorname{rank}(F_1) = \operatorname{rank}(F_2) + \operatorname{rank}(F_1) = \operatorname{rank}(H).\end{aligned}$$

□

Lemma 4.27. *If $s \in \mathcal{P}(\mathcal{G})$, then $t \in \mathcal{P}(\mathcal{G})$ for all $(\min\{1,s_j\})_j \leq t \in [0,\infty)^m$.*

Proof. First suppose some $s_k > 1$ and define $t = (s_1,\ldots,s_{k-1},1,s_{k+1},\ldots,s_m)$; we now show that $t \in \mathcal{P}(\mathcal{G})$. Consider any $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ and let $F = H \cap \ker(\phi_k)$; since $F \in \mathcal{H}^*_{\mathbb{N}}(G)$,

$$\operatorname{rank}(F) \leq \sum_j s_j \operatorname{rank}(\phi_j(F)) = s_k \cdot 0 + \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(F)) = \sum_{j\neq k} t_j \operatorname{rank}(\phi_j(F)).$$

By Lemma 4.2, there exists $K \leq H$ such that $\operatorname{rank}(H) = \operatorname{rank}(F) + \operatorname{rank}(K)$ and $\operatorname{rank}(F \cap K) = 0$. Since $K \cap F = K \cap (H \cap \ker(\phi_k)) = K \cap \ker(\phi_k)$ and $K \in \mathcal{H}^*_{\mathbb{N}}(G)$,

$$\operatorname{rank}(K) = \operatorname{rank}(K) - \operatorname{rank}(K \cap \ker(\phi_k)) = \operatorname{rank}(\phi_k(K));$$

moreover, since $t_k = 1$,

$$\operatorname{rank}(H) = \operatorname{rank}(K) + \operatorname{rank}(F) \leq t_k \operatorname{rank}(\phi_k(K)) + \sum_{j\neq k} t_j \operatorname{rank}(\phi_j(F)) \leq \sum_j t_j \operatorname{rank}(\phi_k(H)).$$

Thus we have identified $t \in \mathcal{P}(\mathcal{G})$ which, relative to s, has one fewer component greater than one. We can repeat this argument for each such component to show that $(\min\{1,s_j\})_j \in \mathcal{P}(\mathcal{G})$. The conclusion for all $t \geq (\min\{1,s_j\})_j$ follows from Lemma 4.5. □

That is, $\mathcal{P}(\mathcal{G})$ is completely specified by $\mathcal{P}(\mathcal{G}) \cap [0,1]^m$. As mentioned in Section 4.4.4, we prefer to analyze $\mathcal{P}(\mathcal{G}) \cap [0,1]^m$ because it equals the convex hull of its extreme points.

Lemma 4.28. *If $s \in [0,1]^m$ has exactly one component $s_k \in (0,1)$ and there exists $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ such that $\operatorname{rank}(G) = \operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H))$ and $\operatorname{rank}(\phi_k(H)) > 0$, then there exists $K \in \mathcal{H}^*_{\mathbb{N}}(G)$ such that $\operatorname{rank}(K) > \sum_j s_j \operatorname{rank}(\phi_j(K))$.*

Proof. Note that these hypotheses imply $G \in \mathcal{H}^*_{\mathbb{N}}(G)$. Letting $I = \{j \mid s_j = 1\}$, we rewrite a hypothesis regarding H,

$$\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H)) = s_k \operatorname{rank}(\phi_k(H)) + \sum_{i \in I} s_i \operatorname{rank}(\phi_i(H)).$$

If $I = \emptyset$ then since $\operatorname{rank}(\phi_k(H)) > 0$, $s_k = \operatorname{rank}(H)/\operatorname{rank}(\phi_k(H))$. If $\operatorname{rank}(\phi_k(H)) = \operatorname{rank}(H)$ then $s_k = 1$, a contradiction to the hypothesis that $s_k \in (0,1)$. So it must be that $\operatorname{rank}(\phi_k(H)) < \operatorname{rank}(H)$, so $\operatorname{rank}(\ker(\phi_k)) > 0$; taking $K = \ker(\phi_k) \in \mathcal{H}^*_{\mathbb{N}}(G)$,

$$\operatorname{rank}(K) > 0 = s_k \operatorname{rank}(\phi_k(K)) = \sum_j s_j \operatorname{rank}(\phi_k(K)).$$

Otherwise, supposing $I \neq \emptyset$, by the hypotheses s_k and $\operatorname{rank}(\phi_k(H)) > 0$,

$$\sum_{i \in I} \operatorname{rank}(\phi_i(H)) = \sum_{i \in I} s_i \operatorname{rank}(\phi_i(H)) = \operatorname{rank}(H) - s_k \operatorname{rank}(\phi_k(H)) < \operatorname{rank}(H).$$

We now show that the existence of H implies $\operatorname{rank}(\bigcap_{i \in I} \ker(\phi_i)) > 0$. Consider the homomorphism $\Phi \colon G \to \bigoplus_{i \in I} \phi_i(G) \colon x \mapsto (\phi_i(x))_{i \in I}$, and let $K = \ker(\Phi) \in \mathcal{H}^*_{\mathbb{N}}(G)$. Since $\bigcap_{i \in I} \ker(\phi_i) = K$, $\operatorname{rank}(H) = \operatorname{rank}(G) < \infty$, $\operatorname{rank}(\Phi(G)) \leq \operatorname{rank}(\bigoplus_{i \in I} \phi_i(G)) = \sum_{i \in I} \operatorname{rank}(\phi_i(G))$, and Lemma 4.3,

$$\begin{aligned} \operatorname{rank}(K) &= \operatorname{rank}(G) - \operatorname{rank}(\Phi(G)) \geq \operatorname{rank}(G) - \sum_{i \in I} \operatorname{rank}(\phi_i(G)) \\ &= \operatorname{rank}(H) - \sum_{i \in I} \operatorname{rank}(\phi_i(H)) = s_k \operatorname{rank}(\phi_k(H)) > 0. \end{aligned}$$

Because $\phi_i(K) = \{0\}$ for all $i \in I$,

$$\begin{aligned} \sum_j s_j \operatorname{rank}(\phi_j(K)) &= s_k \operatorname{rank}(\phi_k(K)) + \sum_{i \in I} \operatorname{rank}(\phi_i(K)) \\ &= s_k \operatorname{rank}(\phi_k(K)) \leq s_k \operatorname{rank}(K) \end{aligned}$$

Since $s_k < 1$ and $\operatorname{rank}(K) > 0$, we have that $\operatorname{rank}(K) > \sum_j s_j \operatorname{rank}(\phi_j(K))$. □

Lemma 4.29. *If $\operatorname{rank}(G) < \infty$ and s is an extreme point of $\mathcal{P}(\mathcal{G}) \cap [0,1]^m$, then $s \in \{0,1\}^m$ or there exists $H \in \mathcal{H}^*_{\mathbb{N}}(G) \setminus \mathcal{H}^*_0(G)$ such that $\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H)) < \operatorname{rank}(G)$.*

Proof. Suppose toward a contradiction that neither case applies, i.e., that some component $s_k \in (0,1)$ and there does not exist $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ such that $0 < \operatorname{rank}(H) < \operatorname{rank}(G)$ and $\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H))$. Under these hypotheses, by Lemma 4.21, there is exactly one such index k. Additionally, under these hypotheses, by the contrapositive of Lemma 4.20

(noting that $\operatorname{rank}(G) < \infty$ implies $\sup_{H \in \mathcal{H}^*_{\mathbb{N}}(G)} \operatorname{rank}(H) < \infty$), there exists $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ with $\operatorname{rank}(H) = \operatorname{rank}(G)$ and $\operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H))$. By Lemma 4.19, $\operatorname{rank}(\phi_k(H)) > 0$, and finally by Lemma 4.28, we obtain $K \in \mathcal{H}^*_{\mathbb{N}}(G)$ such that $\operatorname{rank}(K) > \sum_j s_j \operatorname{rank}(\phi_j(K))$, which contradicts the hypothesis that $s \in \mathcal{P}(\mathcal{G})$. □

The *product case* refers to a special class of HBL data where $\mathcal{P}(\mathcal{G})$ is defined by a simpler set of inequalities. Here we consider the product case in the context of finitely generated torsion-free Abelian groups; a more general setting was considered in [9, Section 7]. The following result is a special case of the more general result [9, Proposition 7.1].

Lemma 4.30. *Consider any HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$ for some $r \in \mathbb{N}$; suppose there exists a $\mathbb{Z}$-basis $B = \{e_1, \dots, e_r\}$ of $\mathbb{Z}^r$ such that each $\ker(\phi_j) = \langle K_j \rangle$ for some $K_j \subseteq B$. Let $\mathcal{H} = \{H_1 = \langle e_1 \rangle, \dots, H_r = \langle e_r \rangle\}$. Then for any $s \in [0,\infty)^m$, $s \in \mathcal{P}(\mathcal{G})$ if and only if $s \in \mathcal{P}(\mathcal{G}, \mathcal{H})$.*

Proof. Necessity follows from Lemma 4.6, i.e., the inequalities defining $\mathcal{P}(\mathcal{G}, \mathcal{H})$ also appear in the definition of $\mathcal{P}(\mathcal{G})$.

To show sufficiency, rewrite the inequality for each $H_i \in \mathcal{H}$,

$$1 = \operatorname{rank}(H_i) \le \sum_j s_j \operatorname{rank}(\phi_j(H_i)) = \sum_j s_j \delta_{i,j},$$

where $\delta_{i,j} = 1_{K_j}(e_i)$, i.e., $\delta_{i,j} = 1$ if $e_i \in K_j$, and $\delta_{i,j} = 0$ otherwise. Now consider any $H \le \mathbb{Z}^r$, and let $h = \operatorname{rank}(H)$. Let M_H be an r-by-h $\mathbb{Z}$-basis matrix of H (coordinates with respect to B). Since the rank of M_H is h, at least one h-by-h minor of M_H is nonzero; pick any such minor and let $R \subseteq \{1, \dots, r\}$ comprise the h corresponding row indices of M_H. It follows that $\operatorname{rank}(\phi_j(H)) \ge \sum_{i \in R} \delta_{i,j}$. Therefore,

$$\sum_j s_j \operatorname{rank}(\phi_j(H)) \ge \sum_j s_j \sum_{i \in R} \delta_{i,j} = \sum_{i \in R} \sum_j s_j \delta_{i,j} \ge \sum_{i \in R} \operatorname{rank}(H_i) = \sum_{i \in R} 1 = h = \operatorname{rank}(H).$$

□

4.5 Analysis of A, B, and C

It is useful to define $A(\mathcal{G}, s, H)$, $B(\mathcal{G}, s, E)$, and $C(\mathcal{G}, s, f)$ to denote the values of A, B, and C in the individual inequalities in (4.5), (4.6), and (4.7) such that equality holds. Thus, $A(\mathcal{G}, s)$, $B(\mathcal{G}, s)$, and $C(\mathcal{G}, s)$ are the suprema of $A(\mathcal{G}, s, H)$, $B(\mathcal{G}, s, E)$, and $C(\mathcal{G}, s, f)$ over the corresponding sets of subgroups H, subsets E, and m-functions f.

In the case of A, recalling from Section 4.4 that $\mathcal{H}^*_{\mathrm{f}}(G)$ comprises the finite subgroups of G, for any $s \in [0,\infty)^m$ and any $H \in \mathcal{H}^*_{\mathrm{f}}(G)$, we define

$$A(\mathcal{G}, s, H) = B(\mathcal{G}, s, H) = \frac{|H|}{\prod_j |\phi_j(H)|^{s_j}} \in (0, \infty). \tag{4.9}$$

Similarly in the case of B, letting $\mathcal{E}^*(G)$ denote the set of all nonempty finite subsets of G, for any $s \in [0,\infty)^m$ and any $E \in \mathcal{E}^*(G)$, we define

$$B(\mathcal{G}, s, E) = C(\mathcal{G}, s, (1_{\phi_j(E)})_j) = \frac{|E|}{\prod_j |\phi_j(E)|^{s_j}} \in (0, \infty). \tag{4.10}$$

In the cases of both (4.5) and (4.6), there are well-defined minimum values A and B for each individual inequality (for each $H \in \mathcal{H}_{\mathrm{f}}^*(G)$ and $E \in \mathcal{E}^*(G)$, respectively). However, in the case of (4.7), there are two degenerate classes of m-functions where a minimum value for C is not well defined. In the first degenerate class of m-functions f, $\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} = 0$, i.e., one or more functions $f_j = 0$; in this case, equality holds in (4.7) with any C. In the second degenerate class of m-functions f, $\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} = \infty$; in this case, equality holds in (4.7) with any $C > 0$, recalling the convention that $0 \cdot \infty = 0$. There is no loss of generality to ignore these classes of m-functions since they do not influence the value of $C(\mathcal{G}, s)$: to see this, consider the m-function with $f_j = 1_{\{\phi_j(0)\}} = 1_{\{0\}}$: (4.7) holds with equality with $C = 1$, so $C(\mathcal{G}, s) \geq 1$, and we can take C in both degenerate cases to be less than 1. Since $\prod_j$ is a finite product, $\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} \in (0, \infty)$ is equivalent to $\|f_j\|_{\ell^{1/s_j}(G_j)} \in (0, \infty)$ for all j. With these observations, we collect the nondegenerate m-functions,

$$\mathcal{F}^*(\mathcal{G}, s) = \left\{ (f_j)_j \in \bigtimes_j (G_j \to [0, \infty]) \,\middle|\, \|f_j\|_{\ell^{1/s_j}(G_j)} \in (0, \infty) \right\}.$$

With this notation, for any $s \in [0, \infty)^m$ and any $f \in \mathcal{F}^*(\mathcal{G}, s)$, we can now define

$$C(\mathcal{G}, s, f) = \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} \in [0, \infty]; \tag{4.11}$$

note that we still permit m-functions f where $\operatorname{supp}(\prod_j f_j \circ \phi_j) = \bigcap_j \operatorname{supp}(f_j \circ \phi_j) = \emptyset$: these are the only $f \in \mathcal{F}^*(\mathcal{G}, s)$ where $C(\mathcal{G}, s, f) = 0$. In conclusion, we have

$$A(\mathcal{G}, s) = \sup_{H \in \mathcal{H}_{\mathrm{f}}^*(G)} A(\mathcal{G}, s, H), \qquad B(\mathcal{G}, s) = \sup_{E \in \mathcal{E}^*(G)} B(\mathcal{G}, s, E), \qquad C(\mathcal{G}, s) = \sup_{f \in \mathcal{F}^*(\mathcal{G}, s)} C(\mathcal{G}, s, f).$$

4.5.1 Basic Properties of $A(\mathcal{G}, s)$, $B(\mathcal{G}, s)$, and $C(\mathcal{G}, s)$

Lemma 4.31. *For any $s \in [0, \infty)^m$, $1 \leq A(\mathcal{G}, s) \leq B(\mathcal{G}, s) \leq C(\mathcal{G}, s) \leq |G|$.*

Proof. Considering the trivial subgroup $\{0\} \in \mathcal{H}_{\mathrm{f}}^*(G)$, we have that $A(\mathcal{G}, s) \geq A(\mathcal{G}, s, \{0\}) = 1$. Then we have that $A(\mathcal{G}, s) \leq B(\mathcal{G}, s) \leq C(\mathcal{G}, s)$, since their defining sets are nested. Lastly, note that for any $f \in \mathcal{F}^*(\mathcal{G}, s)$,

$$\begin{aligned}
\sum_{x \in G} \prod_j f_j(\phi_j(x)) &\leq |G| \sup \left\{ \prod_j f_j(\phi_j(x)) \,\middle|\, x \in G \right\} \leq |G| \prod_j \sup \left\{ f_j(\phi_j(x)) \,\middle|\, x \in G \right\} \\
&= |G| \prod_j \sup \left\{ f_j(y_j) \,\middle|\, y_j \in \phi_j(G) \right\} \leq |G| \prod_j \sup \left\{ f_j(y_j) \,\middle|\, y_j \in G_j \right\} \\
&= |G| \prod_j \left\| f_j \right\|_{\ell^\infty(G_j)} \leq |G| \prod_j \left\| f_j \right\|_{\ell^{1/s_j}(G_j)},
\end{aligned}$$

so $C(\mathcal{G}, s) \leq |G|$. □

A function $F \colon [0, \infty)^m \to [0, \infty]$ is antitone if, for each $s, t \in [0, \infty)^m$, $s \leq t$ (componentwise) implies $F(s) \geq F(t)$.

Lemma 4.32. *$A(\mathcal{G}, s)$, $B(\mathcal{G}, s)$ and $C(\mathcal{G}, s)$ are antitone functions of $s \in [0, \infty)^m$.*

Proof. For each $H \in \mathcal{H}^*_{\mathrm{f}}(G)$, since $1 \le |\phi_j(H)| \le |H| < \infty$ for each j, $A(\mathcal{G}, s, H)$ is a nonincreasing function of each component $s_j \in [0, \infty)$, and thus $A(\mathcal{G}, s)$ is an antitone function of $s \in [0, \infty)^m$. Similarly, for each $E \in \mathcal{E}^*(G)$, since $1 \le |\phi_j(E)| \le |E| < \infty$ for each j, $B(\mathcal{G}, s, E)$ is a nonincreasing function of each component $s_j \in [0, \infty)$, and thus $B(\mathcal{G}, s)$ is an antitone function of $s \in [0, \infty)^m$. More generally, for any discrete measure space $(X, 2^X, |\cdot|)$, any $g \colon X \to \mathbb{C} \cup \{\infty\}$, and any $p, q \in [0, \infty)$, $p \le q$ implies $\|g\|_{\ell^{1/p}(X)} \le \|g\|_{\ell^{1/q}(X)}$. Thus, for any m-exponents s, t and any m-function f, $s \le t$ implies that $\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} \le \prod_j \|f_j\|_{\ell^{1/t_j}(G_j)}$ and, if $f \in \mathcal{F}^*(\mathcal{G}, t)$, then $f \in \mathcal{F}^*(\mathcal{G}, s)$. We conclude that $C(\mathcal{G}, s) \ge C(\mathcal{G}, t)$ by taking suprema over $\mathcal{F}^*(\mathcal{G}, s)$ and $\mathcal{F}^*(\mathcal{G}, t)$. □

Lemma 4.33. *If $s \in [0, \infty)^m$ then $A(\mathcal{G}, s) \le |T(G)|$.*

Proof. If $H \in \mathcal{H}^*_{\mathrm{f}}(G)$ then $H \le T(G)$. □

Lemma 4.34. *If $K \le \bigcap_j \ker(\phi_j)$, $|K| < \infty$, and $s \in [0, \infty)^m$, then $A(\mathcal{G}, s) \ge |K|$.*

Proof. $A(\mathcal{G}, s) \ge A(\mathcal{G}, s, K) = |K| \prod_j |\phi_j(K)|^{-s_j} = |K| \cdot 1$. □

Lemma 4.35. *If $K \le \bigcap_j \ker(\phi_j)$, $|K| = \infty$, and $s \in [0, \infty)^m$, then $B(\mathcal{G}, s) = \infty$.*

Proof. K has a sequence $E_1, E_2, \ldots$ of nonempty subsets of increasing cardinality; each $E_i \in \mathcal{E}^*(G)$, so $B(\mathcal{G}, s, E_i) = |E_i| \prod_j |\phi_j(E_i)|^{-s_j} = |E_i|$, so $B(\mathcal{G}, s) = \infty$. □

Lemma 4.36. *If $I = \{j \mid s_j \ge 1\}$, $K = \bigcap_{i \in I} \ker(\phi_i) \in \mathcal{H}^*_{\mathrm{f}}(G)$, and $s \in [0, \infty)^m$, then $C(\mathcal{G}, s) \le |K|$.*

Proof. If $I = \emptyset$, then $K = G$, and we have that $C(\mathcal{G}, s) \le |G|$ by Lemma 4.31, regardless of $|K|$. Otherwise, for any m-tuple $f = (f_j)_j$ of functions $f_j \colon G_j \to \mathbb{C} \cup \{\infty\}$, because $G \to \bigtimes_j G_j \colon x \mapsto (\phi_i(x))_{i \in I}$ is $|K|$-to-1,

$$\prod_{i \in I} \|f_i\|_{\ell^{1/s_i}(G_i)} \ge \prod_{i \in I} \|f_i\|_{\ell^1(G_i)} = \sum_{(y_i)_i \in \bigtimes_{i \in I} G_i} \prod_{i \in I} |f_i(y_i)| \ge \frac{1}{|K|} \sum_{x \in G} \prod_{i \in I} |f_i(\phi_i(x))|.$$

Therefore,

$$\begin{aligned} \sum_{x \in G} \prod_j |f_j(\phi_j(x))| &\le \Big(\prod_{j \notin I} \|f_j\|_{\ell^\infty(G_j)} \Big) \sum_{x \in G} \prod_{i \in I} |f_i(\phi_i(x))| \\ &\le \Big(\prod_{j \notin I} \|f_j\|_{\ell^\infty(G_j)} \Big) \Big(|K| \prod_{i \in I} \|f_i\|_{\ell^{1/s_i}(G_i)} \Big) \le |K| \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}, \end{aligned}$$

so, specializing for m-functions $f \in \mathcal{F}^*(\mathcal{G}, s)$, we see that $C(\mathcal{G}, s) \le |K|$. □

Lemma 4.37. *If $s \in [0, \infty)^m$, $f \in \mathcal{F}^*(\mathcal{G}, s)$, and $\emptyset \ne I \subseteq \{1, \ldots, m\}$, then $C(\mathcal{G}, s, f) \le C((\phi_i)_{i \in I}, (s_i)_{i \in I}, (f_i)_{i \in I})$, with equality when $s_j = 0$ or $\phi_j(G) = \{0\}$ for each $j \notin I$.*

Proof. The inequality follows from the definition of $C(\mathcal{G}, s, f)$,

$$C(\mathcal{G}, s, f) = \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = \sum_{x \in G} \Big(\prod_{i \in I} \frac{f_i(\phi_i(x))}{\|f_i\|_{\ell^{1/s_i}(G_i)}} \Big) \Big(\prod_{j \notin I} \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} \Big)$$
$$\leq \sum_{x \in G} \Big(\prod_{i \in I} \frac{f_i(\phi_i(x))}{\|f_i\|_{\ell^{1/s_i}(G_i)}} \Big) \cdot 1 = C((\phi_i)_{i \in I}, (s_i)_{i \in I}, (f_i)_{i \in I}),$$

and the fact that the projection $\mathcal{F}^*(\mathcal{G}, s) \to \mathcal{F}^*((\phi_i)_{i \in I}, (s_i)_{i \in I})\colon f \mapsto (f_i)_{i \in I}$ is surjective.

To show equality when $s_j = 0$ or $\phi_j(G) = \{0\}$, for all $j \notin I$, define the m-function g with $g_i = f_i$ for each $i \in I$ and $g_j(y) = \|f_j\|_{\ell^{1/s_j}(G_j)} \cdot 1_{\phi_j(G)}(y)$ for each $j \notin I$. For each $j \notin I$, we observe in both cases $s_j = 0$ and $\phi_j(G) = \{0\}$ that $\|g_j\|_{\ell^{1/s_j}(G_j)} = \|f_j\|_{\ell^{1/s_j}(G_j)}$, and, for all $x \in G$, $g_j(\phi_j(x)) = \|f_j\|_{\ell^{1/s_j}(G_j)}$. Thus we confirm that $g \in \mathcal{F}^*(\mathcal{G}, s)$, and, for all $x \in G$, that

$$\prod_{j \notin I} \frac{g_j(\phi_j(x))}{\|g_j\|_{\ell^{1/s_j}(G_j)}} = \prod_{j \notin I} \frac{\|f_j\|_{\ell^{1/s_j}(G_j)}}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = 1.$$

Therefore,

$$C(\mathcal{G}, s, f) \leq C(\mathcal{G}, s, g) = C((\phi_i)_{i \in I}, (s_i)_{i \in I}, (g_i)_{i \in I}) = C((\phi_i)_{i \in I}, (s_i)_{i \in I}, (f_i)_{i \in I});$$

since for any $f \in \mathcal{F}^*(\mathcal{G}, s)$ we can find such a $g \in \mathcal{F}^*(\mathcal{G}, s)$, the second part of the conclusion (equality) follows by taking suprema over $f \in \mathcal{F}^*(\mathcal{G}, s)$. □

Lemma 4.38. *For all $s \in [0, \infty)^m$, $C(\mathcal{G}, s)$ equals the supremum of $C(\mathcal{G}, s, f^E)$ over all $E \in \mathcal{E}^*(G)$ and $f^E \in \mathcal{F}^*(\mathcal{G}, s)$ where each* $\operatorname{supp}(f_j^E) = \phi_j(E)$.

Proof. Consider any $f \in \mathcal{F}^*(\mathcal{G}, s)$, and for each $E \in \mathcal{E}^*(G)$, query whether $\sum_{x \in E} \prod_j f_j(\phi_j(x)) > 0$. If so, it follows that $\|f_j|_{\phi_j(E)}\|_{\ell^{1/s_j}(\phi_j(E))} > 0$; thus, we can define another m-function f^E with components

$$f_j^E(y) = \frac{\|f_j\|_{\ell^{1/s_j}(G_j)}}{\|f_j|_{\phi_j(E)}\|_{\ell^{1/s_j}(\phi_j(E))}} \cdot f_j(y) \cdot 1_{\phi_j(E)}(y);$$

it follows that $\|f_j^E\|_{\ell^{1/s_j}(G_j)} = \|f_j\|_{\ell^{1/s_j}(G_j)}$, and so $f^E \in \mathcal{F}^*(\mathcal{G}, s)$ as well. Otherwise, if $\sum_{x \in E} \prod_j f_j(\phi_j(x)) = 0$, set each $f_j^E = 0$, so $f^E \notin \mathcal{F}^*(\mathcal{G}, s)$. By the definitions of $C(\mathcal{G}, s, \cdot)$ and the Lebesgue integral for counting measure,

$$C(\mathcal{G}, s, f) = \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = \sup_{E \in \mathcal{E}^*(G)} \sum_{x \in E} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = \sup_{\substack{E \in \mathcal{E}^*(G) \\ f^E \in \mathcal{F}^*(\mathcal{G}, s)}} \sum_{x \in E} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}}$$
$$= \sup_{\substack{E \in \mathcal{E}^*(G) \\ f^E \in \mathcal{F}^*(\mathcal{G}, s)}} \sum_{x \in G} \prod_j \frac{f_j^E(\phi_j(x))}{\|f_j^E\|_{\ell^{1/s_j}(G_j)}} = \sup_{\substack{E \in \mathcal{E}^*(G) \\ f^E \in \mathcal{F}^*(\mathcal{G}, s)}} C(\mathcal{G}, s, f^E).$$

The conclusion follows by taking a supremum over $f \in \mathcal{F}^*(\mathcal{G}, s)$, redefining its associated m-functions f^E appropriately. □

4.5.2 Properties of $B(\mathcal{G}, s)$ and $C(\mathcal{G}, s)$ when $s \in [0,1]^m$

Lemma 4.39. *If $s \in [0,\infty)^m$ and $r = (\min(1, s_j))_j$, then $B(\mathcal{G}, s) = B(\mathcal{G}, r)$ and $C(\mathcal{G}, s) = C(\mathcal{G}, r)$.*

Proof. We prove the second conclusion first, and then specialize that proof to obtain the first conclusion.

