

DOMAIN DECOMPOSITION, INTEGRATION, AND
INCLUSION-EXCLUSION
(Thesis format: Monograph)

by

Mike Ghesquiere

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Mike Ghesquiere 2015

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....
Dr. S. M. Watt

Examiners:

Supervisory Committee:

The thesis by

Mike Ghesquiere

entitled:

Domain Decomposition, Integration, and Inclusion-Exclusion

is accepted in partial fulfillment of the
requirements for the degree of
Masters of Science

.....
Date

.....
Chair of the Thesis Examination Board

Abstract

Mathematic notation has been dominated by sets and, when repeated elements are required, sequences generally make an appearance. Historical inertia has caused these structures to be used in many situations where they are ill-suited often evidenced by phrases like "without loss of generality", "up to ordering of terms", "up to a sign". However, by tackling these problems instead with more apt data structures, we can eliminate some of these stipulations and more formally reduce symmetric cases in reasoning. In particular, this thesis will deal with *hybrid sets* (that is, signed multisets), as well *hybrid functions* (that is, functions with hybrid sets for their domain) with applications in piecewise functions, integration on manifolds, (...). More than just an aesthetic change, by allowing negative multiplicity (even if it would not make physical sense), we may symbolically manipulate structures in ways that might otherwise be cumbersome or inefficient.

Keywords: Hybrid set, Signed multiset, Integration on chains, Inclusion-Exclusion, (...)

Contents

Certificate of Examination	ii
Acknowledgements	iii
Abstract	iii
List of Figures	vi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Related Work	3
1.4 Thesis Outline	3
2 Hybrid Set Theory	5
2.1 Hybrid Sets	7
2.1.1 Example: <i>Rational Arithmetic</i>	9
2.1.2 Example: <i>Rational Polynomials</i>	10
2.2 Hybrid Relations	11
2.2.1 Example: <i>Maximum Flow</i>	12
2.3 Hybrid Functions	13
2.4 Hybrid Function Fold	16
2.4.1 Example: <i>Sign function</i>	17
2.4.2 Example: <i>Piecewise functions on generalized partitions</i>	18
2.5 Pseudo-functions	22
2.5.1 Example: <i>Piecewise functions revisited</i>	23
3 Symbolic Block Linear Algebra	26
3.1 Oriented Intervals	28
3.2 Vector Addition	31
3.3 Higher dimension intervals	33
3.4 Matrix Addition	36
3.4.1 Example: <i>Evaluation at points</i>	38
3.4.2 Larger block matrices	38
3.5 Matrix Multiplication	39
3.5.1 Larger block matrices	43

4	Integration	44
4.1	The Lebesgue Integral	46
4.2	Riemann Integral on n -cubes	48
4.3	Boundary Operator	51
4.3.1	Example: <i>Boundary of a 1-rectangle</i>	52
4.3.2	Example: <i>Boundary of a 3-rectangle</i>	53
4.4	Chains	54
4.4.1	Example: <i>Boundary of a boundary (of a 2-cube)</i>	55
4.5	Differential Forms	58
4.6	Stokes' Theorem	61
4.7	Example	61
4.8	Manifolds	61
4.8.1	Singular Cubes	61
4.8.2	Manifold Integration	63
5	Convolution	64
5.1	Convolution of Piecewise Functions	66
5.2	Hybrid Function Convolution	69
5.2.1	Example: <i>Hybrid Convolution</i>	70
5.3	Infinite Intervals	72
5.4	Discrete Convolution	73
5.5	Implementation	75
6	Conclusions	79
	Bibliography	81
A	Convolution with Infinite End-Points	84
	Curriculum Vitae	92

List of Figures

2.1	Piece-wise Rational Function	24
3.1	Possible block overlaps of 2×2 block matrices.	28
3.2	Cartesian product of two 1-rectangles	34
4.1	Approximations using simple functions	47
4.2	Riemann Integral	49
4.3	Unit cube with boundary	53
4.4	Orientations of 2-rectangles	54
4.5	Boundary of a boundary (of a 2-cube)	56
5.1	Convolution of box signal with itself	64
5.2	Gaussian Blurring	65
5.3	Convolution of “one-piece” functions	67

Chapter 1

Introduction

1.1 Motivation

Some data structures in mathematics are more well-loved than others and none more than Cantor's set. The very foundations of mathematics lie in set theory: numbers are defined in terms of sets as are ordered tuples which in turn lead to relations, functions, sequences and from there branches into countless other structures. But this trunk typically omits a satisfactory treatment of several *generalized sets*. For example, it is difficult to begin speaking about *hybrid sets* without immediately punctuating, "that is, multisets with negative multiplicity". It is a statement of progress that multisets have even entered into (relatively) common mathematical parlance.

Still, sequences need no introduction and are generally relied on when a structure allowing repeated elements is needed. The addition of an ordering is often not even needed but "surely it can't hurt?" Consider the *Fundamental Theorem of Arithmetic*:

"Every positive integer, except 1, is a product of primes." ... "The standard form of n is unique; apart from rearrangement of factors, n can be expressed as a product of primes in one way only." (Hardy and Wright 1979, p.2-3)

By recognizing the possibility of rearranging factors, the authors implicitly define type the "product of primes" as a sequence. But for iterated commutative operators (e.g. \sum, \prod, \cap, \cup), the order of terms is irrelevant. So then, why order terms to begin with? Reisig [17] uses

multisets to define relation nets where “... several individuals of some sort do not have to be distinguished” and furthermore “One should not be forced to distinguish individuals if one doesn’t wish to. This would lead to overspecification”. The same applies here. Secondly, iterated operators over an empty set is simply the respective identity ($\prod_{x \in \emptyset} x = 1$), and so 1 is a product of primes. Despite the empty sequence being just as well-defined as the empty set; it tends to be treated as an aberrant case. Some definitions even disregard the singleton sequence to say, “is prime or the product of primes”.

Stripped of these qualifications we are left with simply:

“Every positive integer is the product of a unique multiset of primes.”

Although the definition is equivalent, by using appropriate data structures, the result is more elegant and less case-based reasoning for later uses of the definition. In this spirit that we will graft hybrid sets into areas of mathematics where conventional structures don’t fit as tightly as we’d like.

1.2 Objectives

This thesis will include and extend the work of [8] on hybrid sets and their applications. In particular, integration is a natural application of signed domains that had not been explored from this perspective. Take the identity:

$$\int_a^b f(x) \, dx = - \int_b^a f(x) \, dx \quad (1.1)$$

The domain of integration on the left-side is considered to be the interval $[a, b]$ or in set builder notation, $\{x \in \mathbb{R} \mid a \leq x \leq b\}$. It would follow then that the right hand should have domain $[b, a]$. Under traditional set definitions, this is not well defined. Using hybrid sets allows us to give meaning to an inverted interval. When generalized to higher dimensions, this goal becomes an attempt to unify the *Lebesgue integral* and *integration of forms*. Such a model

would allow for integration of differential forms over subsets of manifolds.

I don't currently have a set of objectives for the Petri net chapter. Another couple lines will go here once I have a better idea what the chapter will look like.

1.3 Related Work

It is difficult to date the origin of multisets. The term was coined by N.G. de Bruijn in correspondence with Donald Knuth, [14] but thought of as a “collection of objects that may or may not be distinguished” is as old as tally marks. In regards to the generalization to *signed* multisets, Hailperin [13] suggests that Boole's 1854 *Laws of Thought* [7] is a treatise of signed multisets. Whether this was Boole's intent is debated [cite]. Sets with negative membership explicitly began to appear in [24] and were formalized under the name Hybrid sets in Blizard's extensive work with generalized sets [5, 6] Although hybrid set and signed multiset are the most common nomenclature, other names appearing in literature include multiset (specifying positive when for unsigned multisets) [17] and integral multiset [25].

Existing explicit applications of hybrid sets are currently limited. Loeb *et al.* [11, 16] use hybrid sets to generalize several combinatoric identities to negative values. Bailey *et al.* [1] and Banâtre *et al.* [2] have also had success with hybrid sets in chemical programming. Representing a solution is represented as a collection of atoms and molecules, negative multiplicities are treated as “antimatter”. For a deeper overview and systemization of generalized sets, see [20, 21]. Finally, Bartoletti [3, 4] and Schmidt [18] also investigate petri and relation nets (to be examined in chapter 5), albeit not from the perspective of hybrid sets.

1.4 Thesis Outline

In chapter 2, the foundations for hybrid sets and functions with hybrid set domains will be laid. Some immediate applications to piecewise functions will be presented. In chapter 3, the algebra of domains will be more deeply explored with generalized partitions and inclusion-exclusion

and present an method for computing the support of a hybrid set. In chapter 4, hybrid functions will be applied towards integration. Starting from foundations we will use hybrid functions to unify integration of forms with integration on subsets of manifolds and prove Stokes' theorem on the new model. Finally in chapter 5, hybrid sets will be applied to Petri and relation nets with the scope of ...

Chapter 2

Hybrid Set Theory

The motivation behind hybrid sets and functions can be traced to wanting a better approach to piecewise functions. Piecewise functions are enormously useful constructions in many areas (...) The perennial example of a piecewise function is $\text{abs} : \mathbb{R} \rightarrow \mathbb{R}_+ \cup \{0\}$ given in the form:

$$\text{abs}(x) = \begin{cases} -x & : x < 0 \\ x & : x \geq 0 \end{cases} \quad (2.1)$$

To evaluate abs for an argument x , one must first determine which sub-function to use. If $x < 0$ then the first case is evaluated and abs will return the result of $x \mapsto -x$. Otherwise, if $x \geq 0$ the second case is evaluated and the result of $x \mapsto x$ is returned. Rather than as a condition, we could just as easily think of “ $x < 0$ ” and “ $x \geq 0$ ” as partitions of the real line. Evaluation then occurs by checking whether $x \in \mathbb{R}_+ \cup \{0\}$ or $x \in \mathbb{R}_-$. In general, a piecewise function f will take the form:

$$f(x) = \begin{cases} f_1(x) & : x \in P_1 \\ f_2(x) & : x \in P_2 \\ \vdots & \vdots \\ f_n(x) & : x \in P_n \end{cases} \quad (2.2)$$

where the set $\{P_i\}$ forms a partition of the domain of f and for each f_i is defined over all of the corresponding P_i . To formalize this we require the ability to restrict a function's domain and join disjoint pieces together.

Definition Given a function $f : X \rightarrow Y$ for any subset of the domain, $Z \subset X$, the *restriction of f to Z* is the function $f|_Z : Z \rightarrow Y$, such that $f|_Z(x) = f(x)$ for all $x \in Z$.

Definition Define \oplus , the *join* of two functions, f and g by:

$$f \oplus g = \begin{cases} f(x) & \text{if } g(x) = \perp \\ g(x) & \text{if } f(x) = \perp \\ \perp & \text{otherwise} \end{cases} \quad (2.3)$$

This allows us to re-write our previous definition of (2.2) as:

$$f = f|_{P_1} \oplus f|_{P_2} \oplus \dots \oplus f|_{P_n} \quad (2.4)$$

But we must be careful as this definition is not associative. Given functions f, g, h each restricted to the sets A, B, C . Then for any element in the intersection $x \in A \cap B \cap C$:

$$((f|_A \oplus g|_B) \oplus h|_C)(x) = h(x) \quad (2.5)$$

But by rearranging the parentheses we find,

$$(f|_A \oplus (g|_B \oplus h|_C))(x) = f(x) \quad (2.6)$$

Other conventions exist, for example *Maple's* `piecewise(cond_1, f_1, cond_2, f_2, ..., cond_n, f_n, f_otherwise)` effectively uses a short-circuted \oplus . Each of the conditionals, `cond_i` are evaluated in order and when one evaluates to true, the corresponding `f_i` is evaluated. Finally if all of `cond_i` evaluate to false then `f_otherwise` is used. This approach simply trades associativity for commutivity.