Suppose $s_k > 1$ for some $k \in \{1, \ldots, m\}$ and define $t \in [0,\infty)^m$ such that $t_k = 1$ and $t_j = s_j$ for $j \neq k$. Consider any $E \in \mathcal{E}^*(G)$ and any m-function f where for each j, $0 \neq f_j < \infty$ and $\operatorname{supp}(f_j) = \phi_j(E)$. Therefore, $f \in \mathcal{F}^*(\mathcal{G}, r)$ for all $r \in [0,\infty)^m$, in particular, for $r = s$ and $r = t$. For each $y \in \phi_k(E)$, let $E_y = E \cap \phi_k^{-1}(y)$, and define another m-function f^y with components

$$f_j^y(z) = \begin{cases} f_j(z) \neq 0 & z \in \phi_j(E_y) \\ 0 & z \notin \phi_j(E_y). \end{cases}.$$

It follows that $f_k = \sum_{y \in \phi_k(E)} f_k^y$, $\bigcap_j \operatorname{supp}(f_j^y \circ \phi_j) = E_y$, and $\operatorname{supp}(f_j^y) = \phi_j(E_y)$; in particular, $\operatorname{supp}(f_k^y) = \{y\}$. Therefore, for each j, $0 < \|f_j^y\|_{\ell^{1/s_j}(G_j)} \leq \|f_j\|_{\ell^{1/s_j}(G_j)} < \infty$, and

$$\begin{aligned} C(\mathcal{G}, s, f^y) &= \sum_{x \in G} \prod_j \frac{f_j^y(\phi_j(x))}{\|f_j^y\|_{\ell^{1/s_j}(G_j)}} = \sum_{x \in E_y} \prod_j \frac{f_j^y(\phi_j(x))}{\|f_j^y\|_{\ell^{1/s_j}(G_j)}} \\ &= \left(\sum_{x \in E_y} \prod_{j \neq k} \frac{f_j^y(\phi_j(x))}{\|f_j^y\|_{\ell^{1/s_j}(G_j)}} \right) \cdot \frac{f_k^y(y)}{\|f_k^y\|_{\ell^{1/s_k}(G_k)}} = \left(\sum_{x \in E_y} \prod_{j \neq k} \frac{f_j^y(\phi_j(x))}{\|f_j^y\|_{\ell^{1/s_j}(G_j)}} \right) \cdot \frac{f_k^y(y)}{f_k^y(y)} \\ &= \sum_{x \in E_y} \prod_{j \neq k} \frac{f_j^y(\phi_j(x))}{\|f_j^y\|_{\ell^{1/t_j}(G_j)}} \geq \sum_{x \in E_y} \prod_{j \neq k} \frac{f_j^y(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}}. \end{aligned}$$

We will substitute this inequality in the definition of $C(\mathcal{G}, t, f)$,

$$\begin{aligned} C(\mathcal{G}, t, f) &= \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}} = \sum_{x \in E} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}} = \sum_{y \in \phi_k(E)} \sum_{x \in E_y} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}} \\ &= \sum_{y \in \phi_k(E)} \sum_{x \in E_y} \prod_j \frac{f_j^y(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}} = \sum_{y \in \phi_k(E)} \left(\sum_{x \in E_y} \prod_{j \neq k} \frac{f_j^y(\phi_j(x))}{\|f_j\|_{\ell^{1/t_j}(G_j)}} \right) \cdot \frac{f_k^y(y)}{\|f_k\|_{\ell^1(G_k)}} \\ &\leq \sum_{y \in \phi_k(E)} C(\mathcal{G}, s, f^y) \cdot \frac{f_k^y(y)}{\|f_k\|_{\ell^1(G_k)}} \leq \left(\max_{y \in \phi_k(E)} C(\mathcal{G}, s, f^y) \right) \cdot \sum_{y \in \phi_k(E)} \frac{f_k^y(y)}{\|f_k\|_{\ell^1(G_k)}} \\ &= \left(\max_{y \in \phi_k(E)} C(\mathcal{G}, s, f^y) \right) \cdot \frac{\|f_k^y\|_{\ell^1(G_k)}}{\|f_k\|_{\ell^1(G_k)}} \leq \left(\max_{y \in \phi_k(E)} C(\mathcal{G}, s, f^y) \right) \cdot 1 \leq C(\mathcal{G}, s). \end{aligned}$$

Taking a supremum over all $E \in \mathcal{E}^*(G)$ and all $f \in \mathcal{F}^*(\mathcal{G}, t)$ such that each $\operatorname{supp}(f_j) = \phi_j(E)$ and then applying Lemma 4.38, we have that $C(\mathcal{G}, t) \leq C(\mathcal{G}, s)$, thus by Lemma 4.32, $C(\mathcal{G}, t) = C(\mathcal{G}, s)$. This argument can be repeated for each index k such that $s_k > 1$ to obtain $t = (\min(1, s_j))_j$.

We obtain the first conclusion $B(\mathcal{G}, s) = B(\mathcal{G}, (\min(1, s_j))_j)$ as a special case of the preceding argument. In more detail, for each $E \in \mathcal{E}^*(G)$, we consider just the m-function

f with components $f_j = 1_{\phi_j(E)}$, so $C(\mathcal{G}, t, f) = B(\mathcal{G}, t, E)$; it follows that its associated m-functions f^y have components $f_j^y = 1_{\phi_j(E_y)}$, so $C(\mathcal{G}, s, f^y) = B(\mathcal{G}, s, E_y)$, therefore $\max_{y \in \phi_k(E)} C(\mathcal{G}, s, f^y) \leq B(\mathcal{G}, s)$. We then again take a supremum over all $E \in \mathcal{E}^*(G)$, obtaining $B(\mathcal{G}, t, E) \leq B(\mathcal{G}, s)$ for each, so $B(\mathcal{G}, t) \leq B(\mathcal{G}, s)$, and thus by Lemma 4.32, $B(\mathcal{G}, t) = B(\mathcal{G}, s)$. □

In light of this result, it suffices to study $B(\mathcal{G}, s)$ and $C(\mathcal{G}, s)$ for just $s \in [0,1]^m$, rather than all $s \in [0,\infty)^m$: in particular, their values on the m faces of $[0,1]^m$ that intersect $(1, \ldots, 1)$ determine their values for all $s \in [0,\infty)^m \setminus [0,1]^m$.

Lemma 4.40. *If S is a nonempty closed subset of $[0,1]^m$, $c \in [1,\infty)$, and $C(\mathcal{G}, s) \leq c$ for all $s \in S$, then $C(\mathcal{G}, s) \leq c$ for all s in the (closed) convex hull of S.*

Proof. We will use some additional measure-theoretic notation in the following proof, in order to apply a multilinear variant of the Riesz-Thorin interpolation theorem (see, e.g., [8, Theorem 2.7]).

Given a measure space (X, Σ, μ), a function $f \colon X \to \mathbb{C}$ is Σ-simple if $f = \sum_{i=1}^n c_i 1_{E_i}$ for some $n \in \{1, 2, \ldots\}$, $c_1, \ldots, c_n \in \mathbb{C}$, and $E_1, \ldots, E_n \in \Sigma$.

Let $(X_j)_j$ be an m-tuple of sets; a function $T \colon \bigtimes_j (X_j \to \mathbb{C}) \to \mathbb{C}$ is a multilinear form if, for any $f \in \bigtimes_j (X_j \to \mathbb{C})$ with component $f_k = cg + dh$ for $c, d \in \mathbb{C}$ and $g, h \colon X_k \to \mathbb{C}$,

$$T(f) = cT(f_1, \ldots, f_{k-1}, g, f_{k+1}, \ldots, f_m) + dT(f_1, \ldots, f_{k-1}, h, f_{k+1}, \ldots, f_m).$$

(T does not denote a torsion subgroup, as it does elsewhere.)

Now let $((X_j, \Sigma_j, \mu_j))_j$ be an m-tuple of measure spaces and let S_j denote the set of Σ_j-simple functions $f_j \colon X_j \to \mathbb{C}$. With this notation, a variant of the Riesz-Thorin interpolation theorem states the following: given a multilinear form T and $r, t \in [0,1]^m$, if there exist $\rho, \tau \in [0,\infty)$ such that

$$|T(f)| \leq \rho \prod_j \|f\|_{L^{1/r_j}(X_j, \Sigma_j, \mu_j)} \quad \text{and} \quad |T(f)| \leq \tau \prod_j \|f_j\|_{L^{1/t_j}(X_j, \Sigma_j, \mu_j)} \quad \text{for all } f \in \bigtimes_j S_j,$$

then, for any $\theta \in (0,1)$, letting $\sigma = \rho^\theta \tau^{1-\theta}$ and $s = \theta r + (1-\theta)t$,

$$|T(g)| \leq \sigma \prod_j \|g_j\|_{L^{1/s_j}(X_j, \Sigma_j, \mu_j)} \quad \text{for all } f \in \bigtimes_j S_j.$$

To apply this result in the present setting of the discrete Abelian groups G, G_j, we will consider the multilinear form $T(f) = \sum_{x \in G} \prod_j f_j(\phi_j(x))$, where $f \in \bigtimes_j (G_j \to \mathbb{C})$. Here we abuse our notation to suppose, for all $s \in [0,\infty)^m$, $\mathcal{F}^*(\mathcal{G}, s) \subseteq \bigtimes_j (G_j \to \mathbb{C})$: this embedding is possible because for each $f \in \mathcal{F}^*(\mathcal{G}, s)$, each of its components $f_j < \infty$, so we can restrict the codomain each f_j from $[0,\infty]$ to $[0,\infty)$ and then embed $[0,\infty)$ in $\mathbb{C}$.

We first show that $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ for all $f \in \bigtimes_j S_j$ and all $s \in S$. Consider any $s \in S$ and any $f \in \bigtimes_j S_j$. Suppose first that $(|f_j|)_j \notin \mathcal{F}^*(\mathcal{G}, s)$: this leads to the following two cases. In the first case, some $f_j = 0$, in which case $|T(f)| = 0 = c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$. In the second case, some $\|f_j\|_{\ell^{1/s_j}(G_j)} = \infty$; assuming the first case does not apply, we have

that $|T(f)| \leq \infty = c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$. Otherwise, suppose that $(|f_j|)_j \in \mathcal{F}^*(\mathcal{G}, s) \cap \bigtimes_j S_j$ and apply the hypothesis $C(\mathcal{G}, s, (|f_j|)_j) \leq C(\mathcal{G}, s) \leq c$:

$$\begin{aligned}|T(f)| &= \Big| \sum_{x \in G} \prod_j f_j(\phi_j(x)) \Big| \leq \sum_{x \in G} \prod_j |f_j(\phi_j(x))| = C(\mathcal{G}, s, (|f_j|)_j) \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} \\ &\leq C(\mathcal{G}, s) \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)} \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}.\end{aligned}$$

Thus, we confirm that $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ for all $f \in \bigtimes_j S_j$ and all $s \in S$.

Now let $\mathcal{P}$ denote the convex hull of S, the smallest convex set in $\mathbb{R}^m$ containing S: since $S \subseteq [0,1]^m$ is closed and bounded by hypothesis, it folows that $\mathcal{P} \subseteq [0,1]^m$ is also closed and bounded, and moreover each of $\mathcal{P}$'s extreme points is an element of S. By Carathéodory's theorem, any point in $\mathcal{P}$ is a convex combination of at most $m+1$ extreme points. Since the version of the Riesz-Thorin theorem above is stated in terms of pairwise convex combinations rather than general convex combinations, we will apply it iteratively, in a sequence of at most $m+1$ rounds, to obtain the conclusion for all $s \in \mathcal{P}$.

The zeroth round was already established: $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ for all $f \in \bigtimes_j S_j$ at each point s in $\mathcal{P}_0 = S$. For each round $k \in \{1, 2, \ldots\}$, assume the induction hypothesis, that the set $\mathcal{P}_{k-1}$ obtained in the $(k-1)$-th round includes all k-element convex combinations of S, and that $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ for all $f \in \bigtimes_j S_j$ and all $s \in \mathcal{P}_{k-1}$. For the k-th round, we consider all pairwise convex combinations of $r \in \mathcal{P}_{k-1}$ and $t \in \mathcal{P}_0$. By the induction hypothesis (and base case), for all $f \in \bigtimes_j S_j$, $|T(f)| \leq c \prod_j \|f_j\|_{1/r_j}$ for all $r \in \mathcal{P}_{k-1}$ and $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/t_j}(G_j)}$ for all $t \in \mathcal{P}_0$. So, applying the Riesz-Thorin interpolation theorem with T, r, t, and $\rho = \tau = c$, we have that $|T(f)| \leq \sigma \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ with $\sigma = c$ for all $f \in \bigtimes_j S_j$ at all points s on the line segment between r and t (i.e., $s = \theta r + (1-\theta)t$ for $\theta \in [0,1]$). This can be repeated for all s equal to a pairwise convex combination of $r \in \mathcal{P}_{k-1}$ and $t \in \mathcal{P}_0$; the set $\mathcal{P}_k$ of all such s therefore includes all $(k+1)$-element convex combinations of S. After $m+1$ rounds, we have established $|T(f)| \leq c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}$ for all $f \in \bigtimes_j S_j$ and all $s \in \mathcal{P}_m = \mathcal{P}$.

Now, for any $f \in \bigtimes_j S_j$,

$$|T(f)| = \Big| \sum_{x \in G} \prod_j f_j(\phi_j(x)) \Big| \leq \sum_{x \in G} \prod_j |f_j(\phi_j(x))| = T((|f_j|)_j),$$

which holds with equality when each $f_j \geq 0$. Therefore, for any $s \in \mathcal{P}$ and any $f \in \mathcal{F}^*(\mathcal{G}, s) \cap \bigtimes_j S_j$,

$$\begin{aligned}C(\mathcal{G}, s, f) &= \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = \frac{\sum_{x \in G} \prod_j f_j(\phi_j(x))}{\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}} \\ &= \frac{T(f)}{\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}} \leq \frac{c \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}}{\prod_j \|f_j\|_{\ell^{1/s_j}(G_j)}} = c.\end{aligned}$$

Since the simple functions in the discrete setting include all functions with finite supports, applying Lemma 4.38 for each $s \in \mathcal{P}$, we have that $C(\mathcal{G}, s, f) \leq c$ for all $f \in \mathcal{F}^*(\mathcal{G}, s)$; finally, taking a supremum over $\mathcal{F}^*(\mathcal{G}, s)$, we conclude that $C(\mathcal{G}, s) \leq c$ for all $s \in \mathcal{P}$. $\square$

4.5.3 Factorization of HBL Data

We recall the notation introduced in Section 4.4.5 regarding factor data: given an HBL datum $\mathcal{G} = (G, (G_j), (\phi_j))$ and any $H \leq G$, its restricted datum $\mathcal{G}|_H = (H, (\phi_j(H))_j, (\chi_j)_j)$ has homomorphisms $\chi_j \colon H \to \phi_j(H) \colon x \mapsto \phi_j(x)$ and its quotient datum $\mathcal{G}|^H = (G/H, (G_j/\phi_j(H))_j, (\psi_j)_j)$ has homomorphisms $\psi_j \colon G/H \to G_j/\phi_j(H) \colon x + H \mapsto \phi_j(x) + \phi_j(H)$.

Lemma 4.41. *If $s \in [0,\infty)^m$ and $H \leq G$, then $A(\mathcal{G}|_H, s) \leq A(\mathcal{G}, s)$, with equality when $\mathcal{H}^*_{\mathrm{f}}(H) = \mathcal{H}^*_{\mathrm{f}}(G)$.*

Proof. Generally $\mathcal{H}^*_{\mathrm{f}}(H) \subseteq \mathcal{H}^*_{\mathrm{f}}(G)$, and for all $K \in \mathcal{H}^*_{\mathrm{f}}(H)$, $|\chi_j(K)| = |\phi_j(K)|$. □

Lemma 4.42. *If $H \in \mathcal{H}^*_{\mathrm{f}}(G)$, $s \in [0,\infty)^m$, and $A(\mathcal{G}|_H, s, H) = A(\mathcal{G}|_H, s)$, then $A(\mathcal{G}|_H, s)A(\mathcal{G}|^H, s) \leq A(\mathcal{G}, s)$.*

Proof. Every element of $\mathcal{H}^*_{\mathrm{f}}(G/H)$ can be written as K/H for some $H \leq K \in \mathcal{H}^*_{\mathrm{f}}(G)$ (i.e., some $K \in \mathcal{H}^*_{\mathrm{f}}(G)$ such that $H \leq K$); likewise, for each j, each element of $\mathcal{H}^*_{\mathrm{f}}(\phi_j(G)/\phi_j(H))$ can be written as $\phi_j(K)/\phi_j(H) = \psi_j(K/H)$ for some $H \leq K \in \mathcal{H}^*_{\mathrm{f}}(G)$. By Lagrange's theorem, $|K/H| = |K|/|H|$, and, for each j, $|\phi_j(K)/\phi_j(H)| = |\phi_j(K)|/|\phi_j(H)|$. So, for any $H \leq K \in \mathcal{H}^*_{\mathrm{f}}(G)$,

$$\begin{aligned} A(\mathcal{G}, s) \geq A(\mathcal{G}, s, K) &= \frac{|K|}{\prod_j |\phi_j(K)|^{s_j}} = \frac{|H| \cdot |K/H|}{\prod_j (|\phi_j(H)| \cdot |\phi_j(K)/\phi_j(H)|)^{s_j}} \\ &= \left(\frac{|H|}{\prod_j |\chi_j(H)|^{s_j}}\right)\left(\frac{|K/H|}{\prod_j |\psi_j(K/H)|^{s_j}}\right) = A(\mathcal{G}|_H, s, H)A(\mathcal{G}|^H, s, K/H) \\ &= A(\mathcal{G}|_H, s)A(\mathcal{G}|^H, s, K/H). \end{aligned}$$

Repeating for each $H \leq K \in \mathcal{H}^*_{\mathrm{f}}(G)$, the value $A(\mathcal{G}|^H, s, K/H)$ for each $K/H \in \mathcal{H}^*_{\mathrm{f}}(G)$ appears on the right-hand side at least once, so taking a supremum over $H \leq K \in \mathcal{H}^*_{\mathrm{f}}(G)$, the preceding inequality becomes $A(\mathcal{G}, s) \geq A(\mathcal{G}|_H, s)A(\mathcal{G}|^H, s)$. □

Lemma 4.43. *If $H \leq G$ and $s \in [0,\infty)^m$, then $C(\mathcal{G}, s) \leq C(\mathcal{G}|_H, s)C(\mathcal{G}|^H, s)$.*

Proof. First we suppose, without loss of generality, that $s \in (0,\infty)^m$, by application of Lemma 4.37.

Consider any $f \in \mathcal{F}^*(\mathcal{G}, s)$. First consider the quotient datum $\mathcal{G}|^H$. Define the m-function h componentwise by

$$h_j \colon G_j/\phi_j(H) \to [0,\infty) \colon y + \phi_j(H) \mapsto \Bigg(\sum_{z \in \phi_j(H)} f_j(z+y)^{1/s_j}\Bigg)^{s_j};$$

we now check that each component h_j is a function of the coset $y + \phi_j(H)$, meaning that it is independent of the choice of the representative $y \in G_j$: considering any other possible representative $w \in G_j$, we have that $v = w - y \in \phi_j(H)$, so

$$\begin{aligned} h_j(w + \phi_j(H))^{1/s_j} &= \sum_{u \in \phi_j(H)} f_j(u+w)^{1/s_j} = \sum_{u \in \phi_j(H)} f_j(u+v+y)^{1/s_j} \\ &= \sum_{z - v \in \phi_j(H)} f_j(z+y)^{1/s_j} = \sum_{z \in \phi_j(H)} f_j(z+y)^{1/s_j} = h_j(y + \phi_j(H))^{1/s_j}, \end{aligned}$$

since $z - v \in \phi_j(H)$ if and only if $z \in \phi_j(H)$. Moreover,

$$\begin{aligned}\|h_j\|^{1/s_j}_{\ell^{1/s_j}(G_j/\phi_j(H))} &= \sum_{y+\phi_j(H)\in G_j/\phi_j(H)} \Big(\Big(\sum_{z\in\phi_j(H)} f_j(z+y)^{1/s_j}\Big)^{s_j}\Big)^{1/s_j} \\ &= \sum_{y+\phi_j(H)\in G_j/\phi_j(H)} \sum_{z\in\phi_j(H)} f_j(z+y)^{1/s_j} = \sum_{y\in G_j} f_j(y)^{1/s_j} = \|f_j\|^{1/s_j}_{\ell^{1/s_j}(G_j)},\end{aligned}$$

because, letting Y denote the set of coset representatives y appearing in the summation subscript $y+\phi_j(H) \in G_j/\phi_j(H)$, the map $Y \times \phi_j(H) \to G_j \colon (y,z) \mapsto y+z$ is a bijection. It follows that $h \in \mathcal{F}^*(\mathcal{G}|^H, s)$, thus by definition,

$$C(\mathcal{G}|^H, s, h) = \sum_{x+H\in G/H} \prod_j \frac{h_j(\psi_j(x+H))}{\big\|h_j\big\|_{\ell^{1/s_j}(G_j/\phi_j(H))}}$$

Next we consider the restricted datum $\mathcal{G}|_H$. For each $w \in G$, define the m-function $g^{(w)}$ with $g_j^{(w)} \colon \phi_j(H) \to [0,\infty] \colon y \mapsto f_j(y+\phi_j(w))$. For each j, $\|g_j^{(w)}\|_{\ell^{1/s_j}(\phi_j(H))} \le \|f_j\|_{\ell^{1/s_j}(G_j)}$, so there are two mutually exclusive cases. Either $g_j^{(w)} = 0$ for some j, or $g^{(w)} \in \mathcal{F}^*(\mathcal{G}|_H, s)$. In the former case, $\sum_{x\in H} \prod_j g^{(w)}(\phi_j(x)) = 0$; in the latter case,

$$\begin{aligned}C(\mathcal{G}|_H, s, g^{(w)}) &= \sum_{x\in H} \prod_j \frac{g_j^{(w)}(\chi_j(x))}{\|g_j^{(w)}\|_{\ell^{1/s_j}(\phi_j(H)}} = \sum_{x\in H} \prod_j \frac{f_j(\phi_j(x)+\phi_j(w)))}{\big(\sum_{y\in\phi_j(H)} f_j(y+\phi_j(w))^{1/s_j}\big)^{s_j}} \\ &= \sum_{x\in H} \prod_j \frac{f_j(\phi_j(x+w))}{h_j(\psi_j(w+H))}.\end{aligned}$$

Now allowing v to vary over a set of representatives of the cosets $v+H \in G/H$,

$$\begin{aligned}\sum_{x\in G} \prod_j f_j(\phi_j(x)) &= \sum_{v+H\in G/H} \sum_{x\in H} \prod_j f_j(\phi_j(x+v)) \\ &= \sum_{\substack{v+H\in G/H \\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_H,s)}} C(\mathcal{G}|_H, s, g^{(v)}) \cdot \prod_j h_j(\psi_j(v+H)) \\ &\le C(\mathcal{G}|_H, s) \sum_{\substack{v+H\in G/H \\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_H,s)}} \prod_j h_j(\psi_j(v+H)) \\ &\le C(\mathcal{G}|_H, s) \sum_{x+H\in G/H} \prod_j h_j(\psi_j(x+H)) \\ &= C(\mathcal{G}|_H, s) C(\mathcal{G}|^H, s, h) \prod_j \|h_j\|_{\ell^{1/s_j}(G_j/\phi_j(H))} \\ &\le C(\mathcal{G}|_H, s) C(\mathcal{G}|^H, s) \prod_j \|h_j\|_{\ell^{1/s_j}(G_j/\phi_j(H))} \\ &= C(\mathcal{G}|_H, s) C(\mathcal{G}|^H, s) \prod_j \|f_j\|_{\ell^{1/s_j}(G_j)},\end{aligned}$$

so $C(\mathcal{G}, s, f) \le C(\mathcal{G}|_H, s) C(\mathcal{G}|^H, s)$. The conclusion is obtained by repeating this argument for each $f \in \mathcal{F}^*(\mathcal{G}, s)$ and observing the same conclusion each time. □

4.5.4 When G Is Finite

The following three results, in tandem with Lemma 4.42 above, complete our summary of the machinery introduced in [14] to precisely determine $B(\mathcal{G}, s), C(\mathcal{G}, s)$ when G is a finite Abelian group (i.e., a torsion group). In Section 4.6, we synthesize these results with the results from [15] for torsion-free Abelian groups to address the general case. Here we follow [14] and restrict $s \in [0,1]^m$; later, in the proof of Theorem 4.2, we apply Lemmas 4.27 and Lemma 4.39 to address the case of $s \in [0,\infty)^m$.

Lemma 4.44. *If $|G| < \infty$ and $s \in [0,1]^m$ with all $s_j = 0$ except one $s_k \geq 0$, then $A(\mathcal{G}, s) = A(\mathcal{G}, s, G) = |G|^{1-s_k} |\ker(\phi_k)|^{s_k} = C(\mathcal{G}, s)$.*

Proof. Let $K = \ker(\phi_k)$ and consider any $H \leq G$ (noting that $H, K, G \in \mathcal{H}^*_{\mathrm{f}}(G)$):

$$\begin{aligned} A(\mathcal{G}, s, H) &= \frac{|H|}{\prod_j |\phi_j(H)|^{s_j}} = \frac{|H|}{|\phi_k(H)|^{s_k}} = \frac{|H|}{\left(\frac{|H|}{|H \cap K|}\right)^{s_k}} \\ &= |H|^{1-s_k} |H \cap K|^{s_k} \leq |G|^{1-s_k} |H \cap K|^{s_k} \leq |G|^{1-s_k} |K|^{s_k} = A(\mathcal{G}, s, G), \end{aligned}$$

so $A(\mathcal{G}, s) = A(\mathcal{G}, s, G) = |G|^{1-s_k} |K|^{s_k}$, the two equalities in the conclusion.

Now consider any $f \in \mathcal{F}^*(\mathcal{G}, s)$:

$$C(\mathcal{G}, s, f) = \sum_{x \in G} \prod_j \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}} = \sum_{x \in G} \left(\prod_{j \neq k} \frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^\infty(G_j)}} \right) \frac{f_k(\phi_j(x))}{\|f_k\|_{\ell^{1/s_k}(G_k)}} \leq \sum_{x \in G} \frac{f_k(\phi_j(x))}{\|f_k\|_{\ell^{1/s_k}(G_k)}},$$

and, letting V denote a set of representatives of the cosets $v + K \in G/K$,

$$\begin{aligned} \sum_{x \in G} f_k(\phi_k(x)) &= \sum_{v \in V} \sum_{u \in K} f_k(\phi_k(u+v)) = \sum_{v \in V} |K| f_k(\phi_k(v)) = |K| \| f_k|_{\phi_k(V)} \|_{\ell^1(\phi_k(V))} \\ &\leq |K| |\phi_k(V)|^{1-s_k} \| f_k|_{\phi_k(V)} \|_{\ell^{1/s_k}(\phi_k(V))} \leq |K| |\phi_k(V)|^{1-s_k} \|f_k\|_{\ell^{1/s_k}(G_k)} \\ &= |K| |\phi_k(G)|^{1-s_k} \|f_k\|_{\ell^{1/s_k}(G_k)} = |K| \left(\frac{|G|}{|K|} \right)^{1-s_k} \|f_k\|_{\ell^{1/s_k}(G_k)} \\ &= |K|^{s_k} |G|^{1-s_k} \|f_k\|_{\ell^{1/s_k}(G_k)} = A(\mathcal{G}, s) \|f_k\|_{\ell^{1/s_k}(G_k)}, \end{aligned}$$

so $C(\mathcal{G}, s, f) \leq A(\mathcal{G}, s)$ for all $f \in \mathcal{F}^*(\mathcal{G}, s)$, thus $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$. By Lemma 4.31 $A(\mathcal{G}, s) \leq C(\mathcal{G}, s)$, so we conclude $A(\mathcal{G}, s) = C(\mathcal{G}, s)$. □

Lemma 4.45. *If $|G| < \infty$ and $s \in [0,1]^m$ with all $s_j \in \{0,1\}$ except one $s_k \in [0,1]$, then $A(\mathcal{G}, s) = C(\mathcal{G}, s)$.*

Proof. Let $I = \{j \neq k \mid s_j = 1\}$. Let $H = \bigcap_{i \in I} \ker(\phi_i)$ and consider the factor data $\mathcal{G}|_H$ and $\mathcal{G}|^H$.

First we study the restricted datum $\mathcal{G}|_H$ and show that $A(\mathcal{G}|_H, s) = C(\mathcal{G}|_H, s)$. By Lemma 4.37, $C(\mathcal{G}|^H, s) \leq C((\psi_j)_{j \notin I}, (s_j)_{j \notin I})$. Now apply Lemma 4.44 to $(\chi_j)_{j \notin I}$ and $(s_j)_{j \notin I}$:

$$A((\chi_j)_{j \notin I}, (s_j)_{j \notin I}, H) = A((\chi_j)_{j \notin I}, (s_j)_{j \notin I}) = C((\chi_j)_{j \notin I}, (s_j)_{j \notin I}) = |H|^{1-s_k} |\ker(\phi_k)|^{s_k}.$$

Note that $\prod_{i\in I}|\phi_i(H)|^{s_i}=1$, so $A((\chi_j)_{j\notin I},(s_j)_{j\notin I},H)=A(\mathcal{G}|_H,s,H)$ and $A((\chi_j)_{j\notin I},(s_j)_{j\notin I})=A(\mathcal{G}|_H,s)$. Therefore, applying Lemma 4.37,

$$A(\mathcal{G}|_H,s)=A((\chi_j)_{j\notin I},(s_j)_{j\notin I})=C((\chi_j)_{j\notin I},(s_j)_{j\notin I})\geq C(\mathcal{G}|_H,s);$$

by Lemma 4.31, $A(\mathcal{G}|_H,s)\leq C(\mathcal{G}|_H,s)$, so we must have that $A(\mathcal{G}|_H,s)=C(\mathcal{G}|_H,s)$.

Next we study the quotient datum $\mathcal{G}|_H$ and show that $A(\mathcal{G}|^H,s)=C(\mathcal{G}|^H,s)$. Since $H=\bigcap_{i\in I}\ker(\phi_i)$, it follows that $\bigcap_{i\in I}\ker(\psi_i)=\{0\}$. By Lemma 4.36, $C((\psi_i)_{i\in I},(s_i)_{i\in I})\leq 1$. By Lemma 4.37, $C(\mathcal{G}|^H,s)\leq C((\psi_i)_{i\in I},(s_i)_{i\in I})$. By Lemma 4.31, $1\leq A(\mathcal{G}|^H,s)\leq C(\mathcal{G}|^H,s)$, so $A(\mathcal{G}|^H,s)=C(\mathcal{G}|^H,s)$.

Since we concluded above that $A(\mathcal{G}|_H,s,H)=A(\mathcal{G}|_H,s)$, by Lemma 4.42, $A(\mathcal{G},s)\geq A(\mathcal{G}|_H,s)A(\mathcal{G}|^H,s)$. By Lemma 4.43,

$$C(\mathcal{G},s)\leq C(\mathcal{G}|_H,s)C(\mathcal{G}|^H,s)=A(\mathcal{G}|_H,s)A(\mathcal{G}|^H,s)\leq A(\mathcal{G},s);$$

By Lemma 4.31, $A(\mathcal{G},s)\leq C(\mathcal{G},s)$, so it must be that $A(\mathcal{G},s)=C(\mathcal{G},s)$. □

Lemma 4.46. *If $|G|<\infty$ and $s\in[0,1]^m$, then $A(\mathcal{G},s)=C(\mathcal{G},s)$.*

Proof. By Lemma 4.31, we have that $A(\mathcal{G},s)\leq C(\mathcal{G},s)$, so it remains to show the reverse inequality $C(\mathcal{G},s)\leq A(\mathcal{G},s)$. We do so by induction on $|G|,m\in\{1,2,\ldots\}$. In the base cases with $|G|=1$, by Lemma 4.31, we have that $C(\mathcal{G},s)\leq|G|=1\leq A(\mathcal{G},s)$. In the base cases with $m=1$, by Lemma 4.44, we have that $A(\mathcal{G},s)=C(\mathcal{G},s)$.

Now we suppose $|G|>1$ and $m>1$, and assume the induction hypotheses for all HBL data $\mathcal{G}'$ with $|G'|<|G|$ and $m'<m$, for all $s\in[0,1]^{m'}$. We will invoke the induction hypotheses in three scenarios; after concluding the proof in these cases, we will show how the general case reduces to one of these three.

In the first scenario, some component $s_k=0$: by the induction hypotheses and Lemma 4.37, $A(\mathcal{G},s)=A((\phi_j)_{j\neq k},(s_j)_{j\neq k})=C((\phi_j)_{j\neq k},(s_j)_{j\neq k})=C(\mathcal{G},s)$.

In the second scenario, some component $s_k=1$; let $K=\ker(\phi_k)$, let $\mathcal{G}|_K=(K,(\phi_j(K))_j,(\chi_j)_j)$ be the restricted datum, let $\mathcal{G}'|_K=(K,(\phi_j(K))_{j\neq k},(\chi_j)_{j\neq k})$ be $\mathcal{G}|_K$ with its k-th component omitted, and let V be a set of representatives of the cosets $v+K\in G/K$. For each $w\in G$, define $g^{(w)}$ with components $g_j^{(w)}\colon\phi_j(K)\to[0,\infty]\colon y\mapsto f_j(y+\phi_j(w))$. For each j, $\|g_j^{(w)}\|_{\ell^{1/s_j}(\phi_j(K))}\leq\|f_j\|_{\ell^{1/s_j}(G_j)}$, so there are two mutually exclusive cases. Either $g_j^{(w)}=0$ for some j, or $g^{(w)}\in\mathcal{F}^*(\mathcal{G}|_K,s)$. In the former case, $\sum_{x\in K}\prod_j g_j^{(w)}(\phi_j(x))=0$. In the latter case, $(g_j^{(w)})_{j\neq k}\in\mathcal{F}^*(\mathcal{G}'|_K,(s_j)_{j\neq k})$. For any $f\in\mathcal{F}^*(\mathcal{G},s)$,

$$\begin{aligned}
C(\mathcal{G},s,f)&=\sum_{x\in G}\prod_j\frac{f_j(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}}=\sum_{v\in V}\sum_{x\in K}\prod_j\frac{f_j(\phi_j(v+x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}}=\sum_{\substack{v\in V\\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_K,s)}}\sum_{x\in K}\prod_j\frac{g_j^{(v)}(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}}\\
&=\sum_{\substack{v\in V\\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_K,s)}}\frac{f_k(\phi_k(v))}{\|f_k\|_{\ell^1(G_k)}}\sum_{x\in K}\prod_{j\neq k}\frac{g_j^{(v)}(\phi_j(x))}{\|f_j\|_{\ell^{1/s_j}(G_j)}}\leq\sum_{\substack{v\in V\\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_K,s)}}\frac{f_k(\phi_k(v))}{\|f_k\|_{\ell^1(G_k)}}\sum_{x\in K}\prod_{j\neq k}\frac{g_j^{(v)}(\phi_j(x))}{\|g_j^{(v)}\|_{\ell^{1/s_j}(\phi_j(K))}}\\
&\leq\sum_{\substack{v\in V\\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_K,s)}}\frac{f_k(\phi_k(v))}{\|f_k\|_{\ell^1(G_k)}}C(\mathcal{G}'|_K,(s_j)_{j\neq k}).
\end{aligned}$$

Now by the induction hypothesis, we have that $C(\mathcal{G}'|_K, (s_j)_{j\neq k}) = A(\mathcal{G}'|_K, (s_j)_{j\neq k})$. Moreover, since $\phi_k(K) = \{0\}$, applying Lemma 4.37 with m-functions $f \in \mathcal{F}^*(\mathcal{G}|_K, s)$ such that $f_j = 1_{\phi_j(H)}$ for some $H \in \mathcal{H}_{\mathrm{f}}^*(K)$, thus taking a supremum of the conclusion over all such m-functions, we have that $A(\mathcal{G}'|_K, (s_j)_{j\neq k}) = A(\mathcal{G}|_K, s)$. Lastly, by Lemma 4.41, we have that $A(\mathcal{G}|_K, s) \leq A(\mathcal{G}, s)$. Therefore,

$$C(\mathcal{G}, s, f) \leq A(\mathcal{G}, s) \sum_{\substack{v\in V \\ g^{(v)}\in\mathcal{F}^*(\mathcal{G}|_K, s)}} \frac{f_k(\phi_k(v))}{\|f_k\|_{\ell^1(G_k)}} \leq A(\mathcal{G}, s) \sum_{v\in V} \frac{f_k(\phi_k(v))}{\|f_k\|_{\ell^1(G_k)}} \leq A(\mathcal{G}, s) \frac{\|f_k\|_1}{\|f_k\|_{\ell^1(G_k)}} = A(\mathcal{G}, s),$$

since $\phi_k|_V$ is an injection. The conclusion $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$ follows by taking a supremum over all $f \in \mathcal{F}^*(\mathcal{G}, s)$.

In the third scenario, $A(\mathcal{G}, s, K) \geq 1$ for some $\{0\} < K < G$. This implies the existence of some $\{0\} < H < G$ such that $A(\mathcal{G}, s, H) = A(\mathcal{G}, s)$, as we now show. If $A(\mathcal{G}, s, K) = A(\mathcal{G}, s)$, then we pick $H = K$. Otherwise, $1 \leq A(\mathcal{G}, s, K) < A(\mathcal{G}, s)$. Since $\{0\} < K < G$ and $|G| < \infty$, $\mathcal{H} = \operatorname{argmax}_{0<H<G} A(\mathcal{G}, s, H)$ is nonempty and finite. Each $H \in \mathcal{H}$ maximizes $A(\mathcal{G}|_H, s, F) = A(\mathcal{G}, s, F)$ over all $\{0\} \leq F \leq H$, including $F = \{0\}$, since $A(\mathcal{G}, s, \{0\}) = 1 \leq A(\mathcal{G}, s, K)$. Thus, for any $H \in \mathcal{H}$, $A(\mathcal{G}, s, H) = A(\mathcal{G}, s)$. Considering H obtained in either case, since $A(\mathcal{G}, s, H) = A(\mathcal{G}|_H, s, H) \leq A(\mathcal{G}|_H, s) \leq A(\mathcal{G}, s)$, we have that $A(\mathcal{G}|_H, s, H) = A(\mathcal{G}|_H, s)$, so by Lemma 4.43, the induction hypotheses, and Lemma 4.42,

$$C(\mathcal{G}, s) \leq C(\mathcal{G}|_H, s) C(\mathcal{G}|^H, s) = A(\mathcal{G}|_H, s) A(\mathcal{G}|^H, s) \leq A(\mathcal{G}, s).$$

We now show that it suffices to consider just the preceding three scenarios. For each $a \in [1, \infty)$, define $P(a) = \{t \in [0,1]^m \mid A(\mathcal{G}, t) \leq a\}$. Since $1 \leq A(\mathcal{G}, s) \leq |G|$ by Lemma 4.31 and $|G| < \infty$ by hypothesis, $s \in P(A(\mathcal{G}, s))$, thus $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$ is implied by the more general statement, $C(\mathcal{G}, t) \leq A(\mathcal{G}, s)$ for all $t \in P(A(\mathcal{G}, s))$. More generally still, we will show that for all $a \in [1, \infty)$ and all $t \in P(a)$, $C(\mathcal{G}, t) \leq a$. For each $a \in [1, \infty)$, note that $P(a)$ is a closed and bounded convex set, so by Carathéodory's theorem, it equals its extreme points' convex hull; then, by Lemma 4.40, if $C(\mathcal{G}, t) \leq a$ for all extreme points t of $P(a)$ then it holds for all $t \in P(a)$.

Consider any $a \in [1, \infty)$; if $P(a) \neq \emptyset$, examine any of its extreme points t. Since $t \in P(a)$, $A(\mathcal{G}, t) \leq a$. If any of the preceding three scenarios apply, then we conclude $C(\mathcal{G}, t) \leq A(\mathcal{G}, t) \leq a$. Thus, it remains to treat the scenario where $t \in (0,1)^m$ and $A(\mathcal{G}, t, H) < 1 \leq A(\mathcal{G}, t)$ for all $\{0\} < H < G$. In particular, $A(\mathcal{G}, t) = \max(A(\mathcal{G}, t, \{0\}), A(\mathcal{G}, t, G)) = \max(1, A(\mathcal{G}, t, G))$.

We first show that $A(\mathcal{G}, t) = 1 > A(\mathcal{G}, t, G)$. Suppose towards a contradiction that $A(\mathcal{G}, t) = A(\mathcal{G}, t, G)$. By the definition of $P(a)$, we see that each $s \in P(a)$ satisfies the following finite system of linear inequalities (taking logarithms of $A(\mathcal{G}, s) \leq a$):

$$0 \leq s_j \leq 1 \quad \text{for all } j, \qquad \ln|H| - \sum_j s_j \ln|\phi_j(H)| \leq \ln a \quad \text{for all } \{0\} \neq H \leq G.$$

Under the present hypotheses regarding t, since $A(\mathcal{G}, s)$ is a continuous function of $s \in [0, \infty)^m$, for all $s \in P(a)$ in a neighborhood of t, all of these inequalities hold strictly except possibly the one induced by $H = G$ (just in case $A(\mathcal{G}, t) = a$). Therefore, t is contained in

at most one of the hyperplanes that bound $P(a)$: this contradicts extremality of t, because $P(a)$ is defined by at least $m \geq 2$ and at most finitely many linear inequalities.

Therefore, $A(\mathcal{G}, t, G) < A(\mathcal{G}, t) = \max(1, A(\mathcal{G}, t, G))$. That is, $A(\mathcal{G}, t, G) < 1 = A(\mathcal{G}, t)$. Note that for all $H \leq G$, $A(\mathcal{G}, \alpha t, H)$ is a continuous nonincreasing function of $\alpha \in [0, 1]$, so $A(\mathcal{G}, \alpha t)$ is too. For all $\{0\} < H \leq G$, $A(\mathcal{G}, 0 \cdot t, H) > 1 > A(\mathcal{G}, 1 \cdot t, H)$. Thus, by continuity and (anti-) monotonicity, for all $\{0\} < H \leq G$, there exists $\alpha_H \in (0, 1)$ such that $A(\mathcal{G}, \alpha_H t, H) = 1$. Let $\mathcal{K}$ be the set of all subgroups $\{0\} < K \leq G$ which maximize α_K: since G has finitely many subgroups, $\mathcal{K} \neq \emptyset$ (i.e., this maximum is well defined), and it follows that $A(\mathcal{G}, \alpha_K t) = 1 = A(\mathcal{G}, t)$. Moreover, we can find at least one $K \in \mathcal{K}$ such that $K < G$: to show this, suppose towards a contradiction that none exist. Then, abbreviating $r = \alpha_G t$, we must have that $A(\mathcal{G}, r, G) = A(\mathcal{G}, r) = 1 \leq a$, i.e., $r \in P(a)$. By Lemma 4.32, $P(a)$ also contains every $r \leq q \in [0, 1]^m$. By construction, $r \leq t$, and moreover some component $r_k < t_k$ since $\alpha_G \in (0, 1)$. Fixing $q_j = t_j$ for all $j \neq k$ and $q_k = \theta r_k + (1 - \theta)1$ for some $\theta \in [0, 1]$, we observe that $q \in P(a)$ for any $\theta \in [0, 1]$. Since $t \in (0, 1)^m$ is an extreme point of $P(a)$, $r_k < t_k < 1$, so there exists an open line segment in $P(a)$ containing t, defined by q with parameter $\theta \in (0, 1)$, contradicting extremality of t. In light of this contradiction, we have shown that there exists some $\{0\} < K < G$ such that $A(\mathcal{G}, \alpha_K t, K) = A(\mathcal{G}, \alpha_K t)$. Thus, the third scenario applies (replacing s by $\alpha_K t$), and we conclude

$$C(\mathcal{G}, t) \leq C(\mathcal{G}, \alpha_K t) \leq A(\mathcal{G}, \alpha_K t) = A(\mathcal{G}, t) = 1 \leq a.$$

Repeating this argument and obtaining $C(\mathcal{G}, t) \leq a$ for each extreme point t of $P(a)$ for each $a \in [1, \infty)$, we confirm that $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$. As mentioned above, the conclusion $A(\mathcal{G}, s) = C(\mathcal{G}, s)$ follows from Lemma 4.31. □

4.6 Relating $\mathcal{P}$ with A, B, and C

Now we connect the preceding results in order to complete the proof of Theorem 4.2; Theorem 4.1 follows as a special case. We suppose again that $\mathcal{G}$ is any HBL datum.

Lemma 4.47. *If $s \in \mathcal{P}(\mathcal{G}) \cap [0, 1]^m$ then $A(\mathcal{G}, s) = C(\mathcal{G}, s)$.*

Proof. We first show that $\operatorname{rank}(\bigcap_j \ker(\phi_j)) = 0$. If $\operatorname{rank}(\bigcap_j \ker(\phi_j)) > 0$, then by Lemma 4.11, $\mathcal{P}(\mathcal{G}) = \emptyset$, a contradiction to the hypothesis $s \in \mathcal{P}(\mathcal{G})$.

We first consider the special case where G is finitely generated and torsion-free ($T(G) = \{0\}$), and treat the general case afterward. Applying Lemmas 4.31 and 4.33 when G is torsion-free, we see that $1 \leq A(\mathcal{G}, s) \leq |T(G)| = 1$, so it suffices to show that $C(\mathcal{G}, s) = 1$. We know that $C(\mathcal{G}, s) \geq 1$ by Lemma 4.31, so it remains to show that $C(\mathcal{G}, s) \leq 1$, which we will do by induction on $\operatorname{rank}(G)$.

In the base case $\operatorname{rank}(G) = 0$, since G is finitely generated, $G = \{0\}$, i.e., $|G| = 1$; therefore, $1 \leq C(\mathcal{G}, s) \leq |G|$ by Lemma 4.31.

Now suppose that $\operatorname{rank}(G) > 0$ and assume the induction hypothesis, that the conclusion holds for all HBL data $\mathcal{G}'$ with $\operatorname{rank}(G') < \operatorname{rank}(G)$. By Lemma 4.40, it suffices to show that $C(\mathcal{G}, s) \leq 1$ at the extreme points of $\mathcal{P}(\mathcal{G}) \cap [0, 1]^m$. By Lemma 4.29, if s is an extreme point of $\mathcal{P}(\mathcal{G}) \cap [0, 1]^m$, there are two cases, not necessarily exclusive.

In the first case, $s \in \{0,1\}^m$. Let $I = \{j \mid s_j = 1\}$; note that $I \neq \emptyset$, since otherwise $1 \leq \operatorname{rank}(G) \leq \sum_j s_j \operatorname{rank}(\phi_j(G)) = 0$, contradicting the hypothesis that $s \in \mathcal{P}(\mathcal{G})$. Letting $K = \bigcap_{i \in I} \ker(\phi_i) \in \mathcal{H}^*_{\mathrm{fg}}(G) = \mathcal{H}^*_{\mathbb{N}}(G)$, since $s \in \mathcal{P}(\mathcal{G})$,

$$\operatorname{rank}(K) \leq \sum_j s_j \operatorname{rank}(\phi_j(K)) = \sum_{i \in I} \operatorname{rank}(\phi_i(K)) = 0,$$

so $\operatorname{rank}(K) = 0$, i.e., $K = \{0\}$. By Lemmas 4.37 and 4.36,

$$C(\mathcal{G}, s) = C((\phi_i)_{i \in I}, (s_i)_{i \in I}) \leq |K| = 1.$$

In the second case, there exists $H \in \mathcal{H}^*_{\mathbb{N}}(G) = \mathcal{H}^*_{\mathrm{fg}}(G)$ such that $0 < \operatorname{rank}(H) = \sum_j s_j \operatorname{rank}(\phi_j(H)) < \operatorname{rank}(G)$. By Lemma 4.25, $s \in \mathcal{P}(\mathcal{G}|_H) \cap \mathcal{P}(\mathcal{G}|^H)$. Apply Lemma 4.1 to obtain a torsion-free $F \in \mathcal{H}^*_{\mathrm{fg}}(G)$ such that G/F is finitely generated and torsion-free, $\operatorname{rank}(F) = \operatorname{rank}(H)$, and $\operatorname{rank}(\phi_j(F)) = \operatorname{rank}(\phi_j(H))$ for each j. Since $s \in [0,1]^m$, $0 < \operatorname{rank}(F), \operatorname{rank}(G/F) < \operatorname{rank}(G)$, and both F and G/F are finitely generated and torsion-free, applying Lemma 4.43 and the induction hypotheses for $\mathcal{G}|_F$ and $\mathcal{G}|^F$,

$$C(\mathcal{G}, s) \leq C(\mathcal{G}|_F, s) C(\mathcal{G}|^F, s) \leq 1 \cdot 1 = 1.$$

This concludes the proof in the case where G is finitely generated and torsion free.

Now consider the more general case where G is still finitely generated but not necessarily torsion free; our task is to show that $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$. Since G is finitely generated, $H \in \mathcal{H}^*_{\mathrm{f}}(G)$ if and only if $H \leq T(G)$. Therefore, $A(\mathcal{G}|_{T(G)}, s) = A(\mathcal{G}, s)$ by Lemma 4.41, and then applying Lemma 4.46, $A(\mathcal{G}|_{T(G)}, s) = C(\mathcal{G}|_{T(G)}, s)$. Since $G/T(G)$ is finitely generated and torsion-free, by the argument in the previous case, $A(\mathcal{G}|^{T(G)}, s) = C(\mathcal{G}|^{T(G)}, s) = 1$, and so applying Lemma 4.43,

$$C(\mathcal{G}, s) \leq C(\mathcal{G}|_{T(G)}, s) C(\mathcal{G}|^{T(G)}, s) = A(\mathcal{G}|_{T(G)}, s) \cdot 1 = A(\mathcal{G}, s).$$

This concludes the proof in the case where G is finitely generated.