More importantly, when dealing with multiple piecewise functions, expressions cannot be easily simplified without an explosion in terms. For example, given f, g two piecewise functions $f = (f_1|_{P_1} \oplus f_2|_{P_2} \oplus f_3|_{P_3})$ and $g = (g_1|_{Q_1} \oplus g_2|_{Q_2})$ their sum $(f + g)$ is:

$$\begin{aligned} (f + g) = & (f_1 + g_1)|_{P_1 \cap Q_1} \oplus (f_1 + g_2)|_{P_1 \cap Q_2} \oplus (f_1 + g_3)|_{P_1 \cap Q_3} \\ & (f_2 + g_1)|_{P_2 \cap Q_1} \oplus (f_2 + g_2)|_{P_2 \cap Q_2} \oplus (f_2 + g_3)|_{P_2 \cap Q_3} \end{aligned} \quad (2.7)$$

That is, we first create a common refinement by the pairwise intersection of each P_i with each Q_j . Over each intersection we then take the sum of the respective sub-functions. In general, assuming no degenerate intersections, the sum of an n piece function and m piece function will give a piecewise function with $n \times m$ pieces. To flatten an expression containing multiple piecewise functions we can very quickly find ourselves with an unreasonable number of pieces. If we abandon boolean sets, we can construct common refinements much more efficiently as will be shown.

2.1 Hybrid Sets

We shall consider *hybrid sets*. One can define traditional sets by an indicator function which maps each member element to 1 and each non member to 0. A *multiset* or *bag* extends this by allowing multiple copies of the same element. The indicator function of a multiset would therefore range over \mathbb{N}_0 , the set of non-negative integers. For example the types of coins in a currency might be represented as a set: $\{\$0.01, \$0.05, \$0.10, \$0.25, \$1.00\}$. A multiset might represent the physical coins in my pocket: $2 \times \$0.05, 5 \times \$0.10, 1 \times \$1.00$. Hybrid sets take this one step further allowing for an element to occur *negatively many* times.

Definition Let U be a universe, then any function $U \rightarrow \mathbb{Z}$ is called a **hybrid set**.

By universe, we just mean a (traditional) set that *contains everything we may be interested in*. The size of a universe is context dependant but we will rarely be interested in everything

in the universe or its exact size. It is simply a compact way for us to talk about “everything else”. This definition of hybrid sets does little good on its own; much of the usefulness of sets is derived from their rich notation. So with that said,

Definition Let H be a hybrid set. Then we say that $H(x)$ is the **multiplicity** of the element x . We write, $x \in^n H$ if $H(x) = n$. Furthermore we will use $x \in H$ to denote $H(x) \neq 0$ (or equivalently, $x \in^n H$ for $n \neq 0$). Conversely, $x \notin H$ denotes $x \in^0 H$ or $H(x) = 0$. The symbol \emptyset will be used to denote the empty hybrid set for which all elements have multiplicity 0. Finally the **support of a hybrid set** is the (non-hybrid) set $\text{supp } H$, where $x \in \text{supp } H$ if and only if $x \in H$

We will use the notation:

$$H = \left\{ x_1^{m_1}, x_2^{m_2}, \dots \right\}$$

to describe the hybrid set H where the element x_i has multiplicity m_i . We allow for repetitions in $\{x_i\}$ but interpret the overall multiplicity of an element x_i as the sum of multiplicities among copies. This is, (using Iverson brackets):

$$H(x) = \sum_{x_i \in^{m_i} H} [x = x_i] m_i \quad (2.8)$$

For example, $H = \left\{ a^1, a^1, b^{-2}, a^3, b^1 \right\} = \left\{ a^5, b^{-1} \right\}$. A writing in which $x_i \neq x_j$ for all $i \neq j$ is referred to as a *normalized form* of a hybrid set. For normalized hybrid sets it follows that $H(x_i) = m_i$.

Traditional sets use the operations \cup union, \cap intersection, and \setminus complementation. In the same way a hybrid set is a function $H : U \rightarrow \mathbb{Z}$, a set could be considered as function $S : U \rightarrow \{0, 1\}$. Then set operations correspond to point-wise OR, AND, and NOT. That is, for two sets A and B , then $(A \cup B)(x) = A(x) \text{ OR } B(x)$. One could easily extend union and intersection to hybrid sets using point-wise min and max [cite], but it would make more sense to have operations corresponding to primitive operations in \mathbb{Z} instead. Thus we will define \oplus ,

\ominus , and \otimes by point-wise $+$, $-$, and \times respectively.

Definition For any two hybrid sets A and B over a common universe U , we define the operations $\oplus, \ominus, \otimes : \mathbb{Z}^U \times \mathbb{Z}^U \rightarrow \mathbb{Z}^U$ such that for all $x \in U$:

$$(A \oplus B)(x) = A(x) + B(x) \quad (2.9)$$

$$(A \ominus B)(x) = A(x) - B(x) \quad (2.10)$$

$$(A \otimes B)(x) = A(x) \cdot B(x) \quad (2.11)$$

We also define, $\ominus A$ as $\emptyset \ominus A$ and for $c \in \mathbb{Z}$:

$$(cA)(x) = c \cdot A(x) \quad (2.12)$$

Definition We say A and B are **disjoint** if and only if $A \otimes B = \emptyset$

For boolean sets A and B , disjointness would be defined by $A \cap B = \emptyset$. If we consider these boolean sets as simply hybrid sets with multiplicity 0 or 1 then the operations \cap and \otimes identically correspond to elementwise AND. From these definitions, we can use hybrid sets to model various objects that would traditionally be described otherwise.

2.1.1 Example: *Rational Arithmetic*

Any positive rational number can be represented as a hybrid set over the set of primes and vice versa (i.e. we have the group isomorphism $(\mathbb{Z}^{\mathbb{P}}, \oplus) \simeq (\mathbb{Q}_+, \cdot)$). For any rational number a/b , both a and b being integers will have a prime decomposition: $a = p_1^{m_1} \cdot p_2^{m_2} \cdot \dots$ and $b = q_1^{n_1} \cdot q_2^{n_2} \cdot \dots$. Then there is an isomorphism:

$$f(a/b) = \left\{ p_1^{m_1}, p_2^{m_2}, \dots \right\} \ominus \left\{ q_1^{n_1}, q_2^{n_2}, \dots \right\} \quad (2.13)$$

Example Concretely, we have:

$$20/9 \cdot 15/8 = \llbracket 5^1, 2^2, 3^{-2} \rrbracket \oplus \llbracket 5^1, 3^1, 2^{-3} \rrbracket = \llbracket 5^2, 2^{-1}, 3^{-1} \rrbracket = 25/6$$

Typically one would need to specify equivalence classes on \mathbb{Q} to consolidate the identity $ca/cb = a/b$, but with hybrid set representation, this identity comes for free. Any common factor between numerator and denominator will cancel result in cancelling multiplicities. Equivalence of rational numbers is identical to equivalence on hybrid sets and a normalized hybrid set. For example, $2/4 = \llbracket 2^1, 2^{-2} \rrbracket$ which is the un-normalized form of $\llbracket 2^{-1} \rrbracket = 1/2$.

One should note that this does not however extend (nicely) for zero and negative \mathbb{Q} . Allowing for our hybrid sets to contain -1 would mean we can represent negative rational numbers at the expense of having unique a unique representation. The representations of $1/2 = \llbracket 2^{-1} \rrbracket$ and $-1/-2 = \llbracket -1^2, 2^{-1} \rrbracket$ no longer agree! Allowing our hybrid sets to contain 0 leads to rational numbers which are not well defined. We would like our rational numbers to be members of $\mathbb{Z} \times \mathbb{Z} - \{0\}$ but our construction does not let us discriminate.

2.1.2 Example: *Rational Polynomials*

Similarly, we can also use hybrid sets to represent monic rational polynomials by encoding the roots and asymptotes. For example:

$$\frac{(x-2)}{(x-1)^2(x+1)} = \llbracket 2^1, 1^{-2}, -1^{-1} \rrbracket \quad (2.14)$$

Although neither model is a revolutionary method of looking at rational numbers and polynomials, hopefully they demonstrate that hybrid set are not an arcane construction. Negative multiplicities of elements can arise very naturally in many contexts.

In traditional set theory, a partition of X is a collection of subsets of X such that the subsets are mutually disjoint and their union is equivalent to X . When dealing with hybrid sets we no longer have union but use point-wise sum instead.

Definition A **generalized partition P of a hybrid set $H(x)$** is a sequence of hybrid sets $P = \{P_i\}_{i=1}^n$ such that

$$P_1 \oplus P_2 \oplus \dots \oplus P_n = H \quad (2.15)$$

We say that **P is a strict partition of H** if P_i and P_j are disjoint for all $i \neq j$.

Traditional boolean sets can be trivially converted to hybrid sets. For a set X , this is done simply by taking $H(x) = 1$ if $x \in X$ and $H(x) = 0$ if $x \notin X$. In this way any traditional set partition is a strict, generalized set partition. By principle of inclusion-exclusion,

$$P_i \cup P_j = P_i \oplus P_j \ominus (P_i \cap P_j)$$

Since P_i and P_j are disjoint then we have that $\bigcup_i P_i = \bigoplus_i P_i$ and thus a strict partition. What about the converse, how do generalized partitions relate to partitions? First we must consider that not all hybrid sets can be cast back down to traditional sets.

Definition Given a hybrid set H over universe U , if for all $x \in U$ $H(x) = 1$ or $H(x) = 0$ then we say that **$H(x)$ is reducible**. If H is reducible then we denote the **reduction of H** by $\mathcal{R}(H)$ as the (non-hybrid) set over U with the same membership.

If a hybrid set H is reducible, we still cannot be guaranteed that a generalized partition P of H will be a strict. Consider the reducible hybrid set $[0, 1]$ (that is, $H(x) = 1$ if $0 \leq x \leq 1$ and $H(x) = 0$ otherwise). Then $P = \{[0, 2], \ominus(1, 2]\}$ is a generalized partition of H . A generalized partition of a reducible set is strict if and only if each generalized partition is reducible. Generalized partitions and reducibility will be very useful to us over the next section.

2.2 Hybrid Relations

A function is typically defined as a mapping from elements of one set to another set. We will consider functions which have hybrid sets as their domain (but still map to traditional sets) which we will call *hybrid functions*. But first we must establish *hybrid relations*.

Definition For two sets S and T , a hybrid set over their Cartesian product $S \times T$ is called a **hybrid (binary) relation between S and T** . We denote the set of all such hybrid relations by $\mathbb{Z}^{S \times T}$.

2.2.1 Example: *Maximum Flow*

The maximum flow problem involves finding out the maximal volume that can be pushed from a single source node to a single sink node in a flow network. Formally, we will let $G = (V, E)$ be a directed graph such that for every edge $(u, v) \in E$ there is a non-negative **capacity** denoted $c(u, v)$ (for $(u, v) \notin E$, assume that $c(u, v) = 0$). A flow in G is a function $f : V \times V \rightarrow \mathbb{Z}$ subject to the following conditions:

Constrained by capacity: $f(u, v) \leq c(u, v)$

Skew Symmetric: $f(u, v) = -f(v, u)$

Conservative: For all $u \neq s$ and $u \neq t$, $\sum_{w \in V} f(u, w) = 0$

The flow could alternatively be modelled using a hybrid relations. We will use the hybrid set \mathcal{E} to denote a basis for constructing flows. For each edge $(u, v) \in E$, we will have $(u, v) \ominus (v, u)$ in \mathcal{E} with multiplicity $c(u, v)$:

$$\mathcal{E} = \left\{ ((u, v) \ominus (v, u))^{c(u, v)} \mid (u, v) \in E \right\} \quad (2.16)$$

Additionally define $\partial : V \times V \rightarrow V$ by $\partial(u, v) = v$ and extend linearly over hybrid sets. So, for an element of \mathcal{E} , $\partial((u, v) \ominus (v, u)) = \left\{ v^1, u^{-1} \right\}$ which can be thought of as the movement of one unit of volume across one edge that would occur when said edge is included in a flow f .

By constructing f with elements taken from \mathcal{E} , we can be guaranteed skew-symmetry. Then conservation can be written as $\partial(f) = k \cdot \left\{ s^{-1}, t^{+1} \right\}$ for some constant k . This constant k turns out to be the overall flow through the graph. So maximum flow is then just finding the

maximum k for a valid flow. And so long as $\mathcal{E} \ominus f$ is everywhere positive, all three conditions are fulfilled.

Residual graphs $G_f = \mathcal{E} \ominus f$

This can be used to formulate a “reversed” Ford-Fulkerson algorithm. Rather than start with an empty flow, start with $f = \mathcal{E}$, a full-capacity “flow”. It is unlikely that f will initially be a valid flow. By adding el

finish...

2.3 Hybrid Functions

Definition A **hybrid function from S to T** is a hybrid relation H between S and T such that $(x, y) \in H$ and $(x, z) \in H$ implies $y = z$. We denote the set of all such hybrid functions by $\mathbb{Z}^{S \rightarrow T}$.