Now we treat the most general case, where G is not necessarily finitely generated or torsion-free; again, our task is to show that $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$. By Lemma 4.38, it suffices to show that $C(\mathcal{G}, s, f^E) \leq A(\mathcal{G}, s)$ for all $E \in \mathcal{E}^*(G)$ and all $f^E \in \mathcal{F}^*(\mathcal{G}, s)$ where each $\operatorname{supp}(f^E_j) = \phi_j(E)$. Pick any $E \in \mathcal{E}^*(G)$, let $G_E \leq G$ be the subgroup of G generated by E, and examine the restricted datum $\mathcal{G}|_{G_E}$. By Lemma 4.24, $s \in \mathcal{P}(\mathcal{G})$ implies $s \in \mathcal{P}(\mathcal{G}|_{G_E})$, so by applying this result in the finitely generated case (already established), we have that $C(\mathcal{G}|_{G_E}, s) \leq A(\mathcal{G}|_{G_E}, s)$, and moreover $A(\mathcal{G}|_{G_E}, s) \leq A(\mathcal{G}, s)$ by Lemma 4.41. For any $f^E \in \mathcal{F}^*(\mathcal{G}, s)$ where each $\operatorname{supp}(f^E_j) = \phi_j(E)$, it follows that $(f^E_j|_{\phi_j(G_E)})_j \in \mathcal{F}^*(\mathcal{G}|_{G_E}, s)$, therefore,

$$C(\mathcal{G}, s, f^E) = C(\mathcal{G}|_{G_E}, s, (f^E_j|_{\phi_j(G_E)})_j) \leq C(\mathcal{G}|_{G_E}, s) \leq A(\mathcal{G}, s);$$

thus we have $C(\mathcal{G}, s) \leq A(\mathcal{G}, s)$ by taking a supremum over all $E \in \mathcal{E}^*(G)$ and all associated m-functions $f^E \in \mathcal{F}^*(\mathcal{G}, s)$. □

Lemma 4.48. *For each $s \in [0, \infty)^m$, $B(\mathcal{G}, s) < \infty$ implies $s \in \mathcal{P}(\mathcal{G})$.*

Proof. The key step of this proof is the following simple observation regarding free $\mathbb{Z}$-modules. Let H, K be free $\mathbb{Z}$-modules with finite ranks h, k and let $\phi\colon H \to K$ be a surjective $\mathbb{Z}$-module homomorphism (so $k \leq h$). Pick any $\mathbb{Z}$-bases of H and K and let $F \in \mathbb{Z}^{k\times h}$ be the (rank-k) matrix of ϕ with respect to these $\mathbb{Z}$-bases. Consider any element of H, represented as a column vector $x \in \mathbb{Z}^h$ in H's chosen $\mathbb{Z}$-basis, and the column vector $y \in \mathbb{Z}^k$ (representing an element of K) obtained by matrix-vector multiplication $y = F \cdot x$, defined componentwise for each $i \in \{1, \ldots, k\}$ by $y_i = \sum_{j=1}^{h} F_{i,j}x_j$. We have that

$$\max_{i=1}^{k} |y_i| = \max_{i=1}^{k} \left| \sum_{j=1}^{h} F_{i,j}x_j \right| \leq \max_{i=1}^{k} \left(\sum_{j=1}^{h} |F_{i,j}x_j| \right) \leq \left(\max_{i=1}^{k} \sum_{j=1}^{h} |F_{i,j}| \right) \max_{j=1}^{h} |x_j|,$$

Let $F^* = \max_{i=1}^{k} \sum_{j=1}^{h} |F_{i,j}|$; observe that F^* depends only on ϕ and the chosen $\mathbb{Z}$-bases of H and K. For each $N \in \{1, 2, \ldots\}$, let $E_N = \{1, \ldots, N\}^h$ denote a set of N^h column vectors representing elements of H in its chosen $\mathbb{Z}$-basis. For each $x \in E_N$, we have that $\max_{j=1}^{h} |x_j| \leq N$. It follows that $\max_{i=1}^{k} |y_i| \leq F^*N$. Since this holds for all $y \in \phi(E_N)$, we have that $\phi(E_N) \subseteq \{-F^*N, \ldots, F^*N\}^k$, i.e., $|\phi(E_N)| \leq (2F^* + 1)^k N^k$.

Now we apply this observation for each homomorphism ϕ_j in the present setting. Consider any $H \in \mathcal{H}^*_{\mathbb{N}}(G)$, and first consider the case where H is finitely generated and torsion-free. As a torsion-free finitely generated Abelian group, by defining multiplication in terms of addition, H becomes a free $\mathbb{Z}$-module. Letting $r = \operatorname{rank}(H) < \infty$, fix a $\mathbb{Z}$-basis $e_1, \ldots, e_r$ of H and for each $N \in \{1, 2, \ldots\}$, define $E_N = \{\sum_{i=1}^{r} ne_i \mid n \in \{1, \ldots, N\}\}$, so $|E_N| = N^r$.

Now consider each homomorphism ϕ_j in turn. Letting $H_j = \phi_j(H)$, we see that like H, H_j is finitely generated and torsion-free and so can be treated as a free $\mathbb{Z}$-module. Defining $\chi_j\colon H \to H_j\colon x \mapsto \phi_j(x)$ to be a surjective $\mathbb{Z}$-module homomorphism, we can apply the preceding argument: therefore, there exists a constant $A_j \in [0, \infty)$ independent of N such that $|\chi_j(E_N)| \leq A_j N^{r_j}$, where $r_j = \operatorname{rank}(H_j)$. Since the ranks of H, H_j are same when interpreting H, H_j as Abelian groups or as $\mathbb{Z}$-modules, and since $\chi_j(E_N) = \phi_j(E_N)$, we have that $|\phi_j(E_N)| \leq A_j N^{\operatorname{rank}(\phi_j(H))}$. The hypothesis $B(\mathcal{G}, s) < \infty$ implies that $B(\mathcal{G}, s, E_N) < \infty$ for all $N \in \{1, 2, \ldots\}$. In particular, there exists a constant $B \in (0, \infty)$ such that $|E_N| \leq B \prod_j |\phi_j(E_N)|^{s_j}$, so

$$N^{\operatorname{rank}(H)} = |E_N| \leq B \prod_j |\phi_j(E_N)|^{s_j} \leq B \prod_j A_j N^{s_j \operatorname{rank}(\phi_j(H))} = (B \prod_j A_j) N^{\sum_j s_j \operatorname{rank}(\phi_j(H))}.$$

Supposing $N \geq 2$ and taking base-N logarithms, we have that $\operatorname{rank}(H) \leq \log_N(B \prod_j A_j) + \sum_j s_j \operatorname{rank}(\phi_j(H))$. Since B, A_j are independent of N, taking the limit as $N \to \infty$ we have that $\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H))$.

Now consider the case where $H \in \mathcal{H}^*_{\mathbb{N}}(G)$ is neither finitely generated nor torsion-free: pick any $\operatorname{rank}(H)$ independent elements of H and examine the subgroup K of H they generate: K is finitely generated, torsion free, and $\operatorname{rank}(H) = \operatorname{rank}(K)$, so by Lemma 4.3, $\operatorname{rank}(\phi_j(H)) = \operatorname{rank}(\phi_j(K))$ for each j. We can thus apply the argument above to K and obtain the same conclusion for H.

Repeating this argument for all $H \in \mathcal{H}^*_{\mathbb{N}}(G)$, we confirm that $s \in \mathcal{P}(\mathcal{G})$. □

At this point, we are ready to assemble the preceding lemmas to prove Theorem 4.2: all that remains is to handle the cases of $s \in \mathcal{P}(\mathcal{G})$ such that $s \notin [0, 1]^m$.

Proof of Theorem 4.2. If $s \in [0,1]^m$, then we apply Lemma 4.47 to show that $s \in \mathcal{P}(\mathcal{G}) \cap [0,1]^m$ implies $A(\mathcal{G},s) = C(\mathcal{G},s)$; therefore by Lemma 4.31, $B(\mathcal{G},s) = C(\mathcal{G},s) = A(\mathcal{G},s)$. Otherwise, letting $r = (\min(1,s_j))_j$, we apply Lemma 4.39 to show that $B(\mathcal{G},s) = B(\mathcal{G},r)$ and $C(\mathcal{G},s) = C(\mathcal{G},r)$, and apply Lemma 4.5 to show that $r \in \mathcal{P}(\mathcal{G})$; since $r \in [0,1]^m$ as well, we conclude $B(\mathcal{G},s) = C(\mathcal{G},s) = A(\mathcal{G},r)$ by the preceding case. This gives the first conclusion of Theorem 4.2. To show the second conclusion, for any $s \in [0,\infty)^m$, apply Lemma 4.48 to show that $B(\mathcal{G},s) < \infty$ implies $s \in \mathcal{P}(\mathcal{G})$. □

Finally, Theorem 4.1 follows as a special case of the first conclusion of Theorem 4.2.

Proof of Theorem 4.1. In this setting, we have the HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$; since $\mathbb{Z}^r$ is torsion-free, $A(\mathcal{G},s) = 1$ for all $s \in [0,\infty)^m$. Therefore, by Theorem 4.2, if $s \in \mathcal{P}(\mathcal{G})$, then $B(\mathcal{G},s) = 1$. □

4.7 Computing $\mathcal{P}$

Now we address the decidability of the hypothesis of Theorem 4.2, that is, given an HBL datum $\mathcal{G}$ and $s \in [0,\infty)^m$, to decide whether $s \in \mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathbb{N}}(G))$. We will only address the case when $r = \operatorname{rank}(G) < \infty$, in order to exploit Lemma 4.9; we comment briefly on the general case after concluding the proof of Theorem 4.3. Then, in Section 4.7.1, we prove Theorem 4.4.

By Lemma 4.23, since $r < \infty$, it suffices to compute $\mathcal{P}(\mathcal{G}, \mathcal{H}^*_{\mathrm{fg}}(H))$ for any $H \in \mathcal{H}^*_{\mathrm{fg}}(G)$ with $\operatorname{rank}(H) = r$; moreover, there is no loss of generality to suppose H is torsion-free. Therefore, there is no loss of generality hereafter to suppose that G is finitely generated and torsion-free. In particular, there is no loss of generality to suppose $G = \mathbb{Z}^r$; therefore, G's elements (r-tuples over $\mathbb{Z}$) and its action (componentwise addition) are computable.

When convenient, we will treat G and its subgroups as $\mathbb{Z}$-modules and use module-theoretic terminology; multiplication, defined in terms of addition, is also computable. The subgroups of G as $\mathbb{Z}$-modules are finitely generated and torsion-free, thus free (admit $\mathbb{Z}$-bases), and their ranks as $\mathbb{Z}$-modules coincide with their ranks as Abelian groups, so we will not distinguish the two types of rank notationally. For computational reasons, we prefer to work in coordinates: in particular, we assume the standard ordered $\mathbb{Z}$-basis on $\mathbb{Z}^r$, meaning that each element of G, an r-tuple over $\mathbb{Z}$, coincides with its coordinate representation, a column r-vector over $\mathbb{Z}$. Each rank-h subgroup H of G is (non-uniquely) represented by a $\mathbb{Z}$-basis matrix, an r-by-h matrix over $\mathbb{Z}$ that horizontally concatenates the column vectors of some ordered $\mathbb{Z}$-basis of H; we will often use the same symbol H to denote both a subgroup and any of its $\mathbb{Z}$-basis matrices.

We represent the homomorphisms ϕ_j by their kernels $K_j \leq G$, which we assume are provided in $\mathbb{Z}$-basis matrix representation. To see that the codomains G_j are truly irrelevant to the task at hand, rewrite (4.8) in the present notation, $\operatorname{rank}(H) \leq \sum_j s_j \operatorname{rank}(\phi_j(H)) = \sum_j s_j(\operatorname{rank}(H) - \operatorname{rank}(H \cap K_j))$.

We will compute the intersection $H = H_1 \cap H_2$ of any two subgroups by solving the system of $\mathbb{Z}$-linear equations $H_1 x_1 = H_2 x_2$ over $\mathbb{Z}$. In more detail, letting h_1, h_2, h be the ranks of H_1, H_2, H, reduce the matrix $[H_1, -H_2] = TU$ to (column-style) Hermite normal form, where $T \in \mathbb{Z}^{r \times (h_1+h_2)}$ is rank-$(h_1 + h_2 - h)$ and zero outside its first $h_1 + h_2 - h$

columns and $U \in \mathbb{Z}^{(h_1+h_2)\times(h_1+h_2)}$ is unimodular, represented as a sequence of elementary column operations. Compute U^{-1} by applying the reverse sequence of elementary operations to an identity matrix and let $F \in \mathbb{Z}^{(h_1+h_2)\times h}$ be the last h columns of U^{-1}; it follows that $[H_1, -H_2]F = 0_{r\times h}$. Split $F = [F_1^T, F_2^T]^T$ so F_1 is the top h_1 rows and F_2 is the bottom h_2 rows of F, and finally set $H = H_1F_1 = H_2F_2 \in \mathbb{Z}^{r\times h}$.

We now introduce some notation for inequalities of the form $h \leq \sum_j s_j h_j$, defined by parameters $h, h_j \in \mathbb{Z}$, which constrain particular $\mathbb{Z}$-linear combinations of the unknowns $s_j \in \mathbb{R}$. We encode such a constraint as $(h, (h_j)_j) \in \mathbb{Z} \times \mathbb{Z}^m$.

Lemma 4.49. *For any $m \in \{1, 2, \ldots\}$, given a closed, convex subset $\mathcal{P} \subseteq \mathbb{R}^m$ defined by a finite set of constraints on $s \in \mathbb{R}^m$, each of the form $h \leq \sum_j s_j h_j$ and encoded as $(h, (h_j)_j)$, there exists an algorithm that returns the set of ($\mathbb{Q}$-valued) extreme points of the convex polytope $\mathcal{P} \cap [0, 1]^m$.*

Proof. The following is a variation on a standard linear programming problem.

To enforce that $s \in [0, 1]^m$, we augment the given set with the $2m$ inequalities, $(0, e_j)$ and $(-1, -e_j)$ for each j. Now index the inequalities by $\alpha \in A$, yielding the nonempty, finite set $\{(r_\alpha, (r_{\alpha,j})_j) \mid \alpha \in A\}$.

Any $t \in \mathbb{R}^m$ is an extreme point of $\mathcal{P} \cap [0, 1]^m$ if and only if (1) $r_\alpha \leq \sum_j t_j r_{\alpha,j}$ for all $\alpha \in A$ and (2) there exists $B \subseteq A$ of size $|B| = m$ such that the m-tuples $(r_{\beta,j})_j$, $\beta \in B$, are $\mathbb{Z}$-linearly independent and $r_\beta = \sum_j t_j r_{\beta,j}$ for all $\beta \in B$. (Condition (1) enforces that $t \in \mathcal{P}$ and condition (2) enforces that t lies on m $\mathbb{Z}$-linearly independent bounding hyperplanes.) It follows that for each extreme point t of $\mathcal{P} \cap [0, 1]^m$, each component $t_j \in \mathbb{Q}$.

Identify the maximal subset $\mathcal{B}$ of the finite set $\{X \subset A \mid |X| = m\}$ such that for each $B \in \mathcal{B}$, the m-tuples $(r_{\beta,j})_j$, $\beta \in B$, are $\mathbb{Z}$-linearly independent. We test $\mathbb{Z}$-linear independence by defining $R \in \mathbb{Z}^{m\times m}$ with entries $R_{j,\beta} = r_{\beta,j}$, reducing $R = UT$ to (row-style) Hermite normal form where $U \in \mathbb{Z}^{m\times m}$ is unimodular and the number of leading nonzero rows of $T \in \mathbb{Z}^{m\times m}$ equals m if and only if $(r_{\beta,j})_j$ are $\mathbb{Z}$-linearly independent. For each $B \in \mathcal{B}$, the system of m $\mathbb{Z}$-linear equations $r_\beta = \sum_j t_{B,j} r_{\beta,j}$, $\beta \in B$ in the m unknowns $(t_{B,j})_j$ thus has a unique solution over $\mathbb{Q}$, which we compute by inverting R over $\mathbb{Q}$ using Gauss-Jordan elimination and applying R^{-1} to the column vector $(r_\beta)_\beta$. We then check whether, for all $\alpha \in A \setminus B$, $r_\alpha \leq \sum_j t_{B,j} r_{\alpha,j}$, by further rational arithmetic; if so, augment the set of extreme points, initially $\emptyset$, with $t = (t_{B,j})_j$. □

Another task that arises is enumerating the subgroups of $G = \mathbb{Z}^r$.

Lemma 4.50. *Given $r, n \in \mathbb{N}$, there exists an algorithm that outputs the first n elements of a fixed enumeration (independent of n, repetitions allowed) of the subgroups of $\mathbb{Z}^r$, represented as basis matrices.*

Proof. If $n = 0$, terminate with the empty tuple (). Otherwise, if $r = 0$, terminate with the singleton tuple ($\{0\}$), where $\{0\}$ is represented as the empty matrix []. Otherwise, consider any (fixed) computable surjective function $A\colon \{1, 2, \ldots\} \to \mathbb{Z}^{r\times r}$: at least one such function (algorithm) exists since $\mathbb{Z}^{r\times r}$ is countable. Consider any $i \in \{1, 2, \ldots\}$, not necessarily $i \leq n$. The matrix A_i is rank-r_i and its columns, interpreted as elements of G, span (over $\mathbb{Z}$) a rank-r_i subgroup $H_i \leq G$. Moreover, H_i is a surjective function from $i \in \{1, 2, \ldots\}$ to

the subgroups $H_i \leq G$, i.e., every subgroup of G is represented in the enumeration $(A_i)_i$. Returning $(H_1, \ldots, H_n)$ thus satisfies the conclusion.

It remains to show how to reduce each A_i to a $\mathbb{Z}$-basis matrix of the associated subgroup H_i: reduce each $A_i = TU$ to (column-style) Hermite normal form where $T \in \mathbb{Z}^{r \times r}$ is rank-r_i and zero past its first r_i columns and $U \in \mathbb{Z}^{r \times r}$ is unimodular, and let be the $H_i \in \mathbb{Z}^{r \times r_i}$ be the first r_i columns of T (the nonzero ones). □

Note that a subgroup can appear more than once in the enumeration. Also note that this algorithm is generally non-halting, outputting subgroups indefinitely. However, Lemma 4.9 tells us that after some finite number of steps, the output will contain some (finite) subset $\mathcal{H}$ of subgroups such that $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H})$. Thus, our strategy is to enumerate subgroups one-by-one, pausing after each step i to check whether the output, a finite set of subgroups $\mathcal{H}_i = \{H_1, \ldots, H_i\}$, contains such an $\mathcal{H}$.

With this machinery at hand, we now show that $\mathcal{P}(\mathcal{G})$ is computable under the present assumption that $G = \mathbb{Z}^r$ for some $r < \infty$; recall that $\mathcal{H}^*(G)$ denotes the set of all subgroups of G, which here are all finitely generated, torsion free, and encoded as $\mathbb{Z}$-basis matrices. A similar algorithm to the following one was sketched by Valdimarsson following the proof of [35, Theorem 1.8], in the context of computing the hypotheses of [9, Theorem 2.1], which define convex set like $\mathcal{P}(\mathcal{G})$. Valdimarsson's approach avoids enumerating all subgroups (technically, subspaces), and instead only enumerates the lattice generated by $(K_j)_j$; our future work will show that it suffices to search such a lattice in the present setting as well.

Lemma 4.51. *There exists an algorithm that returns a finite $\mathcal{H} \subseteq \mathcal{H}^*(G)$ such that $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H})$.*

Proof. By Lemma 4.8, if $\operatorname{rank}(G) = 0$, then $\mathcal{P}(\mathcal{G}) = [0, \infty)^m$, so we return $\mathcal{H} = \emptyset$. Otherwise, $\operatorname{rank}(G) \geq 1$. By Lemma 4.11, letting $K = \bigcap_j K_j$, computed via pairwise intersections, if $\operatorname{rank}(K) > 0$ then $\mathcal{P}(\mathcal{G}) = \emptyset$, thus we terminate with $\mathcal{H} = \{K\} \subset \mathcal{H}^*(G)$. Otherwise, by Lemma 4.10, $\operatorname{rank}(K) = 0$ and $(1, \ldots, 1) \in \mathcal{P}(\mathcal{G})$. We will now construct a finite $\mathcal{H} \subseteq \mathcal{H}^*(G)$ such that $\mathcal{P}(\mathcal{G}) \cap [0,1]^m = \mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1]^m$: the conclusion $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H})$ follows from Lemma 4.5.

Consider the base case $m = 1$: Since $\operatorname{rank}(K_1) = 0$, for all $H \in \mathcal{H}^*(G)$, $\operatorname{rank}(H) = \operatorname{rank}(\phi_1(H))$, so taking $\mathcal{H} = \{H\}$ for any $H \in \mathcal{H}^*(G)$ with $\operatorname{rank}(H) > 0$, $\mathcal{P}(\mathcal{G}, \mathcal{H}) \cap [0,1] = \mathcal{P}(\mathcal{G}) \cap [0,1] = \{1\}$. (We can also confirm this with Lemma 4.12: $s_1 = \sum_j s_j \geq 1$.)

Now consider the inductive case $m \geq 2$ and $\operatorname{rank}(G) \geq 1$, meaning that we suppose the conclusion holds for all $\mathcal{G}'$ such that $m' < m$ or $\operatorname{rank}(G') < \operatorname{rank}(G)$. We run the following subroutine, which recursively invokes this lemma. Run the algorithm in Lemma 4.50 step-by-step, pausing after each step $i \in \{1, 2, \ldots\}$ to examine the finite $\mathcal{H}_i \subseteq \mathcal{H}^*(G)$ computed thus far. We have that

$$\emptyset \neq \mathcal{H}_1 \subseteq \cdots \subseteq \mathcal{H}_i \subseteq \cdots \subseteq \mathcal{H}^*(G),$$

so by Lemma 4.6,

$$[0, \infty)^m \supseteq \mathcal{P}(\mathcal{G}, \mathcal{H}_1) \supseteq \cdots \supseteq \mathcal{P}(\mathcal{G}, \mathcal{H}_i) \supseteq \cdots \supseteq \mathcal{P}(\mathcal{G}).$$

Moreover, by Lemma 4.9, there exists a $l \in \{1, 2, \ldots\}$ such that $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) = \mathcal{P}(\mathcal{G})$ for all $i \geq l$, so we only need to take finitely many steps i (however, we don't yet have bounds

on how big i has to be). Determine the set $\{(\mathrm{rank}(H), (\mathrm{rank}(\phi_j(H)))_j) \mid H \in \mathcal{H}_i\}$ of linear inequalities that $\mathcal{H}_i$ generates: note that each subgroup's rank is stored as part of the $\mathbb{Z}$-basis matrix representation, and we can compute $\mathrm{rank}(\phi_j(H)) = \mathrm{rank}(H) - \mathrm{rank}(H \cap K_j)$ from H and K_j using the algorithm for computing subgroup intersections given earlier. We then apply Lemma 4.49 to obtain the finite set of extreme points of $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) \cap [0,1]^m$, each of which is $\mathbb{Q}$-valued.

We will analyze the finitely many extreme points of $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) \cap [0,1]^m$ in turn. Each extreme point s of $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) \cap [0,1]^m$ falls into one of three cases, and depending on the outcome, the algorithm either terminates or moves on to the next extreme point.

In the first case, $s \in (0,1)^m$ — note that we can decide inequality over $\mathbb{Q}$, and testing this particular case involves $2m$ inequality decisions. We can adapt the algorithm in Lemma 4.49 to exploit Lemma 4.17 and return $H \in \mathcal{H}_i$ such that $0 < \mathrm{rank}(H) = \sum_j s_j \mathrm{rank}(\phi_j(H)) < \mathrm{rank}(G)$.

A complication now arises: even though G and H are torsion-free, the quotient group G/H need not be, and thus may not have a well-defined basis-matrix representation. As in the proof of Theorem 4.2, we apply Lemma 4.1 to avoid torsion during the factorization step. We now give an algorithm that implements Lemma 4.1, i.e., computing a subgroup $H \leq K \leq G$ such that G/K is torsion-free and $\mathrm{rank}(H) = \mathrm{rank}(K)$. Additionally, since K has a complement in G, we will represent G/K by this subgroup. In case $h = \mathrm{rank}(H) = 0$, it follows that $H = \{0\}$, thus we simply return $K = H$, and represent G/K by G, i.e., the r-by-r identity matrix. In case $h = r = \mathrm{rank}(G)$, we simply return $K = G$ and represent G/K by $\{0\}$, i.e., the empty matrix []. Otherwise $0 < h < r$. Given the r-by-h matrix H, reduce H to (row-style) Hermite normal form $H = UT$ where $U \in \mathbb{Z}^{r \times r}$ is unimodular and $T \in \mathbb{Z}^{r \times h}$ is full rank and zero below its top h rows. Represent K by the leading r-by-h submatrix of U and represent G/K by the trailing r-by-$(r-h)$ submatrix of U.

It follows from Lemma 4.3 that $\mathrm{rank}(\phi_j(H)) = \mathrm{rank}(\phi_j(K))$ for each j as well, so combining Lemmas 4.24 and 4.25, $s \in \mathcal{P}(\mathcal{G})$ if and only if $s \in \mathcal{P}(\mathcal{G}|_K) \cap \mathcal{P}(\mathcal{G}|^K)$.

We now show how to replace $\mathcal{G}$ by $\mathcal{G}|_K$ and $\mathcal{G}|^K$; let $r = \mathrm{rank}(G)$ and $k = \mathrm{rank}(K)$. For each j, compute $K'_j = K \cap K_j$ and $K''_j = G/K \cap K_j$ (recalling that G/K is represented as a complement of K in G): K'_j and K''_j represent $\ker(\chi_j)$ and $\ker(\psi_j)$, respectively, as subgroups of G; however, to apply the induction hypotheses, we need $\ker(\chi_j)$ to be represented as a subgroup of K, and we need $\ker(\psi_j)$ to be represented as a subgroup of G/K. We will change coordinates according the unimodular matrix U obtained above: letting V_1 and V_2 denote the top k and bottom $r-k$ rows of U^{-1}, computed by applying the reverse sequence of elementary operations to an identity matrix, we represent each $\ker(\chi_j)$ by $V_1 K'_j$ and each $\ker(\psi_j)$ by $V_2 K''_j$.