Although this tells us what *is* and *is not* a hybrid function, it is not the most useful definition to work with. We would like to think of a hybrid function not as a mapping from a hybrid set to a boolean set but as a function between two sets with a multiplicity attached to each mapping. From this perspective, we would generally have a function and hybrid set already in mind. Given a hybrid set H over U and a function $f : B \rightarrow S$ be a function where $B \subseteq U$ and S a set. Then we denote by f^H the hybrid function from B to S defined by:

$$f^H := \bigoplus_{x \in B} H(x) \left\| (x, f(x))^1 \right\| \quad (2.17)$$

There is little literature or established use for hybrid functions; their primary use to us will be as something that we can turn back into traditional functions. One may notice that we have defined hybrid functions by their graph, taking the reduction (if it exists) of a hybrid function one naturally gets the graph of a function.

Definition If H is a reducible hybrid set, then f^H is a **reducible hybrid function**. Addition-

ally, if f^H is reducible, we extend \mathcal{R} by:

$$\mathcal{R}(f^H)(x) = f|_{\mathcal{R}(H)}(x) \quad (2.18)$$

Since $\mathcal{R}(H)$ only exists if $H(x)$ is everywhere 0 or 1; $\mathcal{R}(f^H)$ only makes sense when f^H is reducible. Assuming that we end up with a reducible function, hybrid functions make an excellent primitives to construct piecewise functions. Earlier in (2.3) we defined the *join of functions* to allow us to construct piecewise functions from *restricted functions*. The join of two hybrid functions is quite trivially defined.

Definition The **join**, $f^F \oplus g^G$ of two hybrid functions f^F and g^G is the hybrid relation given by their point-wise sum.

$$f^F \oplus g^G := f^F \oplus g^G \quad (2.19)$$

However, we will immediately dispense with using \oplus altogether and simply use \oplus in order to prevent confusion between hybrid functional join (e.g. $f^F \oplus g^G$) and traditional functional join (e.g. $f|_F \oplus g|_G$). It is important to note that the join operator is closed under hybrid relations but not under hybrid functions. For any two hybrid *functions* the result will be a hybrid *relation* but not necessarily another hybrid function. As with traditional functional join, we must still be wary of overlapping regions but non-disjoint hybrid domains are not nearly as “dangerous”. For intersecting regions we do not have to choose between commutivity and associativity, we can have both. In general, all that we can say the result is a hybrid relation but there are many cases where we can guarantee hybrid function status is preserved.

Theorem 2.3.1 *Let A and B be hybrid sets over U and let $f : U \rightarrow S$ be a function. Then $f^A \oplus f^B$ is a hybrid function. Moreover,*

$$f^A \oplus f^B = f^{A \oplus B} \quad (2.20)$$

It should not be a surprise that the same function over different restrictions combine to form a function. Let $(x, f(x)) \in f^A$ and $(y, f(y)) \in B$. Clearly, if $x = y$ then $f(x) = f(y)$. Since f is common between both hybrid functions, there cannot be disagreement among points. Additionally,

$$\begin{aligned}
 f^A \oplus f^B &= \bigoplus_{x \in U} A(x) \left\| (x, f(x))^1 \right\| \oplus \bigoplus_{x \in U} B(x) \left\| (x, f(x))^1 \right\| \\
 &= \bigoplus_{x \in U} (A(x) + B(x)) \left\| (x, f(x))^1 \right\| \\
 &= \bigoplus_{x \in U} (A \oplus B)(x) \left\| (x, f(x))^1 \right\|
 \end{aligned} \tag{2.21}$$

Inductively, this holds for any number of hybrid sets with a common function. For instance, given any generalized partition $P = P_1 \oplus P_2 \oplus \dots \oplus P_n$, we have the following equation reminiscent of (2.4):

$$f^P = f^{P_1} \oplus f^{P_2} \oplus \dots \oplus f^{P_n} \tag{2.22}$$

Joining a function with itself is not the most interesting construction. Piecewise functions are useful for their ability to tie together two *different* functions. If two functions have separate, non-overlapping regions, then our definition is again trivial.

Theorem 2.3.2 *Given two function $f : U \rightarrow S$ and $g : U \rightarrow S$. The following identity holds if and only if A and B are disjoint:*

$$f^A \oplus g^B = (f \oplus g)^{A \oplus B} \tag{2.23}$$

Notice here the use of \oplus on the right hand side. Here \oplus is traditional (non-hybrid) function join. Since A and B are disjoint, the problems that arose earlier are not an issue. In fact, it doesn't even matter which convention we use. Disjointness is still too strong of a condition for two hybrid functions to be compatible. The join of two non-disjoint functions may still be a hybrid function even if their respective functions do not agree at *all* points. So long as long as

they agree on all points in overlapping regions intersection the functions can be safely joined.

Definition For hybrid functions f^A and g^B , $f^A \oplus g^B$ is a hybrid function if and only if for all $x \in \text{supp}(A \otimes B)$, we have $f(x) = g(x)$. We say that f^A and g^B are compatible.

As with our definition of disjointness, we use the pointwise product \otimes of hybrid sets as an analog for intersection \cap of sets. This definition also generalizes the two cases we have already seen in theorems 2.3.1 and 2.3.2. The hybrid functions f^A and f^B are clearly compatible because f agrees with itself everywhere. For A and B disjoint, f^A and g^B are also compatible since $A \otimes B$ is empty; there are no mutual points for f and g to disagree over.

Compatibility becomes less clear when we begin to consider multiple hybrid functions. For one, compatibility is not associative. Consider the following sequence:

$$(f^H \oplus g^H) \oplus g^{\ominus H} = f^H \oplus (g^H \oplus g^{\ominus H}) = f^H \oplus g^{\emptyset} = f^H \quad (2.24)$$

g^H and $g^{\ominus H}$ are clearly compatible by 2.3.1. The resulting g^{\emptyset} is compatible with *any* hybrid function. Although f^H and g^H could be mutually incompatible; we can only certain that $f^H \oplus g^H$ is a hybrid relation. Regardless of their compatibility, when this is joined with $g^{\ominus H}$, the g terms cancel and we are left with *is* a hybrid function.

2.4 Hybrid Function Fold

Compatibility and reducibility are one way of collapsing a hybrid function to a traditional function. Another approach is to fold or aggregate a hybrid function using some operator. To aid in this we will first introduce some notation for iterated operators.

Definition Let $\star : S \times S \rightarrow S$ be an operator on S . Then, for $n > 0$, $n \in \mathbb{Z}$ we use $\star^n : S \times S \rightarrow S$ to denote iterated \star . So,

$$x \star^n y = (((x \star y) \star y) \star \overbrace{\dots \star y}^{n \text{ times}}) \quad (2.25)$$

If \star has an identity e_\star , then we extend $x \star^0 y = x$. If \star is invertible, then we use \star^{-1} to denote its inverse and \star^{-n} for iterated \star^{-1} . Finally, we allow \star^n to be a unary operator, which we define by $\star^n x = e_\star \star^n x$.

Assuming \star^m and \star^n are both defined (e.g. if $m, n \leq 0$ then \star must be invertible), we have $(x \star^m y) \star^n y = x \star^{m+n} y$.

For non-associative magmas, it may be of interest to instead define \star^T for some tree T . For example \star^n above is analogous to Haskell's `foldl`. There are applications where `foldr`: $(x \star (y \star (y \star \dots \star y)))$ or a balanced expression tree like `foldt` might be desired. Most applications however will be interested in group operators and so we will not explore this.

Definition We say that a hybrid relation $f^A = f_1^{A_1} \oplus f_2^{A_2} \oplus \dots$ over $T \times S$ is \star -**reducible** if (S, \star) is an abelian group or if (S, \star) is an abelian semi-group and f^A is everywhere non-negative. If f^A is \star -reducible we define its \star -**reduction**, $\mathcal{R}_\star(f^A) : T \rightarrow S$ as a (non-hybrid) function given by:

$$\mathcal{R}_\star(f^A)(x) = \left(\star^{A_1(x)} f_1(x) \star^{A_2(x)} f_2(x) \star \dots \right) \Big|_{\text{supp} A} \quad (2.26)$$

If f^A is reducible then it is trivially \star -reducible and we have:

$$\mathcal{R}(f^A) = \mathcal{R}_\star(f^A) \quad (2.27)$$

If a hybrid function is already “flattened”, then reducing it will do nothing. So clearly \mathcal{R}_\star is a projection since it is idempotent (i.e. $\mathcal{R}_\star(\mathcal{R}_\star(f^A)) = \mathcal{R}_\star(f^A)$). Moreover,

$$\mathcal{R}_\star(\mathcal{R}_\star(f^F) \oplus \mathcal{R}_\star(g^G)) = \mathcal{R}_\star(f^F \oplus g^G) \quad (2.28)$$

2.4.1 Example: Sign function

One would typically write the sign function out as a piecewise function over 3 regions of the extended real line: $(-\infty, 0)$, $\{0\}$, and $(0, \infty)$. Or alternatively using a $+$ -reduction using 2 pieces:

$(-\infty, 0]$ and $[0, \infty)$.

$$\text{sign} = -1^{(-\infty, 0]} \oplus 0^{[0]} \oplus 1^{(0, \infty)} = \mathcal{R}_+ \left(-1^{(-\infty, 0]} \oplus 1^{[0, \infty)} \right) \quad (2.29)$$

Evaluation of the sign at the points 1 or 0 is performed as follows:

$$\text{sign}(1) = \mathcal{R}_+ \left(-1^{(-\infty, 0]} \oplus 1^{[0, \infty)} \right) (1) = \mathcal{R}_+ \left(-1^0 \oplus 1^1 \right) = +^0(-1) +^1(1) = 1$$

$$\text{sign}(0) = \mathcal{R}_+ \left(-1^{(-\infty, 0]} \oplus 1^{[0, \infty)} \right) (0) = \mathcal{R}_+ \left(-1^1 \oplus 1^1 \right) = +^1(-1) +^1(1) = 0$$

Going from 3 regions to 2 may seem a small step. However, consider two piecewise functions with n and m regions respectively. Taking the sum of these two functions would lead to a new piecewise function with $n \cdot m$ regions. We will show that \star -reductions can allow us to reduce this to only a linear increase!

2.4.2 Example: Piecewise functions on generalized partitions

Let $A_1 = [0, a)$, $A_2 = [0, 1] \setminus A_1$, $B_1 = [0, b)$ and $B_2 = [0, 1] \setminus B_1$. As well as piecewise functions f and g given as:

$$f(x) = f_1^{A_1} \oplus f_2^{A_2} = \begin{cases} f_1(x) & x \in A_1 \\ f_2(x) & x \in A_2 \end{cases} \quad \text{and} \quad g(x) = g_1^{B_1} \oplus g_2^{B_2} = \begin{cases} g_1(x) & x \in B_1 \\ g_2(x) & x \in B_2 \end{cases}$$

If one were interested in computing the piecewise function $(f+g)$, the naïve method would be to take the pairwise sum of each sub-function. Then restrict each sum to the intersection of respective partitions as below:

$$(f+g)(x) = (f_1 + g_1)^{A_1 \cap B_1} \oplus (f_1 + g_2)^{A_1 \cap B_2} \oplus (f_2 + g_1)^{A_2 \cap B_1} \oplus (f_2 + g_2)^{A_2 \cap B_2} \quad (2.30)$$

We will take another approach. First, we can partition $[0, 1]$ into the generalized partition

$A_1, B_1 \ominus A_1, B_2$. The source of this particular partition will remain mysterious for now but observe that we can still construct the partitions: $B_1 = (B_1 \ominus A_1) \oplus A_1$ and $A_2 = (B_1 \ominus A_1) \oplus B_2$. And so we can represent f and g from above with a common partition by using:

$$f = f_1^{A_1} \oplus f_2^{(B_1 \ominus A_1) \oplus B_2} = f_1^{A_1} \oplus f_2^{B_1 \ominus A_1} \oplus f_2^{B_2} \quad (2.31)$$

$$g = g_1^{A_1 \oplus (B_1 \ominus A_1)} \oplus g_2^{B_2} = g_1^{A_1} \oplus g_1^{B_1 \ominus A_1} \oplus g_2^{B_2} \quad (2.32)$$

Since we have a common partition we can avoid computing pairwise intersections altogether and simply add each sub-function to the corresponding sub-function which shares a partition. Since $\{A_1, B_1, (B_1 \ominus A_1)\}$ is a generalized partition, we will need to flatten the expression back down to get a traditional function. We use \mathcal{R}_+ for this so that negative regions properly cancel.

$$(f + g)(x) = \mathcal{R}_+ \left((f_1 + g_1)^{A_1} \oplus (f_2 + g_1)^{B_1 \ominus A_1} \oplus (f_2 + g_2)^{B_2} \right) \quad (2.33)$$

This equation holds regardless of the relative ordering of a and b . Suppose $x \in A_1 \cap B_1$. Then we have $A_1(x) = 1$ and $(B_1 \ominus A_1)(x) = B_2(x) = 0$. And so $(f + g)(x) = (f_1 + g_1)(x)$. Similarly, if $x \in B_1 \cap A_2$ or $x \in A_2 \cap B_2$ then only $(B_1 \ominus A_1)(x)$ or $B_2(x)$ will respectively be non-zero. However, if $x \in B_2 \cap A_1$ then we have $A_1(x) = 1$, $(B_1 \ominus A_1)(x) = -1$ and $B_2(x) = 1$. Simplifying this expression yields:

$$(f + g) = +^1(f_1 + g_1) +^{-1}(f_2 + g_1) +^1(f_2 + g_2) = (f_1 + g_2) \quad (2.34)$$

In the above example only three regions could simultaneously exist. If $a < b$ then $A_1 \cap B_2 = \emptyset$ but if $b < a$ then $A_2 \cap B_1 = \emptyset$. Although it might seem that the three terms are a result of three regions, in the general case where A_1 and B_1 could be arbitrary subsets, we still only have three terms. We will even extend this to any generalized partition. First we will formalize some ideas we have already seen.

Definition A **refinement** of a generalized partition $P = \{P_i\}_{i \in I}$ is another generalized partition $Q = \{Q_j\}_{j \in J}$ such that, for every P_i in P there is a subset of Q : $\{Q_j\}_{j \in J_i}$, $J_i \subseteq J$ such that for some integers $\{a_{i,j}\}_{j \in J_i}$

$$P_i = \bigoplus_{j \in J_i} a_{i,j} Q_j \quad (2.35)$$

Given a *set* of generalized partitions a **common refinement** is a generalized partition which is a refinement of every partition in the set. A refinement Q of P is **strict** if $\text{supp}(Q) = \text{supp}(P)$.

In our previous example we used $\{A_1, B_1 \ominus A_1, B_2\}$ which was a common refinement of both $\{A_1, A_2\}$ and $\{B_1, B_2\}$. A_1 and B_2 have trivial representations while $A_2 = (B_1 \ominus A_1) \oplus B_2$ and $B_1 = A_1 \oplus (B_1 \ominus A_1)$. Another common refinement would be the trivial $\{A_1, B_1, A_2, B_2\}$ This is less preferable due to not only containing 4 regions instead of 3 but also the pointwise sum is $[0, 1]^2$ instead of the reducible $[0, 1]$.

We can now formally phrase our problem. Let $A = \{A_i\}_{i \in [n]}$ and $B = \{B_j\}_{j \in [m]}$ be two generalized partitions of a hybrid set U (where $[n] = \{1, \dots, n\}$ for $n \in \mathbb{N}$). We wish to find a generalized partition C of U which is a *common, strict* refinement of A and B and has minimal cardinality. These conditions can be summarized into the following linear system of $n + m + 1$ simultaneous equations:

$$U = \bigoplus_i C_i \quad (2.36)$$

$$\forall i \in [n] : A_i = \bigoplus_j a_{i,j} C_j \quad (2.37)$$

$$\forall j \in [m] : B_j = \bigoplus_i b_{i,j} C_i \quad (2.38)$$

for some integers $a_{i,j}$ and $b_{i,j}$. Since we know that A and B are each partitions of U , only $n + m - 1$ can be independent. If there are additional dependencies between A and B then this can be even lower. Expressed as a linear system we have:

$$M \cdot \begin{pmatrix} C_1 \\ \vdots \\ C_{n+m-1} \end{pmatrix} = \begin{pmatrix} U \\ A_1 \\ \vdots \\ A_{n-1} \\ B_1 \\ \vdots \\ B_{m-1} \end{pmatrix} \quad \text{where} \quad M = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_{1,1} & a_{1,2} & \cdots & a_{1,n+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n+m-1} \\ b_{1,1} & b_{1,2} & \cdots & b_{1,n+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m-1,1} & b_{m-1,2} & \cdots & b_{m-1,n+m-1} \end{pmatrix} \quad (2.39)$$

By definition, M is an integer matrix. But we are actually more interested in its inverse M^{-1} as this will give us values for C_i relative to $(U, A_1, \dots, A_{m-1}, B_1, \dots, B_{m-1})$. To stay in the realm of hybrid sets, we would also like to enforce that M^{-1} is also an integer matrix. Assuming this, then the determinant of M must be ± 1 . Further restricting ourselves to upper triangular matrices we can choose M to be all 1's along the diagonal as well as the top row. Which has the following inverse:

$$\begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -1 & \cdots & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix} \quad (2.40)$$

Thus we find that for the partitions $A = \{A_i\}_{i \in [n]}$ and $B = \{B_j\}_{j \in [m]}$ we can *always* use the strict, common and minimal refinement:

$$\{(U \ominus A_1 \ominus \dots \ominus A_{n-1} \ominus B_1 \ominus \dots \ominus B_{m-1}), A_1, A_2, \dots, A_{n-1}, B_1, B_2, \dots, B_{m-1}\} \quad (2.41)$$

Finally, to generalize our example from earlier, let $f = f_1^{P_1} \oplus f_2^{P_2} \oplus f_n^{P_n}$ and $g = g_1^{Q_1} \oplus \dots \oplus g_m^{Q_m}$

be two piecewise functions over a common domain U . We can compute $f \star g$ by:

$$\begin{aligned} f \star g = \mathcal{R}_\star \Big(& (f_1 \star g_m)^{P_1} \oplus \dots \oplus (f_{n-1} \star g_m)^{P_{n-1}} \\ & \oplus (f_n \star g_1)^{Q_1} \oplus \dots \oplus (f_n \star g_{m-1})^{Q_{m-1}} \\ & \oplus (f_n \star g_n)^{U \ominus (P_1 \oplus \dots \oplus Q_{n-1} \oplus Q_1 \oplus \dots \oplus Q_{m-1})} \Big) \end{aligned} \quad (2.42)$$

This is not a unique formulation. Any upper triangular matrix containing only 1, 0 and -1 and all 1's in the first row would also be acceptable. Each would lead to slightly different forms of (2.47). This particular choice is merely one of the simplest options. Certain applications could lead to other choices of M for reasons of memory efficiency, but it is not clear this is needed.

2.5 Pseudo-functions

One last detail remains to be settled, the sub-functions f_i and g_j may not be defined outside of P_i and Q_j respectively. Consider the equation in (2.47), evaluated at $x \in P_1 \cap Q_1$. Then we have the following terms with non-zero multiplicity.

$$(f \star g)(x) = \mathcal{R}_\star \left((f_1(x) \star g_m(x))^1 \oplus (f_n(x) \star g_1(x))^1 \oplus (f_n(x) \star g_m(x))^{-1} \right) \quad (2.43)$$

We would like for the $g_m(x)$ in the first term to cancel with the $g_m(x)$ in the third. If q_m (and f_n) are defined and finite at all points in U , then there is no problem. However if $g_m(x) = \infty$, then we need to make cumbersome arguments for: $\infty + y - \infty = y$. To resolve this, we use a lambda-lifting trick to have a hybrid relation over the domain and *the function itself* rather than the domain and the image implied by the function.

Definition Using the same notation as in our definition from (2.17), we define a pseudo-

function \tilde{f}^A as:

$$\tilde{f}^A = \bigoplus_{x \in B} A(x) \llbracket (x, f)^1 \rrbracket \quad (2.44)$$

One should notice the similarity between (2.49) and (2.22). The difference is that we have replaced $(x, f(x))$ with the unevaluated (x, f) . This formally makes \tilde{f}^A a hybrid relation over $U \times (U \rightarrow S)$ as opposed to a hybrid function over $U \times S$. To evaluate \tilde{f}^A we map back to f^A and evaluate that. This mapping between $(x, f(x))$ and (x, f) is very natural and we will perform it unceremoniously, often using f^A and \tilde{f}^A interchangeably. Properties of hybrid functions such as compatibility and reducibility will be lifted to hybrid pseudo-functions by this as well.

2.5.1 Example: *Piecewise functions revisited*

We will repeat the example from 2.4.2 but concretely using the following function:

$$f(x) = \begin{cases} (2 - x^2) & -1 \leq x \leq 1 \\ 1/x^2 & \text{otherwise} \end{cases} \quad (2.45)$$

Graphically, this is a mundane-looking bell-shaped curve shown in black in Figure 2.2. Although f is defined for all of \mathbb{R} , this is not the case for its sub-functions. The plots in red and blue show the behaviour of $(2 - x^2)$ and $1/x^2$ outside of their defined ranges in f .

Represented as a hybrid pseudo-functions we have \tilde{f} as:

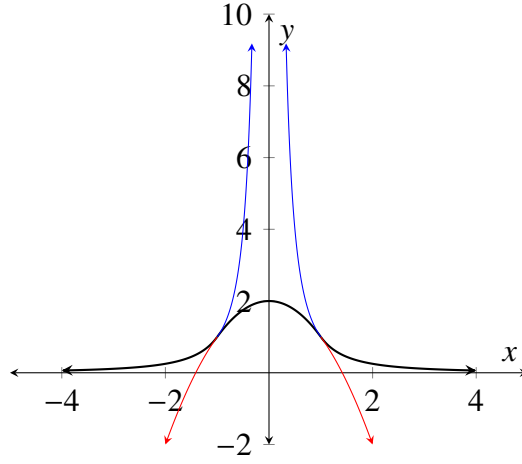
$$\tilde{f} = \tilde{f}_1^{A_1} \oplus \tilde{f}_2^{A_2} = (x \mapsto 2 - x^2)^{[-1,1]} \oplus (x \mapsto 1/x^2)^{(-\infty,-1) \oplus (1,\infty)} \quad (2.46)$$

which we are interested in multiplying by the unevaluated absolute value function:

$$\tilde{g} = \tilde{a}bs = (x \mapsto x)^{[0,\infty)} \oplus (x \mapsto -x)^{(-\infty,0)} \quad (2.47)$$

First we must find a minimal common refinement. Using the formula in (2.46), we decide

Figure 2.1: A piecewise rational function is shown in black. The plots in red and blue are continuations of $(2 - x^2)$ and $1/x^2$ respectively.



to use the partition:

$$P_1 = [-1, 1] \tag{2.48}$$

$$P_2 = [0, \infty) \tag{2.49}$$

$$P_3 = \mathbb{R} \ominus [-1, 1] \ominus [0, \infty) \tag{2.50}$$

where P_3 could also be simplified into disjoint regions by $P_3 = (-\infty, -1) \ominus [0, 1]$. Thus we have the expression

$$\begin{aligned} (\tilde{f} \times \tilde{g}) &= \mathcal{R}_\times \left(\left((x \mapsto 2 - x^2) \times (x \mapsto -x) \right)^{P_1} \right. \\ &\quad \oplus \left((x \mapsto 1/x^2) \times (x \mapsto -x) \right)^{P_2} \\ &\quad \left. \oplus \left((x \mapsto 1/x^2) \times (x \mapsto -x) \right)^{P_3} \right) \end{aligned} \tag{2.51}$$

If we use pseudo-functions then we assign multiplicities to the still unevaluated sub-functions.

So, for example to evaluate $(f \times g)$ at 0, we would evaluate each of P_1 , P_2 and P_3 at 0 and find:

$$\begin{aligned} (\tilde{f} \times \tilde{g})(0) &= \mathcal{R}_\times \left(\left((x \mapsto 2 - x^2) \times (x \mapsto -x) \right)^1 \right. \\ &\quad \oplus \left((x \mapsto 1/x^2) \times (x \mapsto -x) \right)^1 \\ &\quad \left. \oplus \left((x \mapsto 1/x^2) \times (x \mapsto -x) \right)^{-1} \right)(0) \end{aligned} \quad (2.52)$$

Once we have this, we can then evaluate the \times -reduction on the still unevaluated functions. Clearly $x \mapsto -x$ occurs with cancelling signs as does $x \mapsto 1/x^2$. This leaves us with the product of two unevaluated functions:

$$(\tilde{f} \times \tilde{g})(0) = \left((x \mapsto 2 - x^2) \times (x \mapsto x) \right)(0) \quad (2.53)$$

After these cancellations occur, then it is finally safe to evaluate at a point and we find $(\tilde{f} \times \tilde{g})(0) = 0$ as expected. To contrast this, if one were to attempt to evaluate a the (non-pseudo) hybrid function version of $\tilde{f} \times \tilde{g}$, instead of (2.57) one would have the expression:

$$(f \times g)(0) = \mathcal{R}_\times \left((2 \times 0)^1 \oplus (" \infty " \cdot 0)^1 \oplus (" \infty " \cdot 0)^{-1} \right) \quad (2.54)$$

Simplifying the \times -reduction leads to both ∞/∞ and $0/0$. Although they *should* obviously cancel, the behaviour is not well defined.