Since $\mathrm{rank}(K), \mathrm{rank}(G/K) < \mathrm{rank}(G)$, we can apply the induction hypotheses (i.e., apply the algorithm recursively) to compute finite sets $\mathcal{H}' \subseteq \mathcal{H}^*(\mathcal{G}|_K)$ and $\mathcal{H}'' \subseteq \mathcal{H}^*(\mathcal{G}|^K)$ such that $\mathcal{P}(\mathcal{G}|_K) \cap [0,1]^m = \mathcal{P}(\mathcal{G}|_K, \mathcal{H}') \cap [0,1]^m$ and $\mathcal{P}(\mathcal{G}|^K) \cap [0,1]^m = \mathcal{P}(\mathcal{G}|^K, \mathcal{H}'') \cap [0,1]^m$. We can then test whether $s \in \mathcal{P}(\mathcal{G})$ by testing whether, for all $K' \in \mathcal{H}'$ and $K'' \in \mathcal{H}''$, $\mathrm{rank}(K') \leq \sum_j s_j \mathrm{rank}(\chi_j(K'))$ and $\mathrm{rank}(K'') \leq \sum_j s_j \mathrm{rank}(\psi_j(K''))$, using rational arithmetic. If any of these tests fail, then we proceed with taking the next step $i+1$ of the algorithm in Lemma 4.50, and repeat the preceding steps with respect to $\mathcal{H}_{i+1}$.

In the second case, some component $s_k = 1$; we can reduce $\mathcal{G}$ to the HBL datum $\mathcal{G}' = (K_k, (G_j)_{j \neq k}, (\phi_j|_{K_k})_{j \neq k})$ by dropping the k-th homomorphism and, for each $j \neq k$, replacing

K_j by $V_1 K_j$, where V_1 is computed from K_k as it was from the matrix K in the preceding case. By Lemma 4.26, we can apply this algorithm recursively to $\mathcal{G}'$, which has one less homomorphism, obtaining a finite set $\mathcal{H}' \subseteq \mathcal{H}^*(G)$ such that $(s_j)_{j\neq k} \in \mathcal{P}(\mathcal{G}', \mathcal{H}')$ if and only if $s \in \mathcal{P}(\mathcal{G})$. To check whether $s \in \mathcal{P}(\mathcal{G})$, we test whether $\operatorname{rank}(H) \leq \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(H))$ for all $H \in \mathcal{H}'$. If any of these tests fail, then we continue with the next step $i+1$ of the algorithm in Lemma 4.50, and repeat the preceding steps with respect to $\mathcal{H}_{i+1}$.

In the third case, some component $s_k = 0$; by Lemma 4.13, we can apply this algorithm recursively to the HBL datum $\mathcal{G}' = (G, (G_j)_{j\neq k}, (\phi_j)_{j\neq k})$, which has one less homomorphism (i.e., we omit K_k), obtaining a finite set $\mathcal{H}' \subseteq \mathcal{H}^*_{\mathrm{fg}}(\mathcal{G})$ such that $(s_j)_{j\neq k} \in \mathcal{P}(\mathcal{G}', \mathcal{H}')$ if and only if $s \in \mathcal{P}(\mathcal{G})$. To check whether $s \in \mathcal{P}(\mathcal{G})$, we test whether $\operatorname{rank}(H) \leq \sum_{j\neq k} s_j \operatorname{rank}(\phi_j(H))$ for all $H \in \mathcal{H}'$, using rational arithmetic. If any of these tests fail, then continue with the next step $i+1$ of the algorithm in Lemma 4.50, and repeat the preceding steps with respect to $\mathcal{H}_{i+1}$.

If every extreme point $s \in \mathcal{P}(\mathcal{G}, \mathcal{H}_i) \cap [0,1]^m$ is contained in $\mathcal{P}(\mathcal{G})$, which we have already confirmed must happen within finitely many steps, then because any closed and bounded convex subset of $\mathbb{R}^m$ equals the convex hull of its extreme points, $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) \cap [0,1]^m \subseteq \mathcal{P}(\mathcal{G})$. We already concluded the converse inclusion $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) \supseteq \mathcal{P}(\mathcal{G})$ by Lemma 4.6, thus $\mathcal{P}(\mathcal{G}, \mathcal{H}_i) = \mathcal{P}(\mathcal{G})$ and we return the finite set $\mathcal{H} = \mathcal{H}_i$. □

Note that the output of the algorithm in Lemma 4.51 can be then used to decide membership in $\mathcal{P}(\mathcal{G})$ via checking each of a finite list of $\mathbb{Z}$-linear inequalities (this task is already performed within the algorithm, in the inductive step). Since $\mathcal{P}(\mathcal{G})$ is a subset of $\mathbb{R}^m$, cocountably many points of which are uncomputable, membership can only be decided for the computable elements of $\mathbb{R}^m$.

Lemma 4.51 nearly completes the proof of Theorem 4.3, except for one caveat that was glossed over in the theorem statement.

Proof of Theorem 4.3. Lemma 4.51 gives the the desired conclusion in the case $G = \mathbb{Z}^r$, assuming that $\mathcal{G}$ is represented as an m-tuple $(K_j)_j$ of $\mathbb{Z}$-basis matrices, as defined above. We showed earlier that when G is a finite rank Abelian group, the membership problem reduces to the study of the case $G = \mathbb{Z}^r$. Whether or not this reduction itself is computable depends on the given representation of $\mathcal{G}$: we have sidestepped this potential obstacle by assuming a particular representation. □

We briefly comment on the most general case, where $\operatorname{rank}(G) = \infty$. An immediate obstacle facing our approach is that $\mathcal{H}^*_{\mathbb{N}}(G)$ need not be countable, so even if there exists a finite $\mathcal{H} \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$ such that $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H})$, it may not appear after finitely many steps of the algorithm in Lemma 4.50. However, since $\mathcal{P}(\mathcal{G})$ is defined by a countable (not necessarily finite) intersection of half-spaces of $\mathbb{R}^m$, there exists a countable analogue of Lemma 4.9. Therefore, supposing first that we can resolve the issues of representing $\mathcal{G}$ and computing G's elements and action, given an algorithm (like the one in Lemma 4.50) that enumerates a countable $\mathcal{H} \subseteq \mathcal{H}^*_{\mathbb{N}}(G)$ such that $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H})$, we can extend the algorithm in Lemma 4.51 into a nondeterministic one that halts whenever $s \notin \mathcal{P}(\mathcal{G})$, and otherwise potentially runs forever. In other words, membership in $\mathcal{P}(\mathcal{G})$ may still be cosemidecidable in the case $\operatorname{rank}(G) = \infty$. These comments are motivated by Valdimarsson's result [35, Theorem 1.8] regarding the hypotheses of [9, Theorem 2.1], demonstrating in the continuum

setting, where G is a finite-dimensional $\mathbb{R}$-vector space, that it suffices to search a countable set of subspaces — the lattice generated by $(K_j)_j$ — despite G having (as an Abelian group) both infinite rank and uncountably many finitely generated subgroups.

4.7.1 Computing the Inequalities Defining $\mathcal{P}(\mathcal{G})$

Our last investigation regarding computability of $\mathcal{P}(\mathcal{G})$ concerns the question of whether a simpler algorithm would suffice. In particular, in the case $G = \mathbb{Z}^r$ for $r < \infty$, given arbitrary $h, h_j \in \{0, \ldots, r\}$, can we decide whether there exists $H \leq G$ such that $\operatorname{rank}(H) = h$ and each $\operatorname{rank}(\phi_j(H)) = h_j$? (Actually computing H is not required.) An algorithm that decides this problem could be used to iterate over the $(r+1)^{m+1}$ possible values of h, h_j, outputting a $\mathbb{Z}$-linear inequality $h \leq \sum_j s_j h_j$ over $s \in \mathbb{R}^m$, encoded as $(h, (h_j)_j)$, whenever the answer is affirmative. It then remains to test whether the (computable) m-tuple $s \in \mathbb{R}^m$, satisfies a finite set of inequalities, which we have already shown is decidable. However, we have so far failed to devise such an algorithm: we will show that this question of decidability is equivalent to Hilbert's Tenth Problem for $\mathbb{Q}$, a longstanding open problem.

For the following discussion, we will change our notation yet again. In particular, we now suppose that the homomorphisms ϕ_j are endomorphisms of $G = \mathbb{Z}^r$: we can always satisfy this assumption by identifying each homomorphic image $\phi_j(G)$ with an isomorphic subgroup of G. We now represent each ϕ_j as an r-by-r matrix over $\mathbb{Z}$ with respect to the standard ordered $\mathbb{Z}$-basis on $\mathbb{Z}^r$ (rather than by its kernel K_j), and use the symbol ϕ_j to refer to both the endomorphism and its matrix. We also introduce the notation $\phi = (\phi_j)_j \in (\mathbb{Z}^{r \times r})^m$ to denote an m-tuple over $\mathbb{Z}^{r \times r}$.

In the following definitions, we will replace $\mathbb{Z}$ by a more general integral domain, denoted $\mathbb{D}$; we are primarily interested in the integral domains $\mathbb{Z}$ and $\mathbb{Q}$. For any integral domain $\mathbb{D}$, $m, r \in \{1, 2, \ldots\}$, and $h, h_j \in \mathbb{N}$, define (using the notations $\leq_{\mathbb{D}}$ and $\operatorname{rank}_{\mathbb{D}}$ introduced in Section 4.4.2),

$$E^{\mathbb{D},m,r}_{h,(h_j)_j} = \left\{ \phi \in (\mathbb{D}^{r \times r})^m \;\middle|\; (\exists H \leq_{\mathbb{D}} \mathbb{D}^r) \operatorname{rank}_{\mathbb{D}}(H) = h \wedge (\forall j) \operatorname{rank}_{\mathbb{D}}(\phi_j(H)) = h_j \right\};$$

in our application, $\mathbb{D} = \mathbb{Z}$, m, $r = \operatorname{rank}(G)$, and $\phi = (\phi_j)_j$ are specified by $\mathcal{G}$, while h, h_j are specified by the particular inequality $(h, (h_j)_j)$ whose existence in $\mathcal{P}(\mathcal{G})$ we are currently probing; thus, our problem is deciding whether $\phi \in E^{\mathbb{D},m,r}_{h,(h_j)_j}$. This problem reduces to one about solutions of polynomial equations, not over $\mathbb{D}$, but over $\mathbb{D}$'s field of fractions. We demonstrate this below in Lemma 4.53 in the case $\mathbb{D} = \mathbb{Z}$, with field of fractions $\mathbb{Q}$.

First we pause to introduce some notation for polynomials, as well as introduce Hilbert's Tenth Problem. For any integral domain $\mathbb{D}$ and $t \in \mathbb{N}$, let $\mathbb{D}[x_1, \ldots, x_t]$ denote the ring of polynomials over $\mathbb{D}$ in the variables $x_1, \ldots, x_t$. For any $S \subseteq \mathbb{D}[x_1, \ldots, x_t]$, define

$$Z(S) = \{a \in \mathbb{D}^t \mid (\forall f \in S)\, f(a) = 0\},$$

with $Z(f)$ being shorthand for $Z(\{f\})$ when $f \in \mathbb{D}[x_1, \ldots, x_t]$. Hilbert's Tenth Problem for $\mathbb{D}$ is the question of whether $Z(S) \neq 0$ is decidable for all $t \in \mathbb{N}$ and all finite $S \subseteq \mathbb{D}[x_1, \ldots, x_t]$. Matiyasevich-Davis-Putnam-Robinson's theorem (see, e.g., [27]) famously resolved Hilbert's Tenth Problem (for $\mathbb{Z}$, the version Hilbert posed) with a negative answer: there exists no general algorithm for testing solvability over $\mathbb{Z}$. Hilbert's Tenth Problem for the computable

real numbers, however, was resolved with an affirmative answer by Tarski [32]; the asymptotic complexity of Tarski's original algorithm has been improved many times, with Collins' algorithm [17], called cylindrical algebraic decomposition, being a milestone. Hilbert's Tenth Problem for $\mathbb{Q}$, on the other hand, remains open (see, e.g., [28]).

We already saw in Section 4.4.2 that $\mathcal{P}(\mathcal{G}) = \mathcal{P}_{\mathbb{Q}}(\mathcal{G}^{\mathbb{Q}})$, where $\mathcal{G}^{\mathbb{Q}} = (G^{\mathbb{Q}}, (G_j^{\mathbb{Q}})_j, (\phi_j^{\mathbb{Q}})_j)$ is obtained from $\mathcal{G}$ by extending scalars from $\mathbb{Z}$ to $\mathbb{Q}$, and $\mathcal{P}_{\mathbb{Q}}$ is obtained from $\mathcal{P}$ by replacing Abelian group ranks with $\mathbb{Q}$-module ranks in its definition, (4.8).

Lemma 4.52. $E^{\mathbb{Z},m,r}_{h,(h_j)_j} = E^{\mathbb{Q},m,r}_{h,(h_j)_j} \cap (\mathbb{Z}^{r\times r})^m$.

Proof. This follows from specializing Lemma 4.4 to the inclusion $\mathbb{Z} \subset \mathbb{Q}$. □

Lemma 4.53. *Each instance of the problem of whether $\phi \in E^{\mathbb{Q},m,r}_{h,(h_j)_j}$, given ϕ, m, r, h, h_j, computably reduces to an instance of the problem of whether $Z(f) \neq \emptyset$ for some $f \in \mathbb{Q}[x_1, \ldots, x_t]$ and $t \in \mathbb{N}$.*

Proof. Recall that if M is a matrix over $\mathbb{Q}$, $\mathrm{rank}_{\mathbb{Q}}(M) = s$ if and only if all $(s+1)$-by-$(s+1)$ minors of M are zero and at least one s-by-s minor is nonzero. Since for any $x, y \in \mathbb{Q}$, $x^2 + y^2 = 0$ if and only if $x = y = 0$, we can reduce any finite set of polynomial equations to a single equation. In particular, the condition that all $(s+1)$-by-$(s+1)$ minors of M are zero reduces to an equality of the form $\sum_f f^2 = 0$, and the condition that not all s-by-s minors are zero reduces to an inequality of the form $\sum_g g^2 \neq 0$. The latter expression can be converted to an equality, since for any $g \in \mathbb{Q}[X]$, $g \neq 0$ is equivalent to $f = gy - 1 = 0$ where $y \notin X$ and $f \in \mathbb{Q}[X \cup \{y\}]$. Now the remaining two polynomial equations can be combined into a single equation.

Let B be a r-by-h matrix of variables and let $\phi = (\phi_j)_j \in (\mathbb{Q}^{r\times r})^m$ as given. Now using the preceding argument, define $m+1$ polynomial equations for the matrices $M = B$ (with $s = h$) and $M = \phi_j B$ (with $s = h_j$) for each j, and then finally combine them into a single equation. □

If Hilbert's Tenth Problem for $\mathbb{Q}$ has an affirmative solution, then we can use the associated algorithm to directly test each candidate inequality $(h, (h_j))$.

Now we develop a stronger converse result than we need to show equivalence with Hilbert's Tenth Problem for $\mathbb{Q}$, that the ability to decide membership in $E^{\mathbb{Q},m,r}_{h,(h_j)_j}$ enables deciding membership in arbitrary Diophantine sets (over $\mathbb{Q}$). Given $p, q \in \mathbb{N}$ and $S \subseteq \mathbb{Q}[x_1, \ldots, x_p; y_1, \ldots, y_q]$, the parameter triple (p, q, S) define the Diophantine set (over $\mathbb{Q}$),

$$D(p, q, S) = \{a \in \mathbb{Q}^p \mid (\exists b \in \mathbb{Q}^q)(a, b) \in Z(S)\}.$$

Our task above, deciding whether $Z(S) \neq \emptyset$ for some $S \subseteq \mathbb{Q}[x_1, \ldots, x_s]$ and $s \in \mathbb{N}$, is equivalent to deciding whether the empty tuple $() \in D(0, s, S) \subseteq \mathbb{Q}^0 = \{()\}$; a different task, when additionally given some $a \in \mathbb{Q}^s$, is deciding whether $a \in Z(S)$, which is equivalent to deciding whether $a \in D(s, 0, S) \subseteq \mathbb{Q}^s$.

At this point, we observe that $\mathbb{Q}$ is a Noetherian ring, so by Hilbert's basis theorem, for any $S \subseteq \mathbb{Q}[x_1, \ldots, x_s]$, there exists a finite $R \subseteq \mathbb{Q}[x_1, \ldots, x_s]$ such that $Z(R) = Z(S)$. In particular, for any Diophantine set $D(p, q, S)$, where $p, q \in \mathbb{N}$ such that $s = p + q$, there is no loss of generality to suppose S is finite. From here on, we will suppose that all sets of polynomials are finite.

Lemma 4.54. *Each instance of the problem of whether $a \in D(p,q,S)$, given a,p,q,S, computably reduces to the problem of $\phi \in E^{\mathbb{Q},m,r}_{h,(h_j)_j}$, for some ϕ, m, r, h, h_j.*

Proof. Our first step is to simplify the presentation of S by (algorithmically) constructing T, a set of simpler polynomials in a larger set of variables, such that $Z(S)$ equals $Z(T)$ on a subset of T's variables, thus computably reducing the membership problem.

Let $s = p+q$ and substitute the variables $(z_1, \ldots, z_s)$ for $(x_1, \ldots, x_p; y_1, \ldots, y_q)$, rewriting S as a (finite) subset of $\mathbb{Q}[z_1, \ldots, z_s]$; we will dedicate the subscript i to range over $\{1, \ldots, s\}$. Compute $d = \max_{f \in S} \max_i \deg_{z_i}(f)$ and define the index set $A = \{0, \ldots, d\}^s$ and $(d+1)^s$ variables v_α indexed by $\alpha = (\alpha_i)_i \in A$. Construct a finite $T_1 \subseteq \mathbb{Q}[\{v_\alpha\}_{\alpha \in A}]$, initially $T_1 = \emptyset$, by examining each $f \in S$, which can be represented as $f = \sum_{\alpha \in A} c_\alpha \prod_i z_i^{\alpha_i}$ where each $c_\alpha \in \mathbb{Q}$ is uniquely defined by f. For each $f \in S$, augment T_1 with the polynomial $\sum_{\alpha \in A} c_\alpha v_\alpha$. Also construct a finite $T_2 \subseteq \mathbb{Q}[\{v_\alpha\}_{\alpha \in A}]$, initially $T_2 = \emptyset$, by examining each $\alpha \in A$: if $\alpha = (0, \ldots, 0)$, augment T_2 with the polynomial $v_{(0,\ldots,0)} - 1$; otherwise, keeping α fixed, examine each $\beta \in A$: if $\beta \neq (0, \ldots, 0)$ and $\alpha + \beta \in A$, augment T_2 with the polynomial $v_{\alpha+\beta} - v_\alpha v_\beta$ (if not already in T_2). Now let $T = T_1 \cup T_2$.

We now show that $Z(T)$ equals $Z(S)$ when projecting the former set onto to its coordinates corresponding to the s variables v_{e_i}, where each $e_i \in A$ is zero except for its unit i-th entry; this projection, $\pi \colon \mathbb{Q}^A \to \mathbb{Q}^s \colon (v_\alpha)_{\alpha \in A} \mapsto (v_{e_i})_i$, is computable. Now define $\chi \colon \mathbb{Q}^s \to \mathbb{Q}^A \colon (z_i)_i \mapsto (\prod_i z_i^{\alpha_i})_{\alpha \in A}$, also computable. By construction, $\chi(\mathbb{Q}^s) = Z(T_2) \subseteq \mathbb{Q}^A$; defining ψ to be χ with its codomain restricted to $Z(T_2)$, we see that ψ is a bijection with $\psi^{-1} = \pi|_{Z(T_2)}$. Moreover, for any $a \in \mathbb{Q}^s$, $a \in Z(S)$ if and only if $\chi(a) \in Z(T)$. Therefore, $Z(S) = \pi(Z(T))$.

Thus, given $a \in \mathbb{Q}^p$, deciding whether $a \in D(p,q,S)$ is equivalent to deciding whether $a \in D(p, (d+1)^{p+q} - p, T)$, linearizing the indices $\alpha \in A$ so that $T \subseteq \mathbb{Q}[v_{e_1}, \ldots, v_{e_p}; \ldots]$, where the second ellipsis indicates the variables' order is arbitrary after the first p. Moreover, each instance of the former problem (with S) computably reduces to an instance of the latter problem (with T).

In light of this reduction, we suppose that the given S has the form of T. That is, $S \subseteq \mathbb{Q}[z_1, \ldots, z_s]$ with $s = p + q$, and there exist $k, l \in \mathbb{N}$ such that $|S| = k + l$ and the elements of S can be expressed as

$$S = \{z_{i_{1,j}} z_{i_{2,j}} - z_{i_{3,j}}\}_{j=1}^k \cup \{\lambda_{0,j} + \sum_i \lambda_{i,j} z_i\}_{j=1}^l,$$

where the subscript i continues to range over $\{1, \ldots, s\}$, and the double subscripts $i_{1,j}, i_{2,j}, i_{3,j} \in \{1, \ldots, s\}$ are distinct for each j but not necessarily between different j. We suppose that each $\lambda_{0,j}, \lambda_{i,j} \in \mathbb{Z}$: this can be computably enforced, scaling by integers to clear denominators, without affecting $Z(T)$.

Now define $m = 4 + s + k + p + l$, $r = 2s + 2$, $h = 2$, and $h_j = 1$ for $j \leq 4 + s + k$ and $h_j = 0$ otherwise (j continues to range over $\{1, \ldots, m\}$ unless specified otherwise). We will use the coordinates $(u; v) = (u_0, \ldots, u_s; v_0, \ldots, v_s)$ on $\mathbb{Q}^r$. Instead of working with a fixed m-tuple $(\phi_j)_j \in (\mathbb{Q}^{r \times r})^m$, we will parameterize $(\phi_j^{(a)})_j$ by the input $a \in \mathbb{Q}^p$ to the Diophantine membership problem, where we assume that $a = (\alpha_1/\beta_1, \ldots, \alpha_p/\beta_p)$ is given in

lowest terms. We then compute $(\phi_j^{(a)})_j$ componentwise from a as follows:

$$\begin{aligned}
\phi_1^{(a)}&\colon (u;v) \mapsto (u_0, 0, \dots 0; 0, \dots, 0) \\
\phi_2^{(a)}&\colon (u;v) \mapsto (v_0, 0, \dots 0; 0, \dots, 0) \\
\phi_3^{(a)}&\colon (u;v) \mapsto (u_0, \dots, u_s; 0, \dots, 0) \\
\phi_4^{(a)}&\colon (u;v) \mapsto (v_0, \dots, v_s; 0, \dots, 0) \\
\phi_{4+j}^{(a)}&\colon (u;v) \mapsto (u_0 - v_0, u_j - v_j, 0, \dots, 0; 0, \dots, 0) && j \in \{1, \dots, s\} \\
\phi_{4+s+j}^{(a)}&\colon (u;v) \mapsto (u_{i_{1,j}} + v_0, u_{i_{3,j}} + v_{i_{2,j}}, 0, \dots, 0; 0, \dots, 0) && j \in \{1, \dots, k\} \\
\phi_{4+s+k+j}^{(a)}&\colon (u;v) \mapsto (\alpha_j u_0 - \beta_j u_j, 0, \dots, 0; 0, \dots, 0) && j \in \{1, \dots, p\} \\
\phi_{4+s+k+p+j}^{(a)}&\colon (u;v) \mapsto (\lambda_{0,j} u_0 + \sum_i \lambda_{i,j} u_i, 0, \dots, 0; 0, \dots, 0) && j \in \{1, \dots, l\};
\end{aligned}$$

note that only $\phi_{4+s+k+1}^{(a)}, \dots, \phi_{4+s+k+p}^{(a)}$ actually depend on the p-tuple a, and none of these matrices exist in the case $p = 0$. It remains to show that $a \in D(p, q, S)$ if and only if $(\phi_j^{(a)})_j \in E_{h,(h_j)_j}^{\mathbb{Q},m,r}$. We will show sufficiency and then necessity.

Suppose we are given $a = (\alpha_1/\beta_1, \dots, \alpha_p/\beta_p) \in \mathbb{Q}^p$ (in lowest terms) such that $a \in D(p, q, S)$, i.e., there exists $b \in \mathbb{Q}^q$ such that $(a, b) \in Z(S)$. Introduce the notation $(c_1, \dots, c_s) = (a_1, \dots, a_p, b_1, \dots, b_q)$ and define

$$V = \mathbb{Q}(1, c_1, \dots, c_s; 0, \dots, 0, \dots, 0) + \mathbb{Q}(0, \dots, 0; 1, c_1, \dots, c_s).$$

Each element of V has the form $(\mu, \mu c_1, \dots, \mu c_s, \nu, \nu c_1, \dots, \nu c_s)$ for $(\mu, \nu) \in \mathbb{Q}^2$; we thus identify elements of V by their coordinates (μ, ν). Clearly $\operatorname{rank}_{\mathbb{Q}}(V) = 2 = h$, as requested. We now check that for each component j, $\operatorname{rank}_{\mathbb{Q}}(\phi_j^{(a)}(V)) = h_j$. First and second, $\phi_1^{(a)}(V) = \mathbb{Q}(1, 0, \dots, 0; 0, \dots, 0) = \phi_2^{(a)}(V)$, confirming that

$$h_1 = \operatorname{rank}_{\mathbb{Q}}(\phi_1^{(a)}(V)) = 1 = \operatorname{rank}_{\mathbb{Q}}(\phi_2^{(a)}(V)) = h_2.$$

Third and fourth, $\phi_3^{(a)}(V) = \mathbb{Q}(1, c_1, \dots, c_s; 0, \dots, 0) = \phi_4^{(a)}(V)$, confirming that

$$h_3 = \operatorname{rank}_{\mathbb{Q}}(\phi_3^{(a)}(V)) = 1 = \operatorname{rank}_{\mathbb{Q}}(\phi_4^{(a)}(V)) = h_4.$$

For the next $j \in \{1, \dots, s\}$, each $x \in \phi_{4+j}^{(a)}(V)$ has the form

$$x = (\mu - \nu, \mu c_j - \nu c_j, 0, \dots, 0; 0, \dots, 0) = (\mu - \nu)(1, c_j, 0, \dots, 0; 0, \dots, 0)$$

for some $(\mu, \nu) \in \mathbb{Q}^2$, thus $\phi_{4+j}^{(a)}(V) = \mathbb{Q}(1, c_j, 0, \dots, 0; 0, \dots, 0)$, and so each $\operatorname{rank}_{\mathbb{Q}}(\phi_{4+j}^{(a)}(V)) = 1 = h_{4+j}$. For the next $j \in \{1, \dots, k\}$, since $c_{i_{3,j}} = c_{i_{1,j}} c_{i_{2,j}}$, each $x \in \phi_{4+s+j}^{(a)}(V)$ has the form

$$\begin{aligned}
x &= (\mu c_{i_{1,j}} + \nu, \mu c_{i_{3,j}} + \nu c_{i_{2,j}}, 0, \dots, 0; 0, \dots, 0) \\
&= (\mu c_{i_{1,j}} + \nu, \mu c_{i_{1,j}} c_{i_{2,j}} + \nu c_{i_{2,j}}, 0, \dots, 0; 0, \dots, 0) \\
&= (\mu c_{i_{1,j}} + \nu)(1, c_{i_{2,j}}, 0, \dots, 0; 0, \dots, 0),
\end{aligned}$$

so as before $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+j}(V)) = 1 = h_{4+s+j}$. For the next $j \in \{1,\ldots,p\}$, i.e., the components which depend on $a = (\alpha_1/\beta_1,\ldots,\alpha_p/\beta_p)$, each $x \in \phi^{(a)}_{4+s+k+j}(V)$ has the form

$$x = (\alpha_j\mu - \beta_j\mu a_j, 0,\ldots,0;0,\ldots,0) = (0,\ldots,0;0,\ldots,0),$$

confirming that each $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+k+j}(V)) = 0 = h_{4+s+k+j}$. Finally, for the last $j \in \{1,\ldots,l\}$, each $x \in \phi^{(a)}_{4+s+k+p+j}(V)$ has the form

$$x = (\lambda_{0,j}\mu + \sum_i \lambda_{i,j}\mu c_i, 0,\ldots,0;0,\ldots,0) = (0,\ldots,0;0,\ldots,0),$$

confirming that each $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+k+p+j}(V)) = 0 = h_{4+s+k+p+j}$. Thus we conclude that $a \in D(p,q,S)$ implies $(\phi^{(a)}_j)_j \in E^{\mathbb{Q},m,r}_{h,(h_j)_j}$.