Chapter 3

Symbolic Block Linear Algebra

In mathematics literature, it is common practice to represent matrices as being broken up into blocks or sub-matrices. For example if A , B , C and D are matrices given by:

$$A = \begin{bmatrix} A_{1,1} & \dots & A_{1,m} \\ \vdots & & \vdots \\ A_{n,1} & \dots & A_{n,m} \end{bmatrix} \quad B = \begin{bmatrix} B_{1,1} & \dots & B_{1,q} \\ \vdots & & \vdots \\ B_{n,1} & \dots & B_{n,q} \end{bmatrix} \quad C = \begin{bmatrix} C_{1,1} & \dots & C_{1,m} \\ \vdots & & \vdots \\ C_{r,1} & \dots & C_{r,m} \end{bmatrix} \quad D = \begin{bmatrix} D_{1,1} & \dots & D_{1,q} \\ \vdots & & \vdots \\ D_{r,1} & \dots & D_{r,q} \end{bmatrix} \quad (3.1)$$

Where, critically, A and C have the same width (namely m) as do B and D (width q). Additionally, A and B have the same height (in this case n), as do C and D (height r). Then they can be “glued” together into a single **(2 × 2) block matrix** M . Notationally, we write these matrices as elements of M but we *interpret* M as a sort of concatenation of the sub-matrices.

$$M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right] = \left[\begin{array}{ccc|ccc} A_{1,1} & \dots & A_{1,m} & B_{1,1} & \dots & B_{1,q} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{n,1} & \dots & A_{n,m} & B_{n,1} & \dots & B_{n,q} \\ \hline C_{1,1} & \dots & C_{1,m} & D_{1,1} & \dots & D_{1,q} \\ \vdots & & \vdots & \vdots & & \vdots \\ C_{r,1} & \dots & C_{r,m} & D_{r,1} & \dots & D_{r,q} \end{array} \right] \quad (3.2)$$

So M is actually a $(n + r) \times (m + q)$ matrix but we write it as a 2×2 *block* matrix. Within an individual block, there is a one-to-one correspondence from the entries of M to the entries of a sub-matrix (shifted by some offset). In the above example, elements of B would have an offset of $(0, m)$ since $B_{i,j}$ corresponds with $M_{i+0,j+m}$.

There is no reason to stop at combining 4 matrices into a 2×2 block matrix. We can take a set of matrices $A_{i,j}$ and combine them into an **$(n \times m)$ block matrix**:

$$M = \left[\begin{array}{c|c|c|c} A_{1,1} & A_{1,2} & \dots & A_{1,m} \\ \hline A_{2,1} & A_{2,2} & \dots & A_{2,m} \\ \hline \vdots & \vdots & & \vdots \\ \hline A_{n,1} & A_{n,2} & \dots & A_{n,m} \end{array} \right]$$

In the 2×2 case, we enforced that for example the height of A was the same as the height of B . Similarly, here it is an important condition that these partitions are divided by *unbroken* horizontal and vertical lines. Formally, for each sub-matrix $A_{i,j}$ in M then $A_{i,j}$ is a $s_i \times t_j$ matrix for strictly positive integer sequences $\{s_i\}_{i=1}^n$ and $\{t_j\}_{j=1}^m$ common to all sub-matrices. As before we interpret the block matrix as a concatenation of its sub-matrices. Thus M is a $n \times m$ block matrix but a $(\sum_{i=1}^n s_i) \times (\sum_{j=1}^m t_j)$ block matrix.

Clearly block matrices are at the very least a convenient notation but they also have considerable practical applications as well. For example when multiplying large matrices, block matrices can be used to improve cache complexity [15]. Additionally, in some cases, when a sub-matrices are known to have a nice properties, many optimizations can arise. For example to invert what is known as a *block diagonal matrix*, one can invert each block individually:

$$\left[\begin{array}{cccc} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & A_n \end{array} \right]^{-1} = \left[\begin{array}{cccc} A_1^{-1} & 0 & \dots & 0 \\ 0 & A_2^{-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & A_n^{-1} \end{array} \right] \quad (3.3)$$

Figure 3.1: There are 9 possible permutations. 1x (a) 2x (b) 2x (c) 4x (d)



Although block matrices are commonly used with fixed values for bounds. Computer algebra has unsatisfactory structures to represent these when the bounds between blocks are symbolic. For example, the sum of 2×2 block matrices, naively leads to 9 possible cases of overlapping regions depending on the relationship between horizontal and vertical boundaries between blocks. In this chapter we will show a method using hybrid functions to avoid this case based approach for both addition and multiplication of matrices. First we introduce some notation that will be used frequently over the next few chapters.

3.1 Oriented Intervals

Definition Given a totally ordered set (X, \leq) (and with an implied strict ordering $<$), for any $a, b \in X$, an **interval between a and b** is the set of elements in X between a and b , up to inclusion of a and b themselves. Formally:

$$\begin{aligned}
 [a, b]_X &= \{x \in X \mid a \leq x \leq b\} \\
 [a, b)_X &= \{x \in X \mid a \leq x < b\} \\
 (a, b]_X &= \{x \in X \mid a < x \leq b\} \\
 (a, b)_X &= \{x \in X \mid a < x < b\}
 \end{aligned} \tag{3.4}$$

When context makes X obvious or the choice of X is irrelevant, we shall omit the subscript.

It should be noted that when b is less than a , $[b, a]$ is the empty set. In terms of idempotency, the bounds determine whether or not an interval will be empty. $[a, a]$ which contains a and all points equivalent to a while (a, a) , $(a, a]$, and $[a, a)$ are all empty sets. As intervals are simply sets, they can naturally be interpreted as hybrid sets. If $a \leq b \leq c$, for intervals then we have

$[a, b) \oplus [b, c) = [a, c)$. In this case, \oplus seems to behave like concatenation but this is not always true. If instead we had $a \leq c \leq b$ then $[a, b) \oplus [b, c) = [a, b)$.

$$[a, b) \oplus [b, c) = \begin{cases} [a, c) & a \leq b \leq c \\ [a, b) & a \leq c \leq b \\ [b, c) & b \leq a \leq c \\ \emptyset & \text{otherwise} \end{cases}$$

One could alternatively write $[a, b) \oplus [b, c) = [\min(a, b), \max(b, c))$ but this simply sweeps the problem under the rug. When working with intervals, a case-based approach to consider relative ordering of endpoints easily becomes quite cumbersome.

Definition We define **oriented intervals** with $a, b \in X$, where X is a totally ordered set, using hybrid set point-wise subtraction as follows:

$$\begin{aligned} \llbracket a, b \rrbracket &= [a, b) \ominus [b, a) \\ \langle\langle a, b \rangle\rangle &= (a, b] \ominus (b, a] \\ \llbracket a, b \rrbracket &= [a, b] \ominus (b, a] \\ \langle\langle a, b \rangle\rangle &= (a, b) \ominus [b, a] \end{aligned} \tag{3.5}$$

For any choice of *distinct* a and b , exactly one term will be empty; there can be no “mixed” multiplicities from a single oriented interval. Unlike traditional intervals where $[a, b)$ would be empty if $b < a$, the oriented interval $\llbracket a, b \rrbracket$ will have elements with negative multiplicity. Several results follow immediately from this definition.

Theorem 3.1.1 For all a, b, c ,

$$\begin{aligned} \llbracket a, b \rrbracket &= \ominus \llbracket b, a \rrbracket \\ \langle\langle a, b \rangle\rangle &= \ominus \langle\langle b, a \rangle\rangle \\ \llbracket a, b \rrbracket &= \ominus \langle\langle a, b \rangle\rangle \\ \langle\langle a, b \rangle\rangle &= \ominus \llbracket a, b \rrbracket \end{aligned} \tag{3.6}$$

We should make a note here how oriented intervals behave when $a = b$. Like their unoriented analogues, the oriented intervals $\llbracket a, a \rrbracket$ and $\llbracket a, a \rrbracket$ are still both empty sets. The interval $\llbracket a, a \rrbracket$ still contains points equivalent to a (with multiplicity 1). However, unlike traditional intervals $\llbracket a, a \rrbracket$ is *not* empty but rather, $\llbracket a, a \rrbracket = \ominus \llbracket a, a \rrbracket$ and so contains all points equivalent to a but with a multiplicity of -1 . The advantage of using oriented intervals is that now \oplus does behave like concatenation.

Theorem 3.1.2 *For all a, b, c (regardless of relative ordering),*

$$\llbracket a, b \rrbracket \oplus \llbracket b, c \rrbracket = \llbracket a, c \rrbracket \quad (3.7)$$

Proof Following from definitions we have:

$$\begin{aligned} \llbracket a, b \rrbracket \oplus \llbracket b, c \rrbracket &= ([a, b] \ominus [b, a]) \oplus ([b, c] \ominus [c, b]) \\ &= ([a, b] \oplus [b, c]) \ominus ([c, b] \oplus [b, a]) \end{aligned}$$

Case 1: $a \leq c$ then $[c, a] = \emptyset$ and so $\llbracket a, c \rrbracket = [a, c]$.

Case 1.a: $a \leq b \leq c$ then $[c, b] = [b, a] = \emptyset$ and $[a, b] \oplus [b, c] = [a, c]$

Case 1.b: $b \leq a \leq c$ then $[b, c] \ominus [b, a] = [b, a] \oplus [a, c] \ominus [b, a] = [a, c]$

Case 1.c: $a \leq c \leq b$ then $[a, b] \ominus [c, b] = ([a, c] \oplus [c, b]) \ominus [c, b] = [a, c]$

Case 2: $c < a$ then $[a, c] = \emptyset$ and so $\llbracket a, c \rrbracket = \ominus[c, a]$.

Case 2.a: $c \leq b \leq a$ then $[a, b] = [b, c] = \emptyset$ and $\ominus[c, b] \ominus [b, a] = \ominus[c, a]$

Case 2.b: $b \leq c \leq a$ then $\ominus[b, a] \oplus [b, c] = \ominus([b, c] \oplus [c, a]) \oplus [b, c] = \ominus[c, a]$

Case 2.c: $c \leq a \leq b$ then $\ominus[c, b] \oplus [a, b] = \ominus([c, a] \oplus [a, b]) \oplus [a, b] = \ominus[c, a]$

This sort of reasoning is routine but a constant annoyance when dealing with intervals and is exactly the reason we want to be working with oriented intervals. But now that the above

work is done, we can use oriented intervals and not concern ourselves with the relative ordering of points. Many similar formulations such as $\llbracket a, b \rrbracket \oplus \llbracket (b, c) \rrbracket = \llbracket a, c \rrbracket$ or $\llbracket (a, b) \rrbracket \oplus \llbracket b, c \rrbracket = \llbracket (a, c) \rrbracket$ are also valid for any ordering of a, b, c by an identical argument.

3.2 Vector Addition

Addition for partitioned vectors and 2×2 matrices using hybrid functions has already been considered in [19, 8]. The method is nearly identical to that of adding piecewise functions. In fact, one could think of both as simply addition of piecewise functions over a subset of \mathbb{N} and $\mathbb{N} \times \mathbb{N}$ respectively. However it will provide a good example of oriented intervals in use and as an introduction to multiplication of symbolic block matrices.