To show necessity, suppose that we are given $a = (\alpha_1/\beta_1,\ldots,\alpha_p/\beta_p) \in \mathbb{Q}^p$ (in lowest terms) such that $(\phi^{(a)}_j)_j \in E^{\mathbb{Q},m,r}_{h,(h_j)_j}$ with the same parameters m, r, h, h_j; we will show there exists $b \in \mathbb{Q}^q$ such that $(a,b) \in Z(S)$. Pick any $V \leq_{\mathbb{Q}} \mathbb{Q}^r$ such that $\mathrm{rank}_{\mathbb{Q}}(V) = h$ and each $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_j(V)) = h_j$.

We first show that there exist elements $f, g \in V$ of the forms

$$f = (f_0,\ldots,f_s;0,\ldots,0) \qquad g = (0,\ldots,0;g_0,\ldots,g_s),$$

such that $f_0 = g_0 = 1$ and $\mathbb{Q}f + \mathbb{Q}g = V$. Let $d = (d_0,\ldots,d_s;d'_0,\ldots,d'_s)$ and $e = (e_0,\ldots,e_s;e'_0,\ldots,e'_s)$ be a $\mathbb{Q}$-basis of V. Since $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_1(V)) = h_1 = 1$, we may suppose that $d_0 = 1$: otherwise, if $0 \neq d_0 \neq 1$ then we can scale d so that $d_0 = 1$, except if $d_0 = 0$, in which case necessarily $e_0 \neq 0$, thus we can swap d and e and repeat this argument. Since $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_3(V)) = h_3 = 1$, there exists $\gamma \in \mathbb{Q}$ such that $\gamma(d_0,\ldots,d_s) = (e_0,\ldots,e_s)$. Let $g' = e - \gamma d = (0,\ldots,0;g'_0,\ldots,g'_s)$. Since g' and d are $\mathbb{Q}$-linearly independent elements of V and $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_4(V)) = h_4 = 1$, there exists $\delta \in \mathbb{Q}$ such that $\delta(g'_0,\ldots,g'_s) = (d'_0,\ldots,d'_s)$. Now define $f = d - \delta g' = (f_0,\ldots,f_s;0,\ldots,0)$; it has the desired form. Noting that $f = -\delta e + (1+\delta\gamma)d)$, g' and f are $\mathbb{Q}$-linearly independent elements of V, and since $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_2(V)) = h_2 = 1$, it follows that $g'_0 \neq 0$, so we can define $g = (1/g'_0)g'$, which also has the desired form. Additionally, $f_0 = g_0 = 1$ and $\mathbb{Q}f + \mathbb{Q}g = V$, as desired.

It follows that each element of V can be written as $\mu f + \nu g$ for some $(\mu,\nu) \in \mathbb{Q}^2$. For each $j \in \{1,\ldots,s\}$, each $x \in \phi^{(a)}_{4+j}(V)$ has the form

$$x = (\mu f_0 - \nu g_0, \mu f_j - \nu g_j, 0,\ldots,0;0,\ldots,0) = ((\mu-\nu)1, (\mu-\nu)f_j + \nu(f_j - g_j), 0,\ldots,0;0,\ldots,0);$$

if $f_j \neq g_j$, then $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+j}(V)) = 2 \neq h_{4+j} = 1$, a contradiction. Therefore, $f_j = g_j$ for all $j \in \{1,\ldots,s\}$, as well as $f_0 = g_0$ as shown earlier.

For each $j \in \{1,\ldots,p\}$, given $a = (\alpha_1/\beta_1,\ldots,\alpha_p/\beta_p)$, we have that, for any $(\mu,\nu) \in \mathbb{Q}^2$,

$$\phi^{(a)}_{4+s+k+j}(\mu f + \nu g) = (\alpha_j\mu f_0 - \beta_j\mu f_j, 0,\ldots,0;0,\ldots,0);$$

since $\mathrm{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+k+j}(V)) = h_{4+s+k+j} = 0$, it follows that $\beta_j\mu f_j = \alpha_j\mu f_0 = \alpha_j\mu$; now picking $(\mu,\nu) \in \mathbb{Q}^2$ such that $\mu \neq 0$, we have that $f_j = a_j$ for all $j \in \{1,\ldots,p\}$.

Now consider any $F \in S$. If $F = \lambda_{0,j} + \sum_i \lambda_{i,j} z_i$ for some $j \in \{1, \ldots, l\}$, then because $\operatorname{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+k+p+j}(V)) = h_{4+s+k+p+j} = 0$, then $\lambda_{0,j} + \sum_i \lambda_{i,j} f_i = 0$; that is, $F(f_i)_i = 0$. Otherwise, $F = z_{i_{1,j}} z_{i_{2,j}} - z_{i_{3,j}}$ for some $j \in \{1, \ldots, k\}$; because $\operatorname{rank}_{\mathbb{Q}}(\phi^{(a)}_{4+s+j}(V)) = h_{4+s+j} = 1$, then there exists $\gamma \in \mathbb{Q}$ such that $\gamma(\mu f_{i_{1,j}} + \nu) = \mu f_{i_{3,j}} + \nu f_{i_{2,j}}$ for any $(\mu, \nu) \in \mathbb{Q}^2$. Considering the case $\nu = 0$ and $\mu = 1$, it follows that if $f_{i_{1,j}} = 0$ then $f_{i_{3,j}} = 0$; in this case, $f_{i_{1,j}} f_{i_{2,j}} = f_{i_{3,j}}$. Otherwise, $f_{i_{1,j}}, f_{i_{3,j}} \neq 0$, and we have $\gamma = f_{i_{3,j}} / f_{i_{1,j}}$, which we substitute to obtain $f_{i_{3,j}} / f_{i_{1,j}} = f_{i_{2,j}}$. That is, again, $f_{i_{1,j}} f_{i_{2,j}} = f_{i_{3,j}}$, so $F(f_i)_i = 0$.

In conclusion, letting $b = (f_{p+1}, \ldots, f_s)$, we have shown that $(f_1, \ldots, f_s) = (a, b) \in Z(S)$. □

Proof of Theorem 4.4. Together, Lemmas 4.52, 4.53, and 4.54 complete the proof of Theorem 4.4. As in the case of Theorem 4.3, the statement of Theorem 4.4 tacitly assumes a particular representation of $\mathcal{G}$. □

Chapter 5

Communication Bounds for Loop Nests

In this chapter we apply the communication lower bounds framework developed in Section 2.3 to the class of algorithms characterized in Section 3.2, and study the attainability of these bounds. A section-by-section outline is as follows.

- In Section 5.1 we prove Theorem 3.1 (stated in Section 3.2), which yields communication lower bounds via Corollary 2.1 (in Section 2.3).

- In Section 5.2, we discuss attainability of the lower bounds, proposing *blocked implementations* as promising candidates. In Theorem 5.1, we suggest a class of *tiles* which in many cases attain equality in the conclusion (4.2) of Theorem 4.1 (in Chapter 4), and can be used to derive blocked implementations in many important cases.

- In Section 5.3, we discuss the *infeasible case*, the class of HBL interpretations where the hypothesis $\bigcap_j \ker(\phi_j) = \{0\}$ of Theorem 3.1 fails. In this case, the linear program (3.2) defining the parameter σ is infeasible; this turns out to be promising from the standpoint of minimizing communication, because arbitrary data reuse may be possible.

- In Section 5.4, we discuss the *injective case*, the class of HBL interpretations where at least one homomorphism ϕ_j is an injection. In this case, we always have the parameter $\sigma = 1$, its minimum value. This case turns out to be the least promising from the standpoint of minimizing communication, because no asymptotic data reuse is possible.

- In Section 5.5, we discuss the *product case*, mentioned in the context of Lemma 4.30 (in Section 4.4.5). This models nested-loop programs whose array subscripts are subsets of the loop indices, e.g., the matrix multiplication example at the beginning of Chapter 3, which has loop indices i, j, k and array subscripts $(i, k), (k, j), (i, j)$, corresponding to the array elements $A(i, k), B(k, j), C(i, j)$. In the product case, communication lower bounds can be computed by simpler means than using the algorithm in Theorem 4.3 (stated in Section 4.2); moreover, Theorem 5.1 can be specialized to yield a stronger result (see Corollary 5.1). In Sections 5.5.1-5.5.5 we treat several examples of

the product case including dot products, matrix-vector multiplication, matrix-matrix multiplication, tensor operations, and many-body calculations.

5.1 Lower Bounds for Loop Nests

Recall that Theorem 3.1 (in Section 3.2) gives bounds on input-/output-path cutsizes $Q_{\text{in}}, Q_{\text{out}}$ for algorithms with HBL interpretations. In particular, each HBL interpretation defines a real number $\sigma \geq 1$ such that $Q_{\text{in}}(J) + Q_{\text{out}}(J) \geq |J|^{1/\sigma}$ for all nonempty subsets J of some subset K of the algorithm. We can then apply Corollary 2.1 to derive communication lower bounds, as shown following the statement of Theorem 3.1. That is, given an algorithm with an HBL interpretation, in any implementation of that algorithm where some processor p, with M_p local memory cells, is assigned the $N = |K|$ Operations corresponding to the evaluations $K \subseteq F$, if $N \geq (4M_p)^\sigma$, then by Corollary 2.1, processor p performs at least

$$\frac{1}{2 \cdot 3^\sigma} \cdot \frac{N}{M_p^{\sigma-1}} = \Omega\left(\frac{N}{M_p^{\sigma-1}}\right)$$

Loads and Stores between their first and last assigned Operation corresponding to K.

We now prove Theorem 3.1. Recall that we are given the HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$, and recall the definition of $\mathcal{P}(\mathcal{G})$ from (4.8).

Proof of Theorem 3.1. Theorem 4.1 tells us that if $s \in \mathcal{P}(\mathcal{G})$, then for all nonempty finite $E \subseteq \mathbb{Z}^r$, $|E| \leq \prod_j |\phi_j(E)|^{s_j}$. Therefore, since $\mathcal{P}(\mathcal{G})$ is a closed subset of the first orthant of $\mathbb{R}^m$, and since $\mathcal{P}(\mathcal{G}) \neq \emptyset$ via Lemma 4.10 and the hypothesis $\bigcap_j \ker(\phi_j) = \{0\}$, we can take a minimum over $s \in \mathcal{P}(\mathcal{G})$,

$$\begin{aligned}
|E| &\leq \min_{s \in \mathcal{P}(\mathcal{G})} \prod_j |\phi_j(E)|^{s_j} \\
&\leq \min_{s \in \mathcal{P}(\mathcal{G})} \prod_j (\max_{i=1}^{m} |\phi_i(E)|)^{s_j} \\
&= \min_{s \in \mathcal{P}(\mathcal{G})} (\max_j |\phi_j(E)|)^{\sum_j s_j} \\
&= (\max_j |\phi_j(E)|)^\sigma,
\end{aligned}$$

where

$$\sigma = \sigma(\mathcal{G}) = \min_{s \in \mathcal{P}(\mathcal{G})} \sum_j s_j. \tag{5.1}$$

We have that $\sigma \geq 1$ by Lemma 4.12 (by hypothesis, $1 \leq r = \text{rank}(\mathbb{Z}^r)$); additionally, since $s \in \mathcal{P}(\mathcal{G})$ implies $(\min(1, s_j))_j \in \mathcal{P}(\mathcal{G})$ by Lemma 4.27, $\min_{s \in \mathcal{P}(\mathcal{G})} \sum_j s_j = \min_{s \in \mathcal{P}(\mathcal{G}) \cap [0,1]^m} \sum_j s_j$, so $\sigma \leq \sum_j 1 = m$.

Having established the existence of $\sigma \in [1, m]$ such that $\max_j |\phi_j(E)| \geq |E|^{1/\sigma}$ for all nonempty finite $E \subseteq \mathbb{Z}^r$, consider the case $E = Z(J)$ for any nonempty $J \subseteq K$. By definition, $A(J) = \bigcup_j A_j(\phi_j(Z(J)))$, and for each j, $|A_j(\phi_j(Z(J)))| = |\phi_j(Z(J))|$ by injectivity of A_j. So

$$|A(J)| = \left|\bigcup_j A_j(\phi_j(Z(J)))\right| \geq \max_j |\phi_j(Z(J))| \geq |Z(J)|^{1/\sigma} = |J|^{1/\sigma},$$

where the last equality follows from injectivity of Z. The conclusion follows from the fact that $Q_{\text{in}}(J) + Q_{\text{out}}(J) \geq |A(J)|$, noted before the statement of Theorem 3.1. □

As remarked following the statement of Theorem 3.1, the exponent σ is the solution of a linear program, $\min_{s \in \mathcal{P}(\mathcal{G})} \sum_j s_j$. We have studied the linear inequalities defining $\mathcal{P}(\mathcal{G})$ in depth in Sections 4.4 and 4.7. In particular, since $G = \mathbb{Z}^r$ is finitely generated, we can apply Theorem 4.3 to compute a finite set of $\mathbb{Z}$-linear inequalities which define $\mathcal{P}(\mathcal{G})$, and then (computably) solve this linear program for σ. Moreover, note that $\sigma \in \mathbb{Q}$: the extreme points of $\mathcal{P}(\mathcal{G})$ are $\mathbb{Q}$-valued and the hyperplane defined by $\sum_j s_j = \sigma$ must intersect at least one extreme point of $\mathcal{P}(\mathcal{G})$.

We mention several ways in which the computation of $\sigma(\mathcal{G})$ can be simplified. First, by Lemma 4.14, if there exist distinct indices $k, l \in \{1, \ldots, m\}$ such that $\text{rank}(\ker(\phi_k) \cap \ker(\phi_l)) = \text{rank}(\ker(\phi_k)) = \text{rank}(\ker(\phi_l))$, then letting $\mathcal{G}' = (G, (G_j)_{j \neq l}, (\phi_j)_{j \neq l})$, we have that $\sigma(\mathcal{G}) = \sigma(\mathcal{G}')$: that is, dropping 'duplicate' homomorphisms does not affect the communication lower bound. Second, by Lemma 4.15, if there exists an index $k \in \{1, \ldots, m\}$ such that $\text{rank}(\phi_k(G)) = 0$, letting $\mathcal{G}' = (G, (G_j)_{j \neq k}, (\phi_j)_{j \neq k})$, we have have $\sigma(\mathcal{G}) = \sigma(\mathcal{G}')$: that is, dropping rank-0 homomorphisms does not affect the communication lower bound. Third, by Lemma 4.16, if there exists an index $k \in \{1, \ldots, m\}$ such that $\text{rank}(\phi_k(G)) = \text{rank}(G)$, then $e_k \in \mathcal{P}(\mathcal{G})$; since $\text{rank}(G) \geq 1$, by Lemma 4.12 we conclude that $\sigma(\mathcal{G}) = 1$: that is, we need only identify a single corank-0 homomorphism to identify the communication lower bound. Fourth, if the hypotheses of Lemma 4.30 apply, then the set $\mathcal{P}(\mathcal{G})$ has a simpler definition, which we discuss below in Section 5.5.

5.2 Upper Bounds for Loop Nests

We now turn to the question of the attainability of our communication lower bounds in Section 2.3: given an algorithm with an HBL interpretation, can we find an implementation with communication cost within a constant factor of the lower bound? This question is imprecise because there are actually multiple communication lower bounds in Section 2.3, and their derivations involved a number of steps that (possibly) weakened the ultimate results; in order to clarify what we are attaining, we first review the main lower bound results of Section 2.3.

The main lower bound results of Section 2.3 — Theorem 2.1 and its special case Corollary 2.1 — are stated for a fixed algorithm with parameters V, F, x, y, I, O (see Section 2.2 for definitions). Both results hypothesize that there exists a nonempty finite $K \subseteq F$ such that, for all nonempty $J \subseteq K$, the sum $Q_{\text{in}}(J) + Q_{\text{out}}(J)$ of the input- and output-path cutsizes of J is bounded below by a nondecreasing function $q(|J|)$. Both results then conclude that in any implementation of this algorithm in which some processor p is assigned the Operations corresponding to K, then processor p must perform at least a certain number of Loads and Stores.

We now define a special class of implementations, called *blocked implementations*, whose definition is motivated by the segmentation argument in the proof of Theorem 2.1. Consider any algorithm with parameters V, F, x, y, I, O, and any of its implementations. Identifying F with the corresponding Operations, the assignment of instructions to processors induces

a disjoint cover $F = \bigcup_p F_p$, and for each p, F_p induces a chain in the execution order. We say that the given implementation is *blocked* it, for each p, we can write the chain $F_p = B_{p,1} \cdots B_{p,b_p}$ as the concatenation of some $b_p \in \mathbb{N}$ nonempty contiguous subchains $B_{p,i}$, called *blocks*, such that processor p's chain of instructions is a repetition, for each $i \in [b_p]$, of the three-phase instruction sequence,

$$\begin{aligned} \text{Loads,} &\quad \text{for each } v \in \mathcal{X}_{p,i} = X(B_{p,i}) \setminus Y(B_{p,i}) \\ \text{Operations,} &\quad \text{for each } f \in B_{p,i} \\ \text{Stores,} &\quad \text{for each } v \in \mathcal{Y}_{p,i} = Y(B_{p,i}) \cap \Big(O \cup \bigcup_{(q,j) \neq (p,i)} X(B_{q,j}) \Big). \end{aligned}$$

Here we have also introduced the notations $X(J) = \{x_{f,1}, \ldots, x_{f,m_f} \mid f \in J\}$ and $Y(J) = \{y_{f,1}, \ldots, y_{f,n_f} \mid f \in J\}$ for any $J \subseteq F$. It follows that any blocked implementation does not perform 'extra' Operations, i.e., there is a bijection between F and the set of Operations in the execution. The key property of a blocked implementation is that the Operations in each block $B_{p,i}$ are performed without intervening Loads and Stores. It follows that, for each block $B_{p,i}$, $|\mathcal{X}_{p,i}| \leq M_p$ and $|\mathcal{Y}_{p,i}| \leq M_p$. Therefore, each processor p's communication cost is bounded above,

$$W_p \leq 2M_p b_p.$$

Note that this upper bound is reminiscent of the lower bound (2.1) (Section 2.3) that we are trying to attain (in the case $l = 2M_p$). Our approach to deriving communication-reducing implementations, given an algorithm and machine parameters P and $(M_p)_p$, is to construct a blocked implementation that attempts to minimize

$$\max_p 2M_p b_p \geq \max_p W_p.$$

We do not claim that blocked implementations minimize $\max_p W_p$: for example, we are neglecting the possibility of inter-block data reuse in each processor's local memory (some Loads/Stores could be avoided).

We are particularly interested in the attainability of the (weaker) conclusion of Corollary 2.1. Consider any implementation that minimizes $\max_p W_p$, and look at any processor p with maximal W_p. Corollary 2.1 hypothesizes that there exists a nonempty $K \subseteq F_p$ and a $\sigma \in [1, \infty)$ such that $|J|^{1/\sigma} \leq Q_{\text{in}}(J) + Q_{\text{out}}(J)$ for all nonempty $J \subseteq K$. Additionally, if $N = |K| \geq (4M_p)^\sigma$, then Corollary 2.1 gives the (nontrivial) lower bound

$$W_p \geq \frac{1}{2 \cdot 3^\sigma} \frac{N}{M_p^{\sigma-1}} = 2M_p \cdot \frac{N}{4(3M_p)^\sigma}.$$

This motivates us to seek a blocked implementation where $b_p \leq CN/M_p^\sigma$ where C is independent of the machine parameters P and $(M_p)_p$, as well as N; C will, however, depend on another aspect of the HBL interpretation, its induced HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j}), (\phi_j)_j)$.

So far, we have not developed a general framework for deriving blocked implementations (for algorithms with or without HBL interpretations). The examples treated later in this chapter have dependence structures which enable a simpler, geometry-inspired blocking approach which we call *tiling*. For example, as mentioned in Chapter 3, multiplying

N-by-N matrices, $C = A \cdot B$, can be programmed with triply nested loops (assuming C is zero-initialized),

$$\begin{aligned}&\textbf{for } i = 1 : N, \quad \textbf{for } j = 1 : N, \quad \textbf{for } k = 1 : N,\\ &\quad C(i,j) = C(i,j) + A(i,k) * B(k,j).\end{aligned}$$

A blocked version of this code, where the blocking factor $b \in \mathbb{N}_{\geq 1}$ is assumed to divide N, is given by

$$\begin{aligned}&\textbf{for } i_1 = 1 : N/b, \quad \textbf{for } j_1 = 1 : N/b, \quad \textbf{for } k_1 = 1 : N/b,\\ &\quad \textbf{for } i_2 = 1 : b, \quad \textbf{for } j_2 = 1 : b, \quad \textbf{for } k_2 = 1 : b,\\ &\quad\quad // \text{ reindexing } (i,j,k) = b \cdot (i_1 - 1, j_1 - 1, k_1 - 1) + (i_2, j_2, k_2)\\ &\quad\quad C(i,j) = C(i,j) + A(i,k) * B(k,j).\end{aligned}$$

The inner three loops (indexed by i_2, j_2, k_2) represent blocks, of which there are $(N/b)^3$, traversed by the outer three loops (indexed by i_1, j_1, k_1). Each block requires $3b^2$ local memory cells to execute: the submatrices of A, B, C, b^2 entries each, must be loaded, and the submatrix of C must be stored, but each updated entry of C can overwrite its previous value. Therefore, we can define a blocked implementation by picking b such that $3b^2 \leq M_p$ for some processor p, and assigning all the work to that processor p. A parallel implementation can be defined, e.g., by replacing M_p by $\min_p M_p$, and assigning blocks to processors, ensuring that the execution order preserves the partial order on blocks, in particular, when multiple processors are assigned blocks that update the same submatrix of C. (Additionally, more care must be taken when trying to minimize the critical-path communication cost, rather than the per-processor communication costs.) We will revisit this example in Section 5.5.3 more formally (in terms of algorithms), to illustrate the algorithmic dependence structure that permits this code transformation. To foreshadow this discussion, we observe that while the unblocked example suggests a lexicographical order on $(i,j,k) \in [N]^3$, the only inter-iteration dependences are due to the summation order, which impose a lexicographical order on the set $\{(i,j,k) \mid k \in [N]\}$ for each $(i,j) \in [N]^2$, i.e., a partial order on $[N]^3$, and the blocked code respects this partial order.

This matrix multiplication example demonstrates *tiling* $\mathbb{Z}^3$ (in particular, the cube $[N]^3$) by translates of the cube $[b]^3$. In the remainder of this section, we develop a tool (see Theorem 5.1), to assist in finding tilings, which can use more general parallelotopes than cubes like $[b]^3$. This development is motivated by attainability of the upper bounds given by Theorem 4.1 (Section 5.2). That is, given some $s \in \mathcal{P}(\mathcal{G})$, does there exist a nonempty finite $E \subseteq \mathbb{Z}^r$ such that $|E| = \prod_j |\phi_j(E)|^{s_j}$? Actually, recalling the invocation of Theorem 4.1 within the proof of Theorem 3.1, we are really concerned with whether $|E| = \max_j |\phi_j(E)|^\sigma$, where $\sigma = \sigma(\mathcal{G}) = \min_{s \in \mathcal{P}(\mathcal{G})} \sum_j s_j \in [1, m]$ and we assume $r \geq 1$. Of course, either of these equalities are satisfied by taking E to be any singleton — so to get a meaningful answer, we must impose additional constraints.

Let's borrow some notation from Chapter 4: given the HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$ where $r \in \{1, 2, \ldots\}$, we let $\mathcal{H}^*$ denote the set of all nontrivial subgroups of $\mathbb{Z}^r$ and let $\mathcal{E}^*$ denote the set of all nonempty finite subsets of $\mathbb{Z}^r$. For any $\mathcal{H} \subseteq \mathcal{H}^*$, we define $\mathcal{P}(\mathcal{G}, \mathcal{H})$ according to (4.8) in Section 4.4, with the special case $\mathcal{P}(\mathcal{G}) = \mathcal{P}(\mathcal{G}, \mathcal{H}^*)$. We also extend

$\sigma = \sigma(\mathcal{G})$ to the more general notation, continuing to assume $\bigcap_j \ker(\phi_j) = \{0\}$,

$$\sigma(\mathcal{G}, \mathcal{H}) = \min_{s \in \mathcal{P}(\mathcal{G},\mathcal{H})} \sum_j s_j, \tag{5.2}$$

and say $\sigma(\mathcal{G})$ to mean $\sigma(\mathcal{G}, \mathcal{H}^*)$ from now on; as in the case of $\sigma(\mathcal{G})$, $\sigma(\mathcal{G}, \mathcal{H}) \in [1, \sigma(\mathcal{G})] \subseteq [1, m]$ is well defined because $\mathcal{P}(\mathcal{G}, \mathcal{H})$ is a closed subset of the first orthant of $\mathbb{R}^m$.

A nonempty finite subset $\mathcal{H} = \{H_1, \ldots, H_k\}$ of $\mathcal{H}^*$ is *independent* if the subgroups' sum is isomorphic to their direct sum, i.e., $H = \sum_{i=1}^k H_i \cong \bigoplus_{i=1}^k H_i$, or, equivalently, if we can find a basis B (over $\mathbb{Z}$) of H that can be partitioned as $\{B_1, \ldots, B_k\}$, where for each $i \in \{1, \ldots, k\}$, B_i is a basis of H_i.

Theorem 5.1. *If $\bigcap_j \ker(\phi_j) = \{0\}$ and $\mathcal{H} \subseteq \mathcal{H}^*$ is independent, then there exists $\mu \in \{1, 2, \ldots\}$, $c \in (0, \infty)$, $\tau \in [\sigma(\mathcal{G}, \mathcal{H}), \sigma(\mathcal{G})]$, and a sequence $(E_1, E_2, \ldots)$ over $\mathcal{E}^*$ such that, for all $N \in \{\mu, \mu + 1, \ldots\}$,*

Property (1) $\sum_j |\phi_j(E_N)| \leq N$,

Property (2) $|E_N| \geq cN^\tau$, *and*

Property (3) $\mathbb{Z}^r$ *is partitioned by translates of* E_N.

Proof. We adopt a few notational conveniences. We reserve the index i to range over $\{1, \ldots, k\}$, while j continues to range over $\{1, \ldots, m\}$. We treat the Abelian group $\mathbb{Z}^d$ and its subgroups as (sub-) $\mathbb{Z}$-modules, defining (scalar) multiplication in terms of addition. In this case, we assume the standard basis $(e_1, \ldots, e_d)$ on $\mathbb{Z}^d$ and represent tuples over $\mathbb{Z}$ as column vectors. If B is a column vector, or a set or matrix of column vectors, then $\langle B \rangle$ is the set of all linear combinations of B over $\mathbb{Z}$. If y is an n-element column vector, X, X_i are sets of n-element column vectors, and A is an n-column matrix, $y + X$ denotes the set $\{y + x \mid x \in X\}$, $\sum_i X_i$ denotes the set $\{\sum_i x_i \mid x \in \bigtimes_i X_i\}$, and $A \cdot X$ denotes the set $\{Ax \mid x \in X\}$.

For each $n \in \{1, 2, \ldots\}$, $h \in \{1, \ldots, r\}$, rank-h matrix $B = [b_1, \ldots, b_h] \in \mathbb{Z}^{r \times h}$, and $t \in \mathbb{Z}^r$, we define the *cube*

$$C(n, h, B, t) = t + B \cdot \{0, \ldots, n-1\}^h.$$

Suppose we have k cubes $C_i = C(n_i, h_i, B_i, t_i)$ and we define $h = \sum_i h_i$, $B = [B_1, \ldots, B_k]$, and $t = \sum_i t_i$. If B is rank-h, then we define the *tile*

$$D(C_1, \ldots, C_k) = t + B \cdot \bigtimes_i \{0, \ldots, n_i - 1\}^{h_i}.$$

For each i, let $h_i = \operatorname{rank}(H_i)$ and let B_i be an arbitrary basis of H_i represented as a matrix of column vectors, defining the family of cubes $C(N, h_i, B_i, 0)$ for $N \in \{1, 2, \ldots\}$. Adapting the argument in the proof of Lemma 4.48 (Section 4.6), for each j, there exists $A_{i,j} \in [1, \infty)$ such that $|\phi_j(C(N, h_i, B_i, 0))| \leq A_{i,j} N^{\operatorname{rank}(\phi_j(H_i))}$ for all N; in case $\operatorname{rank}(\phi_j(H_i)) = 0$ we set $A_{i,j} = 1$. Let $x = (x_1, \ldots, x_k)$ be any optimal solution of the linear program

$$\max \left\{ \sum_i x_i \operatorname{rank}(H_i) \,\middle|\, x \in [0, \infty)^k \wedge (\forall j)\, 1 \geq \sum_i x_i \operatorname{rank}(\phi_j(H_i)) \right\}, \tag{5.3}$$

which is the dual of the linear program (5.2) for the given $\mathcal{H}$, whose optimal value $\sigma(\mathcal{G},\mathcal{H})$ we have already confirmed exists; therefore, we know that $\sum_i x_i \operatorname{rank}(H_i) = \sigma(\mathcal{G},\mathcal{H})$. We can also show that any feasible dual solution $x \in [0,1]^m$: supposing towards a contradiction that some $x_i > 1$, it follows that $\operatorname{rank}(\phi_j(H_i)) = 0$ for all j, so $H_i \leq \bigcap_j \ker(\phi_j)$, but $\operatorname{rank}(H_i) > 0$ by hypothesis, contradicting the hypothesis that $\bigcap_j \ker(\phi_j) = \{0\}$. Additionally, since $\sigma(\mathcal{G},\mathcal{H}) \geq 1$, at least one $x_i > 0$. Consider each index i: if $x_i = 0$ then we set $n_i(N) = 1$ and $\mu_{i,j} = 1$ for all j. Otherwise, if $x_i > 0$, consider each index j; let μ_i be the smallest $N \in \{1,2,\ldots\}$ such that $n_i(N) = \lfloor N^{x_i}/(m\prod_j A_{i,j})\rfloor > 0$; μ_i is well defined since $N^{x_i}/(m\prod_j A_{i,j})$ is an increasing function of N; note that $\mu_i \geq m^{1/x_i} \geq m$, since $x_i \in (0,1]$. Let $\mu = \max_i \mu_i \geq m$.

For each $N \in \{1,\ldots,\mu-1\}$, let $E_N = \{0\}$. For each $N \in \{\mu,\mu+1,\ldots\}$, let $C_i(N) = C(n_i(N), h_i, B_i, 0)$ for each i; by independence, $B = [B_1,\ldots,B_k]$ has rank $h = \sum_i h_i$ so we can define the tile $E_N = D(C_1(N),\ldots,C_k(N))$. We now verify that the sequence of tiles $E_1, E_2, \ldots$ has the three desired properties.

Property (1). Consider any $N \in \{\mu,\mu+1,\ldots\}$. For each j,

$$\begin{aligned}\phi_j(E_N) = \phi_j(D(C_1(N),\ldots,C_k(N))) &= \phi_j\Big(B \cdot \bigtimes_i \{0,\ldots,n_i(N)-1\}^{h_i}\Big)\\ &= \sum_i \phi_j(B_i \cdot \{0,\ldots,n_i(N)-1\}^{h_i})\\ &= \sum_i \phi_j(C_i(N));\end{aligned}$$

Consider any $k \in \{1,\ldots,m\}$. Let $I = \{i \mid x_i > 0 \wedge \operatorname{rank}(\phi_k(H_i)) > 0\}$; if $i \notin I$, then $|\phi_k(C_i(N))| = 1$. So, if $I = \emptyset$, then $|\phi_k(E_N)| = 1$. Otherwise, $I \neq \emptyset$, then

$$\begin{aligned}|\phi_k(E_N)| &\leq \prod_i |\phi_k(C_i(N))| = \prod_{i\in I} |\phi_k(C_i(N))| \leq \prod_{i\in I} A_{i,k} n_i^{\operatorname{rank}(\phi_k(H_i))}\\ &= \prod_{i\in I} A_{i,k} \left\lfloor \frac{N^{x_i}}{m\prod_j A_{i,j}} \right\rfloor^{\operatorname{rank}(\phi_k(H_i))} \leq \prod_{i\in I} A_{i,k} \left(\frac{N^{x_i}}{m\prod_j A_{i,j}}\right)^{\operatorname{rank}(\phi_k(H_i))}\\ &= \left(\prod_{i\in I} \frac{A_{i,k}}{\left(m\prod_j A_{i,j}\right)^{\operatorname{rank}(\phi_k(H_i))}}\right) \prod_{i\in I} N^{x_i \operatorname{rank}(\phi_k(H_i))} \leq \frac{1}{m}\prod_{i\in I} N^{x_i \operatorname{rank}(\phi_k(H_i))}\\ &= \frac{1}{m} N^{\sum_i x_i \operatorname{rank}(\phi_k(H_i))} \leq \frac{N^1}{m}.\end{aligned}$$

where we have exploited the facts that all $A_{i,j} \geq 1$, that $\operatorname{rank}(\phi_k(H_i)) \geq 1$ for all $i \in I$, and that $\sum_i x_i \operatorname{rank}(\phi_k(H_i)) \leq 1$ by the fact that x is a feasible solution of (5.3).

Since $N \geq \mu \geq m$, we have thus shown that for all j, $|\phi_j(E_N)| \leq \lfloor N/m \rfloor$, therefore

$$\sum_j |\phi_j(E_N)| \leq \sum_j \left\lfloor \frac{N}{m} \right\rfloor = m \left\lfloor \frac{N}{m} \right\rfloor \leq N,$$

as desired.

Property (2). Consider any $N \in \{\mu, \mu+1, \ldots\}$; by independence (of $\mathcal{H}$ and B) and definition of μ,

$$|E_N| = \prod_i \left\lfloor \frac{N^{x_i}}{m \prod_j A_{i,j}} \right\rfloor^{h_i} \geq \prod_i \left(\frac{N^{x_i}}{2m \prod_j A_{i,j}} \right)^{h_i} = \left(\prod_i \frac{1}{\left(2m \prod_j A_{i,j}\right)^{h_i}} \right) \prod_i N^{x_i h_i};$$

letting $c = \prod_i \left(2m \prod_j A_{i,j}\right)^{-h_i} \in (0, 1]$, we have that

$$|E_N| \geq c \prod_i N^{x_i h_i} = c N^{\sum_i x_i \operatorname{rank}(H_i)} = c N^{\sigma(\mathcal{G},\mathcal{H})}.$$

We also have that

$$|E_N| \leq (\max_j |\phi_j(E_N)|)^{\sigma(\mathcal{G})} \leq (\sum_j |\phi_j(E_N)|)^{\sigma(\mathcal{G})} \leq N^{\sigma(\mathcal{G})},$$

Therefore, there exists $c \in (0, \infty)$ and $\tau \in [\sigma(\mathcal{G}, \mathcal{H}), \sigma(\mathcal{G})]$ such that, for all $N \in \{\mu, \mu+1, \ldots\}$, $|E_N| \geq cN^\tau$.

Property (3). For each $N \in \{1, 2, \ldots\}$, the cubic structure of E_N makes it straightforward to partition $\mathbb{Z}^r$ with its translates as follows. If $N < \mu$, then $E_N = \{0\}$ by construction and the conclusion is immediate: $\mathbb{Z}^r$ is partitioned by the set of all its singletons. Otherwise $N \geq \mu$, and for each i, the cube C_i can be translated to partition H_i by varying t_i over the linear combinations of the columns of B_i by all integer multiples of n_i. By partitioning each subgroup H_i with translated cubes in this manner, we obtain a partition of $\bigoplus_i H_i$ and thus of $H = \sum_i H_i$ by isomorphism, and each part in this induced partition of H is a tile, in particular, $t + E_N$ for some $t \in H$. The cosets of H in $\mathbb{Z}^r$ (i.e., the elements of the quotient group $\mathbb{Z}^r/H$) partition $\mathbb{Z}^r$, and each coset $y + H$ is a translation of H by some representative $y \in \mathbb{Z}^r$, so the partition of H (i.e., the coset $0 + H$) into tiles induces partitions into tiles of all the other cosets of H by translation by y. □

The constant c can be improved in a number of ways. For example, we can weakening Property 1 to bound $\max_j |\phi_j(E_N)|$ instead, we can model some instances of aliasing arrays (see Section 3.2). Also, a few simplifications to the dual linear program (5.3) are possible. For example, the constraints corresponding to any rank-0 homomorphisms ϕ_j have no impact, and could be omitted. Other possible simplifications include combining duplicate homomorphisms $\phi_k = \phi_l$, as discussed in the preceding section. In the case of injective ϕ_j, since $\sigma(\mathcal{G}) = 1$, a simpler special case of Theorem 5.1 applies, as discussed below in Section 5.4.

At this point, we have defined blocked implementations, which enable a simpler communication upper bound analysis, as well as defined a tool, Theorem 5.1, for finding tilings of $\mathbb{Z}^r$ that have certain properties with respect to a given HBL datum. It remains future work to develop a general approach for using Theorem 5.1 to define blocked implementations of given algorithms with HBL interpretations. In the remainder of this chapter, we apply Theorem 5.1 to a sequence of examples, demonstrating its practical use.

5.3 Infeasible Case

Our first class of examples is called the *infeasible case*. These are algorithms with HBL interpretations where $\mathcal{P}((\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)) = \emptyset$, i.e., the linear program (5.1) is infeasible. By Lemma 4.10, this means that $\text{rank}(\bigcap_j \ker(\phi_j)) > 0$.

As an example of the infeasible case, recall the "path graph" example in Section 3.1.3 (see also Figure 3.3), meant to model a loop like

$$\textbf{for } i = 1 : N, \quad a_i = f(a_{i-1}),$$

which transform an input a_0 to an output a_N; a single processor can perform the corresponding sequence of Ops entirely in their local memory, without any communication other than (perhaps) initially Loading a_0. From the perspective of our lower-bound analysis, infeasibility is the best-case scenario for minimizing communication. We interpret the preceding code as a family of algorithms whose parameters satisfy

$$V = \{a_0, a_i \mid i \in [N]\}, \quad F = \{f_i \mid i \in [N]\}, \quad x\colon (f_i, 1) \mapsto a_{i-1}, \quad y\colon (f_i, 1) \mapsto a_i. \tag{5.4}$$

Such algorithms admit the HBL interpretation

$$m = 2, \quad r = 1, \quad r_1 = r_2 = 0, \quad K = F, \quad Z(f_i) = i,$$
$$\phi_1(i) = \phi_2(i) = 0, \quad A_1(0) = a_0, \quad A_2(0) = a_N$$

This induces the HBL datum $\mathcal{G} = (\mathbb{Z}^1, (\mathbb{Z}^0)^2, (\mathbb{Z}^1 \to \mathbb{Z}^0)^2)$, and we immediately see that $\bigcap_j \ker(\phi_j) = \mathbb{Z}^1$, implying infeasibility.

Suppose the primal linear program $\min\{\sum_j s_j \mid s \in \mathcal{P}(\mathcal{G})\}$ is infeasible. Since $x = (0, \ldots, 0)$ is always a feasible solution of the dual linear program (5.3), the dual must be unbounded. This confirms the intuition in the preceding section, that we can increase the tile edge lengths (e.g., via the dual variables $\{x_i\}$) without bound. Note that for arbitrary $\mathcal{H} \subseteq \mathcal{H}^*$, possibly $\mathcal{P}(\mathcal{G}, \mathcal{H}) \neq \emptyset$ even if $\mathcal{P}(\mathcal{G}) = \emptyset$. That is, the (generalized) primal linear program $\min\{\sum_j s_j \mid s \in \mathcal{P}(\mathcal{G}, \mathcal{H})\}$ may be feasible, and the primal/dual solution $\sigma(\mathcal{P}, \mathcal{H}) \in [1, \sigma(\mathcal{P})]$ as otherwise expected. For example, $H \cap \bigcap_j \ker(\phi_j) = \{0\}$ for all $H \in \mathcal{H}$. If algorithm dependences do not permit a tiling that takes care of infeasibility in the manner discussed next, then it may be worthwhile to study such an $\mathcal{H}$.

Note than in the path graph example above, picking $E_N = \{0, \ldots, N-1\}$, we have that $|\phi_1(E_N)| = 1 \leq N$ and $|E_N| = N = N^1$ for all $N \in \{1, 2, \ldots\}$. In the case of more general infeasible HBL data, we can pick E_N as a cube in $\bigcap_j \ker(\phi_j)$ and apply similar reasoning.

5.4 Injective Case

Our next class of examples is called the *injective case*. These are algorithms with HBL interpretations which induce HBL data $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$ where at least one homomorphism ϕ_j is an injection, and $\sigma(\mathcal{G}) = 1$.

As an example of the injective case, recall the example in Section 3.1.1 (see also Figure 3.1), meant to model a loop like

$$\textbf{for } i = 1 : N,\ b_i = f(a_i),$$

which transforms an input sequence $a_1, \dots, a_N$ to an output sequence $b_1, \dots, b_N$ in a componentwise fashion. If a processor performs sufficiently many of the corresponding Operations (at least their local memory size), then they must perform Loads/Stores to read inputs a_i (which cannot all initially be in their local memory) or write outputs b_i (which cannot be overwritten by definition of implementation). From the perspective of our lower-bound analysis, injectivity is the worst-case scenario for minimizing communication. We interpret the pr

Consider the HBL datum $\mathcal{G} = (\mathbb{Z}^1, (\mathbb{Z}^1), (\phi_1 \colon x \mapsto x))$. We interpret the preceding code as a family of algorithms whose parameters satisfy

$$V = \{a_i, b_i \mid i \in [N]\}, \quad F = \{f_i \mid i \in [N]\}, \quad x \colon (f_i, 1) \mapsto a_i, \quad y \colon (f_i, 1) \mapsto b_i. \tag{5.5}$$

Such algorithms admit the HBL interpretation

$$m = 2, \quad r = 1, \quad r_1 = r_2 = 1, \quad K = F, \quad Z(f_i) = i,$$
$$\phi_1(i) = \phi_2(i) = i, \quad A_1(i) = a_i, \quad A_2(i) = b_i$$

This induces the HBL datum $\mathcal{G} = (\mathbb{Z}^1, (\mathbb{Z}^1)^2, (x \mapsto x)^2)$, and we confirm that both ϕ_1 and ϕ_2 are injections (in particular, identity functions).

If an algorithm admits an HBL interpretation with $\sigma(\mathcal{G}) = 1$, it means that there is a set of disjoint input- and output-paths that connect at least one input or output argument variable of every $f \in K$. Conversely, if some nonempty $K \subseteq F$ has this property, then we can record a (distinct) $v(f) \in I \cup O$ for each $f \in K$ and define an HBL interpretation with $m = r = 1$, $Z(K) = \{1, \dots, |K|\}$, $\phi_1 = \mathrm{id}_{\mathbb{Z}}$, and $A_1(i) = v(Z^{-1}(i))$ for each $i \in \{1, \dots, |K|\}$; we observe that $\sigma((\mathbb{Z}, \mathbb{Z}, \mathrm{id}_{\mathbb{Z}})) = 1$.

From an asymptotic standpoint, we are done the moment we identify an injective homomorphism ϕ_j. Picking $E_N = \{0, \dots, N-1\}$, we have that $|\phi_1(E_N)| = N$ and $|E_N| = N = N^{\sigma(\mathcal{G})}$. In more general injective HBL data, we can pick E_N to be any size-N tile in $\mathbb{Z}^r$, with obtaining similar results (possibly) with constants depending on m.

We will see two more examples of injectivity in Sections 5.5.1 and 5.5.2.

5.5 Product Case

Our main class of examples is called the *product case*, introduced in Lemma 4.30 (in Section 4.4.5). These are algorithms with HBL interpretations inducing HBL data $\mathcal{G}$ such that $\mathcal{P}(\mathcal{G})$ is defined by a simpler set of inequalities. We will consider applying Lemma 4.30 using the standard coordinate $\mathbb{Z}$-basis of $\mathbb{Z}^r$: in this case, each ϕ_j "forgets" a subset of the coordinates $i_1, \dots, i_r$. Consider, for example, the HBL datum

$$\mathcal{G} = (\mathbb{Z}^r, \{\mathbb{Z}^{r-1}\}^m, (\phi_j \colon (i_1, \dots, i_r) \mapsto (i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_r))_j).$$

This is an HBL interpretation for the r-dimensional diamond DAG example in Section 3.1.5, revisited below in Section 5.5.5. We have found a number of practical applications of the product case, which we explore below.

Corollary 5.1. *Consider an HBL datum $\mathcal{G} = (\mathbb{Z}^r, (\mathbb{Z}^{r_j})_j, (\phi_j)_j)$, where each $\phi_j(i_1, \ldots, i_r) = (i_{\pi_j(1)}, \ldots, i_{\pi_j(r_j)})$ for some $\pi_j \colon \{1, \ldots, r_j\} \to \{1, \ldots, r\}$. Given $\mathcal{H} = \{\langle e_1 \rangle, \ldots, \langle e_r \rangle\}$ and any optimal solution x to the linear program* (5.3)*, the conclusion of Theorem 5.1 holds with $\mu = m^{\max\{1/x_i | x_i > 0\}}$, $c = (2m)^{-r}$, and $\tau = \sigma(\mathcal{G}, \mathcal{H})$*

Proof. To specialize Theorem 5.1 in this manner, all we need to show is that the constants $A_{i,j}$ appearing in the proof can all be made equal to one. To see this, we return to the proof of Lemma 4.48 (Section 4.6) where these constants originated, and exploit the fact that the matrix for each ϕ_j is 0/1-valued with at most one nonzero in any row. □

In the following, we go through a number of examples of the product case. We will use the following notation about the linear programs related to HBL data. For any HBL datum $\mathcal{G}$ and any subset $\mathcal{H}$ of the finite-rank subgroups of G, let $y_H = \operatorname{rank}(H)$ and $\Delta_{H,j} = \operatorname{rank}(\phi_j(H))$, and assemble $y = (y_H)_H$ and $\Delta = (\Delta_{H,j})_{H,j}$, where y is a column vector and Δ is an m-column matrix both with (possibly infinitely many) rows indexed by $\mathcal{H}$. We also let 1_m denote the column vector of all ones of length m. In this notation, the primal and dual linear programs (5.2) (5.3) are the following.

$$\begin{array}{cc} \text{Primal linear program} & \text{Dual linear program} \\ \min_{s \in [0,\infty)^m} 1_m^T s \quad \text{s.t.} \quad \Delta s \geq y & \max_{x \in [0,\infty)^{\mathcal{H}}} y^T x \quad \text{s.t.} \quad \Delta^T x \leq 1_m. \end{array}$$

In the present (product case) setting, $G = \mathbb{Z}^r$ and $\mathcal{H} = \{\langle e_1 \rangle, \ldots, \langle e_r \rangle\}$, so $|\mathcal{H}| = r$ and $y = 1_r$, the column vector of all ones of length r. Moreover, Δ is 0/1-valued: $\Delta_{i,j} = 1$ if $e_i \in \ker(\phi_j)$ and $\Delta_{i,j} = 0$ otherwise.

$$\begin{array}{cc} \text{Primal linear program (product case)} & \text{Dual linear program (product case)} \\ \min_{s \in [0,\infty)^m} 1_m^T s \quad \text{s.t.} \quad \Delta s \geq 1_r & \max_{x \in [0,\infty)^r} 1_r^T x \quad \text{s.t.} \quad \Delta^T x \leq 1_m. \end{array}$$

We give a number of examples (Sections 5.5.1-5.5.5) of the product case, including several from our earlier work [15]. Many of these examples are related to 'independent evaluations' studied by Hong-Kung [21, Section 6] and expanded in subsequent works [1, 33, 22, 5]. That is, communication lower bounds and, in many cases, matching upper bounds, are already known for several of these examples. Our contribution is to generalize this to more general loop nests.

5.5.1 Dot Product

We model computing sums-of-products, expressed mathematically as $\sum_i a_i b_i$, or in pseudocode as

$$\begin{aligned} &\textbf{for } i = 1 : N, \\ &\qquad s = s + a_i * b_i, \end{aligned}$$

by a family of algorithms whose parameters satisfy

$$\begin{gathered} I = \{a_i, b_i \mid i \in [N]\}, \quad O = \{t\}, \quad R = V \setminus I \supseteq \{s_i \mid i \in [N]\}, \quad F \supseteq \{z_i \mid i \in [N]\} \\ \{a_i, b_i\} \subseteq \operatorname{im}(x(z_i, \cdot)) \subseteq \{a_i, b_i\} \cup (R \setminus \{s_i\}), \quad \{s_i\} \subseteq \operatorname{im}(y(z_i, \cdot)) \subseteq R \setminus \{s_k \mid i \neq k \in [N]\}. \end{gathered} \tag{5.6}$$

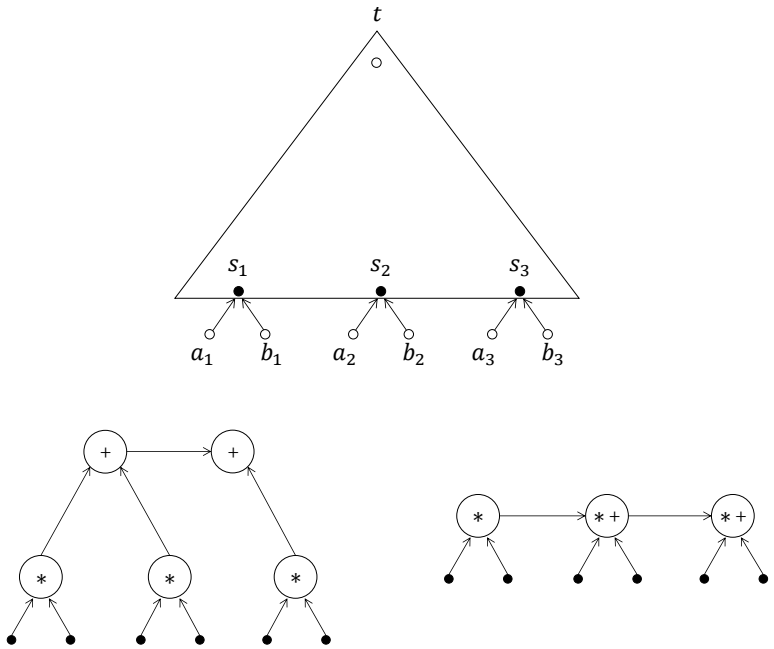


Figure 5.1: Dot product example (Section 5.5.1) in the case $N = 3$: general form (top); using binary $+, *$ operations (bottom-left); using "fused multiply-add" $*+$ operation (bottom-right).