First we will consider the addition of two n -dimensional vectors. Addition of two vectors: $U = (u_1, u_2, \dots, u_n)$ and $V = (v_1, v_2, \dots, v_n)$ is itself an n dimensional vector defined as:

$$U + V = (u_1 + v_1, u_2 + v_2, \dots, u_n + v_n) \quad (3.8)$$

In particular, we would like to consider the addition of vectors U and V which are each partitioned into two intervals, $[1, k]$ and $(k, n]$ as well as $[1, \ell]$ and $(\ell, n]$. Over each interval, taking the value of different functions, as in:

$$U = [u_1, u_2, \dots, u_k, u'_1, u'_2, \dots, u_{n-k}] \quad (3.9)$$

$$V = [v_1, v_2, \dots, v_\ell, v'_1, v'_2, \dots, v_{n-\ell}] \quad (3.10)$$

These can be written more concisely as hybrid functions over intervals. Using intervals, these vectors can be represented by hybrid functions over their indices. For example

$$U = (i \mapsto u_i)^{\llbracket 1, k \rrbracket} \oplus (i \mapsto u_{i-k})^{\llbracket (k, n] \rrbracket} \quad (3.11)$$

$$V = (i \mapsto v_i)^{\llbracket 1, \ell \rrbracket} \oplus (i \mapsto v_{i-\ell})^{\llbracket (\ell, n] \rrbracket} \quad (3.12)$$

Although for clarity and succinctness we will use (u_i) instead of $(i \mapsto u_i)$.

$$U = (u_i)^{\llbracket 1, k \rrbracket} \oplus (u_{i-k})^{\langle\langle k, n \rangle\rangle} \quad (3.13)$$

$$V = (v_i)^{\llbracket 1, \ell \rrbracket} \oplus v_{i-\ell}^{\langle\langle \ell, n \rangle\rangle} \quad (3.14)$$

To add U and V

$$U + V = \left((u_i)^{\llbracket 1, k \rrbracket} \oplus (u'_{i-k})^{\langle\langle k, n \rangle\rangle} \right) + \left((v_i)^{\llbracket 1, \ell \rrbracket} \oplus (v'_{i-\ell})^{\langle\langle \ell, n \rangle\rangle} \right) \quad (3.15)$$

$$= \left((u_i)^{\llbracket 1, k \rrbracket} \oplus (u'_{i-k})^{\langle\langle k, \ell \rangle\rangle} \oplus (u'_{i-k})^{\langle\langle \ell, n \rangle\rangle} \right) + \left((v_i)^{\llbracket 1, k \rrbracket} \oplus (v_i)^{\langle\langle k, \ell \rangle\rangle} \oplus (v'_{i-\ell})^{\langle\langle \ell, n \rangle\rangle} \right) \quad (3.16)$$

$$= \mathcal{R}_+ \left((u_i + v_i)^{\llbracket 1, k \rrbracket} \oplus (u'_{i-k} + v_i)^{\langle\langle k, \ell \rangle\rangle} \oplus (u'_{i-k} + v'_{i-\ell})^{\langle\langle \ell, n \rangle\rangle} \right) \quad (3.17)$$

The choice to partition $\llbracket 1, n \rrbracket$ into $\llbracket 1, k \rrbracket \oplus \langle\langle k, \ell \rangle\rangle \oplus \langle\langle \ell, n \rangle\rangle$ is only one common refinement.

We can just as easily use $\llbracket 1, \ell \rrbracket \oplus \langle\langle \ell, k \rangle\rangle \oplus \langle\langle k, n \rangle\rangle$ to get the equivalent expression:

$$U + V = \mathcal{R}_+ \left((u_i + v_i)^{\llbracket 1, \ell \rrbracket} \oplus (u_i + v'_{i-\ell})^{\langle\langle \ell, k \rangle\rangle} \oplus (u'_{i-k} + v'_{i-\ell})^{\langle\langle k, n \rangle\rangle} \right) \quad (3.18)$$

We must be careful while evaluating these expressions to not forget that $(u'_{i-k} + v_i)$ is actually shorthand for the function:

$$(u'_{i-k} + v_i) = (i \mapsto u'_{i-k}) + (i \mapsto v_i) = (i \mapsto u'_{i-k} + v_i)$$

As a function, it may not be evaluable over the entire range implied in a given term. The same lambda-lifting trick of using pseudo-functions as in the previous section easily solves this.

For example, consider the concrete example where $n = 5$, $k = 4$ and $\ell = 1$ so that $U = [u_1, u_2, u_3, u_4, u'_1]$ and $V = [v_1, v'_1, v'_2, v'_3, v'_4]$. We will also only assume that the functions u_i, u'_i, v_i and v'_i are defined only on the intervals in which they appear (e.g. u_5 is undefined, as is v'_1).

Then we have:

$$U + V = (u_i + v_i)^{\llbracket 1, 4 \rrbracket} \oplus (u'_{i-4} + v_i)^{\llbracket 4, 1 \rrbracket} \oplus (u'_{i-4} + v'_{i-1})^{\llbracket 1, 5 \rrbracket}$$

None of the individual sub-terms cannot be evaluated directly. In the first term, v_i is not totally defined over the interval $\llbracket 1, 4 \rrbracket$. In the third term, on the interval $\llbracket 1, 5 \rrbracket$, u'_{i-4} would even be evaluated on negative indices. However, these un-evaluable terms also appear in the middle term however the interval $\llbracket 4, 1 \rrbracket$ is a negatively oriented interval and the offending points cancel exactly as in the previous chapter.

$$\begin{aligned} U + V &= (u_i + v_i)^{\llbracket 1, 1 \rrbracket \oplus \llbracket 1, 4 \rrbracket} \oplus (u'_{i-4} + v_i)^{\ominus \llbracket 1, 4 \rrbracket} \oplus (u'_{i-4} + v'_{i-1})^{\llbracket 1, 4 \rrbracket \oplus \llbracket 4, 5 \rrbracket} \\ &= (u_i + v_i)^{\llbracket 1, 1 \rrbracket} \oplus ((u_i + v_i) - (u'_{i-4} + v_i) + (u'_{i-4} + v'_{i-1}))^{\llbracket 1, 4 \rrbracket} \oplus (u'_{i-4} + v'_{i-1})^{\llbracket 4, 5 \rrbracket} \\ &= (u_i + v_i)^{\llbracket 1, 1 \rrbracket} \oplus (u_i + v'_{i-1})^{\llbracket 1, 4 \rrbracket} \oplus (u'_{i-4} + v'_{i-1})^{\llbracket 4, 5 \rrbracket} \end{aligned}$$

3.3 Higher dimension intervals

Oriented intervals work perfectly well when we are only dealing with the indices of a vector. However, we are more interested in the rectangular blocks of a matrix. We can move from 1-dimensional intervals to 2-dimensional blocks using the Cartesian product

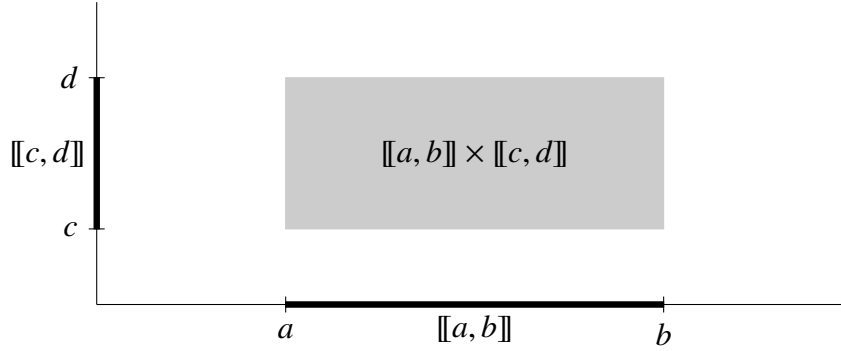
Definition Let $X = \{x_1^{m_1}, \dots, x_k^{m_k}\}$ and $Y = \{y_1^{n_1}, \dots, y_\ell^{n_\ell}\}$ be hybrid sets over sets S and T . We define the **Cartesian product of hybrid sets X and Y** , to be a hybrid set over $S \times T$ and denoted with \times operator as

$$X \times Y = \{(x, y)^{m \cdot n} : x \in^m X, y \in^n Y\} \quad (3.19)$$

If $\llbracket a, b \rrbracket$ and $\llbracket c, d \rrbracket$ are both positively oriented intervals in \mathbb{R} then their Cartesian product $\llbracket a, b \rrbracket \times \llbracket c, d \rrbracket$ is shown in Figure 4.1 is clearly a two dimensional rectangle in \mathbb{R}^2 . If one of $\llbracket a, b \rrbracket$ or $\llbracket c, d \rrbracket$ were negatively oriented then we would have a negatively oriented rectangle.

If both were negative, then the signs will cancel and the Cartesian product will be *positively* oriented.

Figure 3.2: The Cartesian product of two positively oriented 1-rectangles $\llbracket a, b \rrbracket$ and $\llbracket c, d \rrbracket$ is a positively oriented 2-rectangle.



There is no reason to stop here. $\llbracket a, b \rrbracket \times \llbracket c, d \rrbracket$ is still a hybrid set, we can take its Cartesian product with another interval, say $\llbracket e, f \rrbracket$ to get a rectangular cuboid in \mathbb{R}^3 . We should note here that we do not distinguish between $((x, y), z)$ and $(x, (y, z))$ but rather we treat both as different names for the ordered triple (x, y, z) . That is, the Cartesian product is associative, and any difference in the brackets that arise:

$$\left\{ ((x, y), z)^{(m \cdot n) \cdot p} \mid x \in^m X, y \in^n Y, z \in^p Z \right\} = X \times Y \times Z = \left\{ (x, (y, z))^{m \cdot (n \cdot p)} \mid x \in^m X, y \in^n Y, z \in^p Z \right\}$$

Although we will not be using them in this chapter, the objects resulting from iterated Cartesian product of intervals turn out to be quite useful. We will call them k -rectangles. A 1-dimensional (non-degenerate) oriented interval will be called a 1-rectangle. A 2-dimensional rectangle will be called a 2-rectangle and a cuboid a 3-rectangle, and so on.

Theorem 3.3.1 *The Cartesian product of a k -rectangle in \mathbb{R}^m (where, $k \leq m$) and ℓ -rectangle in \mathbb{R}^n (again, $\ell \leq n$) is a $(k + \ell)$ -rectangle in \mathbb{R}^{m+n} .*

For completeness we will also define a 0-rectangle as a hybrid set containing a single point with multiplicity 1 or -1 . This allows us to embed k -rectangles in \mathbb{R}^n . For example $\llbracket a, b \rrbracket_{\mathbb{R}} \times \llbracket c, d \rrbracket_{\mathbb{R}} \times \left\{ e^1 \right\}$ is the product of two 1-rectangles and a 0-rectangle and so it is a 2-rectangle.

But it was still a Cartesian product of 3 hybrid sets (each over \mathbb{R}) and so is a 2-rectangle in \mathbb{R}^3 . Specifically, it is the 2-rectangle $\llbracket a, b \rrbracket \times \llbracket c, d \rrbracket$ on the plane $z = e$. This also illustrates the principle that given a k -rectangle in \mathbb{R}^n where $n > k$ we can always find a k dimensional subspace which also contains the rectangle.

Finally, one last note regarding k -rectangles before we return to the realm of symbolic linear algebra. We will re-use the interval notation and allow for intervals between two vectors: $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$. But one should be careful to “type-check” when interpreting. When a and b are real numbers then we continue to use the definition $\llbracket a, b \rrbracket = [a, b) \ominus [b, a)$. However, when \mathbf{a} and \mathbf{b} are n -tuples (for example, coordinates in \mathbb{R}^n then this is *not* the oriented line interval, $\llbracket \mathbf{a}, \mathbf{b} \rrbracket \ominus \llbracket \mathbf{b}, \mathbf{a} \rrbracket$ rather we define it as follows:

Definition Let $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$ be ordered n -tuples then we use the notation:

$$\llbracket \mathbf{a}, \mathbf{b} \rrbracket = \llbracket a_1, b_1 \rrbracket \times \llbracket a_2, b_2 \rrbracket \times \dots \times \llbracket a_n, b_n \rrbracket \quad (3.20)$$

The dimension of $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ is equal to the number of indices where a_i and b_i are distinct. For any i where $a_i = b_i$, the corresponding term: $\llbracket a_i, b_i \rrbracket$ will be a hybrid set containing a single point, that is, a 0-rectangle. The orientation of $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ is based on the number of negatively oriented intervals $\llbracket a_i, b_i \rrbracket$. Should there be an odd number of indices i such that $a_i > b_i$ then $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ will also be negatively oriented. Otherwise, it will be positively oriented.

For the remainder of this chapter, we will only be interested in the space $\mathbb{N}_0 \times \mathbb{N}_0$ where there is only enough space for a single Cartesian product. This discussion of higher dimension rectangles will be returned to in Chapter 4 when investigating integration.

vbox

vbox

vbox

3.4 Matrix Addition

Now we will consider the addition of 2×2 block matrices A and B with overall dimensions $n \times m$ of the form:

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array} \right] \quad \text{and} \quad B = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array} \right]$$

Since these are block matrices then A_{ij} and B_{ij} are not entries but sub matrices themselves. We shall assume that A_{11} is a $(q \times r)$ matrix and B_{11} is a $(s \times t)$ matrix. The sum of A and B will also be a $n \times m$ matrix. Our universe, \mathcal{U} is therefore the set of all indices in an $n \times m$ matrix:

$$\mathcal{U} = \llbracket 0, n \rrbracket_{\mathbb{N}_0} \times \llbracket 0, m \rrbracket_{\mathbb{N}_0} = \{(i, j) \mid 0 \leq i < n \text{ and } 0 \leq j < m \text{ and } i, j \in \mathbb{N}_0\}$$

First we must convert A and B to hybrid function notation. We will use \mathcal{A}_{ij} and \mathcal{B}_{ij} to respectively denote the regions for which A_{11} and B_{ij} are defined. Explicitly,

$$\begin{aligned} \mathcal{A}_{11} &= \llbracket 0, q \rrbracket \times \llbracket 0, r \rrbracket & \mathcal{A}_{12} &= \llbracket 0, q \rrbracket \times \llbracket r, m \rrbracket & \mathcal{A}_{21} &= \llbracket q, n \rrbracket \times \llbracket 0, r \rrbracket & \mathcal{A}_{22} &= \llbracket q, n \rrbracket \times \llbracket r, m \rrbracket \\ \mathcal{B}_{11} &= \llbracket 0, s \rrbracket \times \llbracket 0, t \rrbracket & \mathcal{B}_{12} &= \llbracket 0, s \rrbracket \times \llbracket t, m \rrbracket & \mathcal{B}_{21} &= \llbracket s, n \rrbracket \times \llbracket 0, t \rrbracket & \mathcal{B}_{22} &= \llbracket s, n \rrbracket \times \llbracket t, m \rrbracket \end{aligned}$$

Which will allow us to rewrite A and B as:

$$\begin{aligned} A &= A_{11}^{\mathcal{A}_{11}} \oplus A_{12}^{\mathcal{A}_{12}} \oplus A_{21}^{\mathcal{A}_{21}} \oplus A_{22}^{\mathcal{A}_{22}} \\ B &= B_{11}^{\mathcal{B}_{11}} \oplus B_{12}^{\mathcal{B}_{12}} \oplus B_{21}^{\mathcal{B}_{21}} \oplus B_{22}^{\mathcal{B}_{22}} \end{aligned}$$

Depending on the relation of q with s and r with t the regions in the sum of A and B may vary. In Figure 3.1, the shapes of block matrices that can arise are shown. Intuitively, the approach we will take is to not concern ourselves with all possible cases that *could* arise but to just choose one ordering. If this ordering is wrong, then the hybrid function multiplicities will handle cancellations to yield the correct expression regardless.

Since there are 4 partitions in A and 4 partitions in B , we only require 7 pieces to form a common refinement. To this refinement for, we follow the same method as in the previous chapter.

$$\mathcal{A}_{11}, \mathcal{A}_{12}, \mathcal{A}_{21}, \mathcal{B}_{11}, \mathcal{B}_{12}, \mathcal{B}_{21}, \mathcal{P} \quad (3.21)$$

where \mathcal{P} is defined as, $\mathcal{P} = \mathcal{U} \ominus (\mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21} \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21})$. Clearly we can still express \mathcal{A}_{22} using only the terms in (3.21) by

$$\begin{aligned} \mathcal{A}_{22} &= \mathcal{U} \ominus (\mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21}) \\ &= \mathcal{U} \ominus (\mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21} \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21}) \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21} \\ &= \mathcal{P} \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21} \end{aligned}$$

Similarly \mathcal{B}_{22} can be represented as $\mathcal{B}_{22} = \mathcal{P} \oplus \mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21}$ and \mathcal{U} as the sum of all 7 regions, $\mathcal{U} = \mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21} \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21} \oplus \mathcal{P}$. Thus A and B can be rewritten using this new generalized partition as:

$$\begin{aligned} A &= A_{11}^{\mathcal{A}_{11}} \oplus A_{12}^{\mathcal{A}_{12}} \oplus A_{21}^{\mathcal{A}_{21}} \oplus A_{22}^{\mathcal{P} \oplus \mathcal{B}_{11} \oplus \mathcal{B}_{12} \oplus \mathcal{B}_{21}} \\ B &= B_{11}^{\mathcal{B}_{11}} \oplus B_{12}^{\mathcal{B}_{12}} \oplus B_{21}^{\mathcal{B}_{21}} \oplus B_{22}^{\mathcal{P} \oplus \mathcal{A}_{11} \oplus \mathcal{A}_{12} \oplus \mathcal{A}_{21}} \end{aligned}$$

And addition becomes straightforward. We add functions for terms over corresponding regions. Since we are using *generalized partitions*, not traditional partitions we cannot guarantee disjointness. As such we must also apply a +-reduction after summing each matching pair:

$$\begin{aligned} (A + B) &= \mathcal{R}_+ \left((A_{11} + B_{22})^{\mathcal{A}_{11}} \oplus (A_{12} + B_{22})^{\mathcal{A}_{12}} \oplus (A_{21} + B_{22})^{\mathcal{A}_{21}} \right. \\ &\quad \oplus (A_{22} + B_{11})^{\mathcal{B}_{11}} \oplus (A_{22} + B_{12})^{\mathcal{B}_{12}} \oplus (A_{22} + B_{21})^{\mathcal{B}_{21}} \\ &\quad \left. \oplus (A_{22} + B_{22})^{\mathcal{P}} \right) \end{aligned}$$

3.4.1 Example: *Evaluation at points*

We will now demonstrate evaluating this expression. Let us assume a point (i, j) exists in the region $\mathcal{A}_{11} \cap \mathcal{B}_{12}$. That is, $0 \leq i < \min(q, s)$ and $t \leq j < r$. Evaluating each of the hybrid sets from (3.21) we find that only three have non-zero multiplicities: $\mathcal{A}_{11}(i, j) = 1$, $\mathcal{B}_{12} = 1$ and $\mathcal{P}(i, j) = 1 - (1 + 0 + 0 + 0 + 1 + 0) = -1$. After removing all zero terms, this yields:

$$\begin{aligned} (A + B)(i, j) &= \mathcal{R}_+ \left((A_{11} + B_{22})^1 \oplus (A_{22} + B_{12})^1 \oplus (A_{22} + B_{22})^{-1} \right) \\ &= (A_{11} + B_{22}) + (A_{22} + B_{12}) - (A_{22} + B_{22})(i, j) \\ &= (A_{11} + B_{12})(i, j) \end{aligned}$$

As a second example assume $(i, j) \in \mathcal{A}_{22} \cap \mathcal{B}_{12}$. Then we find there is only one partition with non-zero multiplicity. Clearly $\mathcal{B}_{12} = 1$ but $\mathcal{A}_{22} \notin (3.21)$. Calculating the multiplicity of \mathcal{P} also yields $1 - (0 + 0 + 0 + 0 + 1 + 0) = 0$. Very simply:

$$\begin{aligned} (A + B)(i, j) &= \mathcal{R}_+ \left((A_{22} + B_{12})^1 \right)(i, j) \\ &= (A_{22} + B_{12})(i, j) \end{aligned}$$

3.4.2 Larger block matrices

This method extends easily from addition of two 2×2 block matrices to arbitrary addition of block matrices. If we consider (*conformable*) $k \times \ell$ and $n \times m$ block matrices A and B respectively of the form:

$$A = \begin{bmatrix} A_{11} & \dots & A_{1\ell} \\ \vdots & & \vdots \\ A_{k1} & \dots & A_{k\ell} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & \dots & B_{1m} \\ \vdots & & \vdots \\ B_{n1} & \dots & B_{nm} \end{bmatrix}$$

For matrices to be conformable for addition they must have the same dimensions. So we can partition the rows of A by the strictly increasing sequence $\{q_i\}_{i=0}^k$ and the columns by $\{r_j\}_{j=0}^\ell$.

Similarly for B we partition the rows by $\{s_i\}_{i=0}^n$ and the columns by $\{t_j\}_{j=0}^m$. With the additional constraints that $q_0 = r_0 = s_0 = t_0 = 0$ and $q_k = s_n$ and $r_\ell = t_m$. Each A_{ij} and B_{ij} is defined over a rectangular region \mathcal{A}_{ij} and \mathcal{B}_{ij} :

$$\mathcal{A}_{ij} = \llbracket q_{i-1}, q_i \rrbracket \times \llbracket r_{j-1}, r_j \rrbracket \quad \mathcal{B}_{ij} = \llbracket s_{i-1}, s_i \rrbracket \times \llbracket t_{j-1}, t_j \rrbracket$$

which gives the expression:

$$(A + B) = \mathcal{R}_+ \left(\left(\bigoplus_{(i,j) \neq (n,m)} (A_{ij} + B_{nm})^{\mathcal{A}_{ij}} \right) \oplus \left(\bigoplus_{(i,j) \neq (n,m)} (A_{nm} + B_{ij})^{\mathcal{B}_{ij}} \right) \oplus (A_{nm} + B_{nm})^{\mathcal{P}} \right) \quad (3.22)$$

3.5 Matrix Multiplication

Next we will consider the product of symbolic block matrices. Again, we will assume 2×2 block matrices A and B . However for these matrices to be conformable for multiplication they must be $n \times m$ and $m \times p$ rather than the same size as was required for addition.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad (3.23)$$

Where A_{11} is a $q \times r$ matrix and B_{11} is a $s \times t$ matrix. Note that $0 \leq r, s \leq m$ but the ordering of r and s is unknown.

In the simplest case, $r = s$, four regions will arise each with simple closed expressions.

$$AB = \begin{bmatrix} (A_{11}B_{11} + A_{12}B_{21}) & (A_{11}B_{12} + A_{12}B_{22}) \\ (A_{21}B_{11} + A_{22}B_{21}) & (A_{21}B_{12} + A_{22}B_{22}) \end{bmatrix} \quad (3.24)$$

One should notice the similarity between this and multiplication of simple 2×2 matrices. If we consider only the top-left block, since $r = s$ then the $(q \times r)$ matrix A_{11} and the $(s \times t)$ matrix B_{11} are conformable. As are the $(q \times m - r)$ matrix A_{12} and the $(m - s \times t)$ matrix B_{21} . Both products

will result in a $q \times t$ matrix which are conformable for addition. Thus the term $A_{11}B_{11} + A_{12}B_{21}$ is a $q \times t$ block.

If $r \neq s$ then one approach would be to partition A into a 2×3 block matrix split along the vertical lines r and s and the horizontal line q . And split B into a 3×2 block matrix split along the vertical line t and the horizontal lines r and s : Depending on the relative ordering of r and s this may cause different blocks to be split. If $s < r$ then A_{11} and A_{21} will be split into blocks with columns from 0 to s and then from s to r while B_{21} and B_{22} would be split into blocks with rows from s to r and from r to m .

$$A = \left[\begin{array}{cc|c} A_{11}^{(1)} & A_{11}^{(2)} & A_{12} \\ \hline A_{21}^{(1)} & A_{21}^{(2)} & A_{22} \end{array} \right] \quad \text{and} \quad B = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21}^{(1)} & B_{22}^{(1)} \\ B_{21}^{(2)} & B_{22}^{(2)} \end{array} \right]$$

The resulting product is still a 2×2 matrix. Additionally, each block is still the same size; the first block in the top-left is still $q \times t$. However each block is now the sum of three block products:

$$AB = \left[\begin{array}{cc} (A_{11}^{(1)}B_{11} + A_{11}^{(2)}B_{21}^{(1)} + A_{12}B_{21}^{(2)}) & (A_{11}^{(1)}B_{12} + A_{11}^{(2)}B_{22}^{(1)} + A_{12}B_{22}^{(2)}) \\ (A_{21}^{(1)}B_{11} + A_{21}^{(2)}B_{21}^{(1)} + A_{22}B_{21}^{(2)}) & (A_{21}^{(1)}B_{12} + A_{21}^{(2)}B_{22}^{(1)} + A_{22}B_{22}^{(2)}) \end{array} \right]$$

On the other hand, if $r < s$ then A_{12} and A_{22} will be the blocks split vertically while B_{11} and B_{12} will be split horizontally. In turn, this leads to a different expression for the product of A and B . In a now familiar, pattern we can use hybrid functions to give a single expression to deal with all permutations simultaneously.