(Informally, the constraints on $\mathrm{im}(x(z_i, \cdot))$ and $\mathrm{im}(y(z_i, \cdot))$ mean that additions can be fused with multiplications, but multiplications cannot be fused.) The variable $t \in V$ is any element of R (perhaps one of $s_1, \ldots, s_N$). Algorithms satisfying (5.6) induce DAGs of the form shown in Figure 5.1. Note that the summation is modeled by a more general reduction, introduced in Section 3.1.6. In particular, note that the operations z_i are not necessarily independent; the important property is that they each depend on a distinct pair $\{a_i, b_i\}$ of inputs.

Algorithms satisfying (5.6) admit the following HBL interpretation.

$$\begin{gathered} m = 3, \quad r = 1, \quad (r_1, r_2, r_3) = (1, 1, 0) \\ K = \{z_i \mid i \in [N]\}, \quad Z(z_i) = i \\ \phi_1 = \phi_2 = \mathrm{id}_{\mathbb{Z}}, \quad \phi_3 \colon \mathbb{Z} \to \{0\} \\ A_1(i) = a_i, \quad A_2(i) = b_i, \quad A_3(i) = t \end{gathered}$$

We confirm that $\bigcap_j \ker(\phi_j) = \{0\}$. We also recognize that ϕ_1 and ϕ_2 are both injections, so the conclusions in Section 5.4 apply. However, we can arrive at the same conclusions by exploiting the machinery of the product case. That is, we observe that

$$\Delta = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix},$$

so the LPs have value $\sigma = 1$, and the dual solution $x = (x_1) = 1$.

Although blocking cannot demonstrate an asymptotic benefit (this is the injective case), we can apply Corollary 5.1 to obtain the tiles

$$E_M = \{0, \dots, \lfloor M^{x_1}/m \rfloor - 1\} = \{0, \dots, \lfloor M/3 \rfloor - 1\},$$

supposing $M \geq \mu = 3$. A blocked version of the preceding code, where the blocking factor $b = \lfloor M/3 \rfloor$ is assumed to divide N, is given by

$$\begin{array}{ll} \textbf{for } j = 1 : N/b, & \\ \quad \textbf{for } k = 1 : b, & \text{// reindexing } i = b \cdot (j-1) + k \\ \quad\quad s = s + a_i * b_i. & \end{array}$$

5.5.2 Matrix-Vector Multiplication

Next we model premultiplying a length-N column vector by a N-by-N matrix, expressed componentwise mathematically as $\sum_{i_2} a_{i_1,i_2} b_{i_2}$, or in pseudocode as

$$\textbf{for } i_1 = 1 : N, \quad \textbf{for } i_2 = 1 : N, \\ s_{i_1} = s_{i_1} + a_{i_1,i_2} * b_{i_2},$$

by a family of algorithms whose parameters satisfy

$$\begin{gathered} I = \{a_{i_1,i_2}, b_{i_2} \mid (i_1, i_2) \in [N]^2\}, \quad O = \{t_{i_1} \mid i_1 \in [N]\} \\ R = V \setminus I \supseteq \{s_{i_1,i_2} \mid (i_1, i_2) \in [N]^2\}, \quad F \supseteq \{z_{i_1,i_2} \mid (i_1, i_2) \in [N]^2\} \\ \{a_{i_1,i_2}, b_{i_2}\} \subseteq \operatorname{im}(x(z_{i_1,i_2}, \cdot)) \subseteq \{a_{i_1,i_2}, b_{i_1}\} \cup (R \setminus \{s_{i_1,i_2}\}) \\ \{s_{i_1,i_2}\} \subseteq \operatorname{im}(y(z_{i_1,i_2}, \cdot)) \subseteq R \setminus \{s_{k,l} \mid (i_1, i_2) \neq (k, l) \in [N]^2\}. \end{gathered} \tag{5.7}$$

(The constraints on x and y are analogous to those in the dot product example.) The variables $t_1, \dots, t_M \in V$ can be any N distinct elements of R (perhaps s_{i_1,i_2}). Algorithms satisfying (5.7) induce DAGs of the form shown in Figure 5.2. Each element of the output vector is computed by a dot product (see Section 5.5.1) of a distinct matrix row and the input vector.

Algorithms satisfying (5.7) admit the following HBL interpretation.

$$\begin{gathered} m = 3, \quad r = 2, \quad (r_1, r_2, r_3) = (2, 1, 0) \\ K = \{z_{i_1,i_2} \mid (i_1, i_2) \in [N]^2\}, \quad Z(z_{i_1,i_2}) = (i_1, i_2) \\ \phi_1(i_1, i_2) = (i_1, i_2), \quad \phi_2(i_1, i_2) = i_2, \quad \phi_3(i_1, i_2) = i_1 \\ A_1(i_1, i_2) = a_{i_1,i_2}, \quad A_2(i_2) = b_{i_2}, \quad A_3(i_1) = t_{i_1}. \end{gathered}$$

We confirm that $\bigcap_j \ker(\phi_j) = \{0\}$. We also recognize that ϕ_1 is an injection, so as in the previous example, the conclusions in Section 5.4 apply, but we can arrive at the same conclusions by exploiting the machinery of the product case. That is, we observe that

$$\Delta = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix},$$

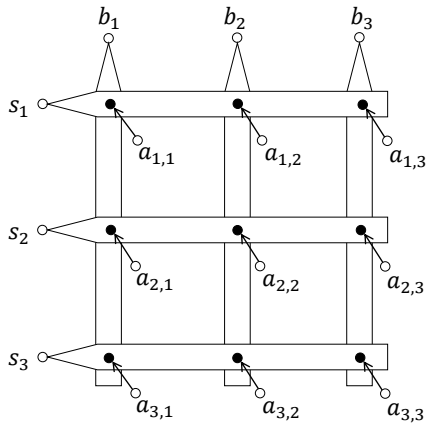


Figure 5.2: Matrix-vector multiplication example (Section 5.5.2) in the case $N = 3$. Pencils are meant to suggest reductions and expansions (see Section 3.1.6).

so the LPs have value $\sigma = 1$, and the dual solution $x = (x_1, x_2) = (\alpha, 1-\alpha)$ for any $\alpha \in [0, 1]$.

Although blocking cannot demonstrate an asymptotic benefit (again, this is the injective case), we can apply Corollary 5.1 to obtain the tiles

$$\begin{aligned} E_M &= \{0, \dots, \lfloor M^{x_1}/m \rfloor - 1\} \times \{0, \dots, \lfloor M^{x_2}/m \rfloor - 1\} \\ &= \{0, \dots, \lfloor M^{\alpha}/3 \rfloor - 1\} \times \{0, \dots, \lfloor M^{1-\alpha}/3 \rfloor - 1\}, \end{aligned}$$

supposing $M \geq \mu = 3$. A blocked version of the preceding code, where the blocking factors

$$b_1 = \begin{cases} \lfloor M^{\alpha}/3 \rfloor & \alpha > 0 \\ 1 & \text{otherwise,} \end{cases} \qquad b_2 = \begin{cases} \lfloor M^{1-\alpha}/3 \rfloor & \alpha < 1 \\ 1 & \text{otherwise,} \end{cases}$$

are both assumed to divide N, is given by

$$\begin{aligned} &\textbf{for } j_1 = 1 : N/b_1, \quad \textbf{for } j_2 = 1 : N/b_2, \\ &\quad \textbf{for } k_1 = 1 : b_1, \quad \textbf{for } k_2 = 1 : b_2, \\ &\quad\quad // \text{ reindexing } (i_1, i_2) = b \cdot (j_1 - 1, j_2 - 1) + (k_1, k_2) \\ &\quad\quad s_{i_1} = s_{i_1} + a_{i_1,i_2} * b_{i_2}. \end{aligned}$$

5.5.3 Matrix-Matrix Multiplication

Next we model multiplying N-by-N matrices, componentwise mathematically as $\sum_{i_3} a_{i_1,i_3} b_{i_3,i_2}$, or in pseudocode as

$$\begin{aligned} &\textbf{for } i_1 = 1 : N, \quad \textbf{for } i_2 = 1 : N, \quad \textbf{for } i_3 = 1 : N, \\ &\quad\quad s_{i_1,i_2} = s_{i_1,i_2} + a_{i_1,i_3} * b_{i_3,i_2}, \end{aligned}$$

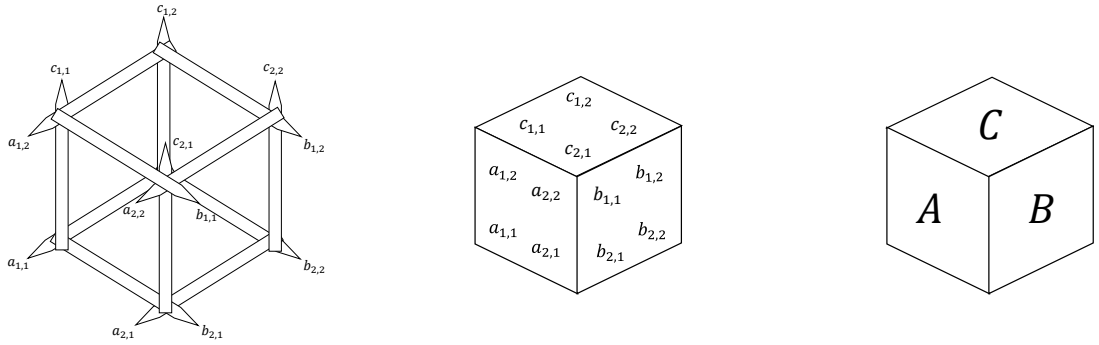


Figure 5.3: Matrix-matrix multiplication example (Section 5.5.3) in the case $N = 3$.

by a family of algorithms whose parameters satisfy

$$\begin{gathered} I = \{a_{i_1,i_3}, b_{i_3,i_2} \mid (i_1,i_2,i_3) \in [N]^3\}, \quad O = \{t_{i_1,i_2} \mid (i_1,i_2) \in [N]^2\} \\ R = V \setminus I \supseteq \{s_{i_1,i_2,i_3} \mid (i_1,i_2,i_3) \in [N]^3\}, \quad F \supseteq \{z_{i_1,i_2,i_3} \mid (i_1,i_2,i_3) \in [N]^3\} \\ \{a_{i_1,i_3}, b_{i_3,i_2}\} \subseteq \mathrm{im}(x(z_{i_1,i_2,i_3}, \cdot)) \subseteq \{a_{i_1,i_3}, b_{i_3,i_2}\} \cup (R \setminus \{s_{i_1,i_2,i_3}\}) \\ \{s_{i_1,i_2,i_3}\} \subseteq \mathrm{im}(y(z_{i_1,i_2,i_3}, \cdot)) \subseteq R \setminus \{s_{k,l,m} \mid (i_1,i_2,i_3) \neq (k,l,m) \in [N]^3\}. \end{gathered} \tag{5.8}$$

The variables $t_{i_1,i_2} \in V$ can be any N^2 distinct elements of R (perhaps s_{i_1,i_2,i_3}). Algorithms satisfying (5.8) induce DAGs of the form shown in Figure 5.3. Each element of the output matrix is computed by a dot product (see Section 5.5.1) of a distinct matrix row and matrix column.

Algorithms satisfying (5.8) admit the following HBL interpretation.

$$\begin{gathered} m = 3, \quad r = 3, \quad (r_1, r_2, r_3) = (2,2,2) \\ K = \{z_{i_1,i_2,i_3} \mid (i_1,i_2,i_3) \in [N]^3\}, \quad Z(z_{i_1,i_2,i_3}) = (i_1,i_2,i_3) \\ \phi_1(i_1,i_2,i_3) = (i_1,i_3), \quad \phi_2(i_1,i_2,i_3) = (i_3,i_2), \quad \phi_3(i_1,i_2,i_3) = (i_1,i_2) \\ A_1(i_1,i_3) = a_{i_1,i_3}, \quad A_2(i_3,i_2) = b_{i_3,i_2}, \quad A_3(i_1,i_2) = t_{i_1,i_2}. \end{gathered}$$

We confirm that $\bigcap_j \ker(\phi_j) = \{0\}$. We have the linear system

$$\Delta = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix},$$

so the LPs have value $\sigma = 3/2$, and the dual solution $x = (x_1, x_2, x_3) = (1/2, 1/2, 1/2)$.

In this case, we can apply Corollary 5.1 to obtain the tiles

$$E_M = \bigtimes_{i \in [r]} \{0, \ldots, \lfloor M^{x_i}/m \rfloor - 1\} = \{0, \ldots, \lfloor M^{1/2}/3 \rfloor - 1\}$$

supposing $M \geq \mu = 3$. A blocked version of the preceding code, where the blocking factor $b = \lfloor M^{1/2}/3 \rfloor$ is assumed to divide N, is given by

$$\begin{aligned}
&\textbf{for } j_1 = 1 : N/b, \quad \textbf{for } j_2 = 1 : N/b, \quad \textbf{for } j_3 = 1 : N/b, \\
&\quad \textbf{for } k_1 = 1 : b, \quad \textbf{for } k_2 = 1 : b, \quad \textbf{for } k_3 = 1 : b \\
&\qquad \text{// reindexing } (i_1, i_2, i_3) = b \cdot (j_1 - 1, j_2 - 1, j_3 - 1) + (k_1, k_2, k_3) \\
&\qquad s_{i_1,i_2} = s_{i_1,i_2} + a_{i_1,i_3} * b_{i_3,i_2}.
\end{aligned}$$

This example extends to multiplication of rectangular and sparse matrices, as well as tensor contractions, which can be recast as matrix multiplication (see, e.g., [3, Section 2.6.1]).

5.5.4 Multiple-Tensor Contractions

Multiplying multiple matrices or tensors is usually done more efficiently by exploiting associativity, i.e., inserting parentheses and multiplying pairs of matrices/tensors; in this case, we can apply the preceding example (matrix multiplication, Section 5.5.3) to each pair, obtaining a communication lower bound based on $\sigma = 3/2$. Now we consider two examples where this natural optimization is not exploited.

In the first example, we suppose we obtain an HBL datum with $m = r$ and homomorphisms

$$\phi_j(i_1, \ldots, i_r) = (i_{j \bmod r}, i_{(j+1) \bmod r}, \ldots, i_{(j+k-1) \bmod r})$$

given some $k \in \{1, \ldots, r\}$. We can solve the corresponding linear program to see that $\sigma = r/k$. The case $r = 3$ and $k = 2$ is equivalent to the matrix multiplication example.

In the second example, meant to model a "matricized tensor times Khatri-Rao product operation" (given by G. Ballard in personal communication, May 14, 2015), $m = r \geq 2$ and

$$\begin{aligned}
\phi_1(i_1, \ldots, i_r) &= (i_1, \ldots, i_{r-1}) \\
\phi_2(i_1, \ldots, i_r) &= (i_2, i_r) \\
&\vdots \\
\phi_m(i_1, \ldots, i_r) &= (i_{r-1}, i_r);
\end{aligned}$$

we can solve the corresponding linear program to see that $\sigma = 2 - 1/(r-1)$. Note that when $r \geq 3$, $\sigma > 3/2$.

5.5.5 N-body Simulation

In our last example, we return to the diamond DAG examples introduced in Section 3.1.5. We have that $m = r$ and for each j,

$$\phi_j(i_1, \ldots, i_r) = (i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_r).$$

We find that $\sigma = r$, and that $x = (x_j)_j = (1, \ldots, 1)$ is the optimal solution of the dual linear program, indicating that the natural approach of tiling these computations with cubes is optimal.

Chapter 6

Conclusion

Motivated by the fact that communication (moving data) frequently dominates the costs of computation (performing operations on that data), we have studied the communication costs within a class of nested-loop programs.

We started in Chapter 2 by modeling communication in an abstract machine model, and deriving communication lower bounds. The main contribution of Chapter 2 is a new shared-memory parallel variant of Hong-Kung's sequential communication lower bounds approach [21], based on the notions of input-/output-path cutsize.

In Chapter 3, we demonstrated how algorithms can model a class of nested-loop programs with group-theoretic interpretations. The main contribution of Chapter 3 is the notion of an HBL interpretation, which generalizes previous work.

In Chapter 4, we developed a deeper theory needed show that algorithms with HBL interpretations have particular input-/output-path cutsize properties. The main contribution of Chapter 4 is deriving sharper constants in certain HBL-type inequalities, which in turn give sharper constants in our lower bounds.

Finally, in Chapter 5, we studied the attainability of the lower bounds, showing that they are attainable within constant factors in many practical cases. The main contribution of Chapter 5 is a framework for finding communication-optimal implementations by tiling.

There are a number of different directions for future work.

First, there are a number of possible improvements to our approach for computing the constraints for HBL-type inequalities in Section 4.7. For example, we have identified simpler approaches when all the homomorphisms have rank-1 or corank-1. Additionally, we believe that a simplification proposed by Valdimarsson [34] in the continuum setting extends to the discrete setting; this would simplify computing the constraints in many more cases, for example, whenever there are at most three homomorphisms, but it would also simplify our algorithm for the general case.

Second, there are number of ways to tighten the communication lower bounds. We have suggested changing the segment length as one option, as well as exploiting knowledge about aliasing arrays, but there are a wide variety of options when it comes to picking an HBL interpretation. It would be interesting to characterize different HBL interpretations that a single algorithm admits.

Third, there are other ways to model communication cost other than the ones proposed here. Our analysis was mostly in terms of per-processor communication costs, and it remains

future work to develop tighter bounds on critical-path communication cost. This will involve a more complicated proof approach, since we largely ignored the influence of processors on each other. Another important aspect of communication is overlap: there is often concurrency in the memory system (network, etc.) which is exploited by overlapping the cost of moving single values by moving data in blocks/messages. Yet another issue is distributed versus shared-memory. In this work we model shared-memory parallel machines but in our previous work we modeled distributed-memory machines. We can model distributed memory in the present model by simulating message-passing in shared-memory; it remains to demonstrate this extension, and show how tiling changes to address it.

Fourth, there is much to be done from an upper-bounds standpoint. We would like to give an automated approach to transform a nested-loop program into a communication-optimal version. There are a number of complications both theoretically (algorithm dependences) and practically (program representations), and this remains future work.

References

[1] A. Aggarwal, A. Chandra, and M. Snir. "Communication complexity of PRAMs". In: *Theoretical Computer Science* 71 (1 Mar. 1990), pp. 3–28. ISSN: 0304-3975. DOI: 10.1016/0304-3975(90)90188-N. URL: http://portal.acm.org/citation.cfm?id=77831.77836.

[2] G. Ballard. "Avoiding Communication in Dense Linear Algebra". PhD thesis. EECS Department, University of California, Berkeley, Aug. 2013. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-151.html.

[3] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz. "Communication lower bounds and optimal algorithms for numerical linear algebra". In: *Acta Numerica* 23 (2014), pp. 1–155.

[4] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. "Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds". In: *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*. ACM. 2012, pp. 77–79.

[5] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. "Minimizing communication in numerical linear algebra". In: *SIAM Journal on Matrix Analysis and Applications* 32.3 (2011), pp. 866–901.

[6] F. Barthe. "The Brunn-Minkowski theorem and related geometric and functional inequalities". In: *International Congress of Mathematicians*. Vol. 2. 2006, pp. 1529–1546.

[7] W. Beckner. "Inequalities in Fourier analysis". In: *Annals of Mathematics* (1975), pp. 159–182.

[8] C. Bennett and R. Sharpley. *Interpolation of Operators*. Vol. 129. Pure and Applied Mathematics. Academic Press, 1988.

[9] J. Bennett, A. Carbery, M. Christ, and T. Tao. "Finite bounds for Hölder-Brascamp-Lieb multilinear inequalities". In: *Mathematical Research Letters* 17.4 (2010), pp. 647–666.

[10] J. Bennett, A. Carbery, M. Christ, and T. Tao. "The Brascamp-Lieb inequalities: finiteness, structure, and extremals". In: *Geometric and Functional Analysis* 17.5 (Jan. 1, 2008), pp. 1343–1415.

[11] G. Bilardi, A. Pietracaprina, and P. D'Alberto. "On the space and access complexity of computation DAGs". In: *Graph-Theoretic Concepts in Computer Science. 26th International Workshop, WG 2000 Konstanz, Germany, June 15-17, 2000 Proceedings*. 26th International Workshop on Graph-Theoretic Concepts in Computer Science. WG 2000 (Konstanz, Germany, June 15–17, 2000). Ed. by U. Brandes and D. Wagner. Vol. 1928. Lecture Notes in Computer Science. Algorithms and Data Structures Group of the Department of Computer and Information Science, University of Konstanz. Springer-Verlag, 2000, pp. 47–58.

[12] H. J. Brascamp and E. H. Lieb. "Best constants in Young's inequality, its converse, and its generalization to more than three functions". In: *Advances in Mathematics* 20.2 (1976), pp. 151–173.

[13] E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, and H. V. Simhadri. *Write-Avoiding Algorithms*. Tech. rep. UCB/EECS-2015-163. EECS Department, University of California, Berkeley, June 2015. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-163.html`.

[14] M. Christ. "The optimal constants in Hölder-Brascamp-Lieb inequalities for discrete Abelian groups". In: (July 31, 2013). arXiv:1307.8442v1.

[15] M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick. *Communication lower bounds and optimal algorithms for programs that reference arrays (part 1)*. Tech. rep. UCB/EECS-2013-61. University of California, Berkeley, Dept. of Electrical Engineering and Computer Science, May 14, 2013.

[16] M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick. "Communication lower bounds for programs with affine dependences". Talk at SIAM Conference on Parallel Processing for Scientific Computing, Portland, Oregon, USA. Feb. 19, 2014. URL: `http://www.cs.berkeley.edu/~knight/cdksy_PP14_slides.pdf`.

[17] G. Collins. "Quantifier elimination for real closed fields by cylindrical algebraic decomposition". In: *Automata Theory and Formal Languages: 2nd GI Conference*. Springer. 1975, pp. 134–183.

[18] V. Elango, F. Rastello, L. Pouchet, J. Ramanujam, and P. Sadayappan. "On Characterizing the Data Access Complexity of Programs". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*. 2015, pp. 567–580. DOI: `10.1145/2676726.2677010`. URL: `http://doi.acm.org/10.1145/2676726.2677010`.

[19] P. van Emde Boas and J. van Leeuwen. "Move rules and trade-offs in the pebble game". English. In: *Theoretical Computer Science 4th GI Conference*. Ed. by K. Weihrauch. Vol. 67. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1979, pp. 101–112. ISBN: 978-3-540-09118-9. DOI: `10.1007/3-540-09118-1_12`. URL: `http://dx.doi.org/10.1007/3-540-09118-1_12`.

[20] O. Hölder. "Über einen Mittelwertssatz". In: *Nachr. Akad. Wiss. Göttingen Math.-Phys. Kl.* (1889), pp. 38–47.

[21] J.-W. Hong and H. T. Kung. "I/O complexity: the red-blue pebble game". In: *Proceedings of the 13th annual ACM symposium on theory of computing*. STOC '81. (Milwaukee, Wisconsin, USA, 1981). ACM special interest group on algorithms and computation theory (SIGACT). New York, New York, USA: ACM, 1981, pp. 326–333.

[22] D. Irony, S. Toledo, and A. Tiskin. "Communication lower bounds for distributed-memory matrix multiplication". In: *Journal of Parallel and Distributed Computing* 64.9 (2004), pp. 1017–1026.

[23] S. Lang. *Algebra*. 3rd. Vol. 211. Graduate Texts in Mathematics. Springer, 2002. ISBN: 9780387953854.

[24] L. Loomis and H. Whitney. "An inequality related to the isoperimetric inequality". In: *Bulletin of the American Mathematical Society* 55 (1949), pp. 961–962.

[25] B. Lowery and J. Langou. "Improving Communication Lower Bounds for Matrix-Matrix Multiplication". Talk at the 9th Scheduling for Large Scale Systems Workshop, Lyon, France. July 2, 2014. URL: `http://scheduling2014.sciencesconf.org/conference/scheduling2014/pages/julien_langou.pdf`.

[26] L. Maligranda. "Why Hölder's inequality should be called Rogers' inequality". In: *Math. Inequal. Appl* 1.1 (1998), pp. 69–83.

[27] Y. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

[28] B. Poonen. "Hilbert's Tenth Problem over rings of number-theoretic interest". Notes from Arizona Winter School, "Logic and Number Theory", March 15-19, 2003, Tuscon, AZ. 2003.

[29] L. J. Rogers. "An extension of a certain theorem in inequalities". In: *Messenger of Math* 17 (1888), pp. 145–150.

[30] W. Rudin. *Real and Complex Analysis*. Tata McGraw-Hill Education, 1987.

[31] J. E. Savage. "Extending the Hong-Kung model to memory hierarchies". In: *Computing and Combinatorics. First Annual International Conference, COCOON '95 Xi'an, China, August 24-26, 1995 Proceedings*. First Annual International Conference on Computing and Combinatorics. COCOON '95 (Xi'an, China, Aug. 24–26, 1995). Ed. by D.-Z. Du and M. Li. Vol. 959. Lecture Notes in Computer Science. Springer, 1995, pp. 270–281.

[32] A. Tarski. *A decision method for elementary algebra and geometry*. Tech. rep. R-109. RAND Corporation, 1951.

[33] A. Tiskin. "The design and analysis of bulk-synchronous parallel algorithms". PhD thesis. Oxford Univ., 1998.

[34] S. I. Valdimarsson. "Optimizers for the Brascamp-Lieb inequality". In: *Israel Journal of Mathematics* 168.1 (Nov. 1, 2008), pp. 253–274.

[35] S. I. Valdimarsson. "The Brascamp-Lieb polyhedron". In: *Canadian Journal of Mathematics* 62.4 (2010), pp. 870–888.

[36] W. Young. "On the multiplication of successions of Fourier constants". In: *Proc. Roy. Soc. of London, Ser. A* (1912), pp. 331–339.