First we shall refer to the product AB by the block matrix C :

$$AB = C = \left[\begin{array}{cc} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array} \right] \quad (3.25)$$

C is an $n \times p$ matrix as determined by the sizes of A and B and C_{11} is a $q \times t$ sub-matrix. This

leaves C_{12} , C_{21} and C_{22} to be $q \times (p - t)$, $(n - q) \times t$ and $(n - q) \times (p - t)$ respectively. We will partition all three matrices along the axes $0..n$, $0..p$ and $0..m$ into the oriented intervals:

$$N_1 = \llbracket 0, q \rrbracket \quad N_2 = \llbracket q, n \rrbracket$$

$$P_1 = \llbracket 0, t \rrbracket \quad P_2 = \llbracket t, p \rrbracket$$

$$M_1 = \llbracket 0, r \rrbracket \quad M_2 = \llbracket r, s \rrbracket \quad M_3 = \llbracket s, m \rrbracket$$

Assumption is too strong a word, but these partitions follow the *guess* that $r < s$. So we will be constructing expressions with this in mind. If we chose incorrectly, then we plan to use the negative orientation of M_2 to correct our expression. Using these intervals, we can now rewrite our matrices inline as:

$$A = A_{11}^{N_1 \times M_1} \oplus A_{12}^{N_1 \times (M_2 \oplus M_3)} \oplus A_{21}^{N_2 \times M_1} \oplus A_{22}^{N_2 \times (M_2 \oplus M_3)} \quad (3.26)$$

$$B = B_{11}^{(M_1 \oplus M_2) \times P_1} \oplus B_{12}^{(M_1 \oplus M_2) \times P_2} \oplus B_{21}^{M_3 \times P_1} \oplus B_{22}^{M_3 \times P_2} \quad (3.27)$$

$$C = C_{11}^{N_1 \times P_1} \oplus C_{12}^{N_1 \times P_2} \oplus C_{21}^{N_2 \times P_1} \oplus C_{22}^{N_2 \times P_2} \quad (3.28)$$

It should be noted here that \oplus is still the point-wise sum of hybrid functions. It should not be confused with the direct sum nor the Kronecker sum of matrices which both use the same \oplus operator. The \times operator refers to the Cartesian product of intervals.

For $i, j \in \{1, 2\}$ the terms of C are given by.

$$C_{i,j} = A_{i,1}^{N_i \times M_1} B_{1,j}^{M_1 \times P_j} + A_{i,2}^{N_i \times M_2} B_{1,j}^{M_2 \times P_j} + A_{i,2}^{N_i \times M_3} B_{2,j}^{M_3 \times P_j} \quad (3.29)$$

If $r = s$ then $M_2 = \emptyset$. We can think of multiplying a $n \times 0$ matrix by a $0 \times p$ matrix as giving a $n \times p$ matrix which is the sum over an empty set, 0 everywhere. If $r < s$ then this is like treating A and B instead as 2×3 and 3×2 block matrices. M_1 , M_2 and M_3 are all positive intervals and form conformable blocks.

If $r > s$ then M_2 is a negative interval. The product $A_{i,2}^{N_i \times M_2} B_{1,j}^{M_2 \times P_j}$ may be symbolically conformable but it begs the question of what exactly a (2×-3) or (-1×4) matrix would even look like. To simplify $C_{i,j}$, we can use the generalized partition:

$$\left\{ (M_1 \oplus M_2), \ominus M_2, (M_2 \oplus M_3) \right\}$$

which contains only positive oriented intervals. Now we can split the first and third terms using $M_1 = (M_1 \oplus M_2) \ominus M_2$ and $M_3 = \ominus M_2 \oplus (M_2 \oplus M_3)$:

$$\begin{aligned} C_{i,j} = & \left(A_{i,1}^{N_i \times M_1 \oplus M_2} \oplus A_{i,1}^{N_i \times \ominus M_2} \right) \left(B_{1,j}^{M_1 \oplus M_2 \times P_j} \oplus B_{1,j}^{\ominus M_2 \times P_j} \right) + A_{i,2}^{N_i \times M_2} B_{1,j}^{M_2 \times P_j} \\ & + \left(A_{i,2}^{N_i \times \ominus M_2} \oplus A_{i,2}^{N_i \times M_3 \oplus M_2} \right) \left(B_{2,j}^{\ominus M_2 \times P_j} \oplus B_{2,j}^{M_3 \oplus M_2 \times P_j} \right) \end{aligned}$$

These resulting hybrid function sums are themselves inline block matrices! The products, $\left(A_{i,1}^{N_i \times M_1 \oplus M_2} \oplus A_{i,1}^{N_i \times \ominus M_2} \right) \left(B_{1,j}^{M_1 \oplus M_2 \times P_j} \oplus B_{1,j}^{\ominus M_2 \times P_j} \right)$ and $\left(A_{i,1}^{N_i \times M_1 \oplus M_2} \oplus A_{i,1}^{N_i \times \ominus M_2} \right) \left(B_{1,j}^{M_1 \oplus M_2 \times P_j} \oplus B_{1,j}^{\ominus M_2 \times P_j} \right)$ are each conformable 1×2 and 2×1 block matrices. Their product is a 1×1 block matrix given by a sum of two $N_i \times P_j$ matrices.

$$\begin{aligned} C_{i,j} = & \left(A_{i,1}^{N_i \times M_1 \oplus M_2} B_{1,j}^{M_1 \oplus M_2 \times P_j} + A_{i,1}^{N_i \times \ominus M_2} B_{1,j}^{\ominus M_2 \times P_j} \right) + A_{i,2}^{N_i \times M_2} B_{1,j}^{M_2 \times P_j} \\ & + \left(A_{i,2}^{N_i \times \ominus M_2} B_{2,j}^{\ominus M_2 \times P_j} + A_{i,2}^{N_i \times M_3 \oplus M_2} B_{2,j}^{M_3 \oplus M_2 \times P_j} \right) \end{aligned}$$

Rearranging the braces, we can recombine the middle three terms to form the product of 1×3 and 3×1 block matrices:

$$\begin{aligned} C_{i,j} = & \left(A_{i,1}^{N_i \times \ominus M_2} \oplus A_{i,2}^{N_i \times M_2} \oplus A_{i,2}^{N_i \times \ominus M_2} \right) \left(B_{1,j}^{\ominus M_2 \times P_j} \oplus B_{1,j}^{M_2 \times P_j} \oplus B_{2,j}^{\ominus M_2 \times P_j} \right) \\ & + A_{i,1}^{N_i \times M_1 \oplus M_2} B_{1,j}^{M_1 \oplus M_2 \times P_j} + A_{i,2}^{N_i \times M_3 \oplus M_2} B_{2,j}^{M_3 \oplus M_2 \times P_j} \end{aligned}$$

But the term $A_{i,2}$ occurs twice; once over $N_i \times M_2$ and once over $N_i \times \ominus M_2$. As hybrid functions with the same corresponding function, we can combine them by $f^A \oplus f^B = f^{A \oplus B}$. But the

regions only differ by a sign and sum to the empty set. Similarly, the $B_{1,j}$ terms also cancel:

$$C_{i,j} = \left(A_{i,1}^{N_i \times \ominus M_2} \oplus A_{i,2}^{N_i \times \emptyset} \right) \left(B_{2,j}^{\ominus M_2 \times P_j} \oplus B_{1,j}^{\emptyset \times P_j} \right) \\ + A_{i,1}^{N_i \times M_1 \oplus M_2} B_{1,j}^{M_1 \oplus M_2 \times P_j} + A_{i,2}^{N_i \times M_3 \oplus M_2} B_{2,j}^{M_3 \oplus M_2 \times P_j}$$

And finally cleaning up this expression by removing the terms over empty regions gives us the desired:

$$C_{i,j} = A_{i,1}^{N_i \times M_1 \oplus M_2} B_{1,j}^{M_1 \oplus M_2 \times P_j} + A_{i,1}^{N_i \times \ominus M_2} B_{2,j}^{\ominus M_2 \times P_j} + A_{i,2}^{N_i \times M_3 \oplus M_2} B_{2,j}^{M_3 \oplus M_2 \times P_j} \quad (3.30)$$

3.5.1 Larger block matrices

The most difficult part of extending block matrix multiplication to larger block matrices turns out to be the number of variables needed. Once again we will use N_i to divide the rows of blocks in A and P_j to divide the block columns of B .

$$C = \bigoplus_{i \in I} \bigoplus_{j \in J} C_{i,j}^{N_i \times P_j} \quad (3.31)$$

Where each $C_{i,j}$ is defined as:

$$C_{i,j} = \left(\sum_{k \in K/last} A_{i,k}^{N_i \times M_k} B_{last,j}^{M_k \times P_j} \right) + \left(\sum_{k \in K'/last} A_{i,last}^{N_i \times M_k} B_{k,j}^{M_k \times P_j} \right) + A_{i,last}^{N_i \times M_{U \oplus all}} B_{last,j}^{M_{U \oplus all} \times P_j} \quad (3.32)$$

Unfortunately these formulation for multiplication is not the easiest to work with. The arithmetic used to convert any negatively oriented matrices to all positive ones is *required* before the equation can be interpreted. Another approach would be to use:

$$C_{i,j} = \sum_{\llbracket 0,m \rrbracket} \mathcal{R}_\times \left(A_1^{M_1 \oplus M_2} \oplus A_2^{M_3} \oplus B_1^{M_1} \oplus B_2^{M_2 \oplus M_3} \right)$$

Chapter 4

Integration

Given a function f with real variable x and an interval $[a, b)$ on the (extended) real line, a traditional **definite integral** would be of the form:

$$\int_a^b f(x) \, dx \quad \text{or} \quad \int_{[a,b)} f(x) \, dx$$

Which we interpret as the signed area bounded by f between $x = a$ and $x = b$. However, defining this definite integral using (unoriented) intervals like this is a bit of a misnomer. In the case where $a \geq b$ it is typical to use the identity:

$$\int_a^b f(x) \, dx = - \int_b^a f(x) \, dx \tag{4.1}$$

to evaluate the integral. But, as we saw in the previous chapter, when $a \geq b$, the interval $[a, b)$ is the empty set, hence we *cannot* make the similar translation to

$$\int_{[a,b)} f(x) \, dx = - \int_{[b,a)} f(x) \, dx \tag{4.2}$$

Using sets as domains of integration like this generally appears in context of Lebesgue integrals, but Riemann integrals (which generally use $\int_a^b f(x) \, dx$ instead) often just hide their

mis-use of intervals. For example, it is typically glossed over that when doing the formal Riemann sum for an integral $\int_a^b f(x) dx$, one would use the tagged partition given as a series x_i such that $a = x_1 < x_2 < \dots < x_n = b$.

(citations
needed)

Another advantage to using oriented sets is a more natural language for manipulating domains of integration than sets. There is no common understanding of the sum of two sets, but since we have the point-wise sum \oplus for hybrid sets, we simply say that the integral operator is *bi-linear*. By this we mean, it is linear with respect to both operands: the function it is integrating over:

$$\int_X f(x) + g(x) dx = \int_X f(x) dx + \int_X g(x) dx \quad (4.3)$$

and the domain of integration:

$$\int_{X \oplus Y} f(x) dx = \int_X f(x) dx + \int_Y f(x) dx \quad (4.4)$$

By definition, this immediately gives way to the very useful identities:

$$\int_{\llbracket a, c \rrbracket} f(x) dx = \int_{\llbracket a, b \rrbracket} f(x) dx + \int_{\llbracket b, c \rrbracket} f(x) dx \quad (4.5)$$

$$\int_{\llbracket a, b \rrbracket} f(x) dx = \int_{\ominus \llbracket b, a \rrbracket} f(x) dx = - \int_{\llbracket b, a \rrbracket} f(x) dx \quad (4.6)$$

for all a, b and c .

In one dimension, many of these changes may seem trivial advances but in higher dimensions, the oriented and measure-theoretic approaches diverge. [22] Using hybrid sets as domains of integration allow us to use the best features of both. In this chapter we will present an introduce integration over oriented intervals and generalize to oriented n -cubes in higher dimensions. We will also explore the boundary operator and the cubic homology formed by n -cubes. This will provide a base for the following chapter to investigate the cubic singular homology, integration of forms and Stokes' theorem. In Chapter 6, we will introduce the oriented Lebesgue integral.

4.1 The Lebesgue Integral

Definition Let (X, μ) be a measure space and $S \in \mathcal{L}(X, \mu)$ a measurable set. If \mathbb{I}_S is indicator function $\mathbb{I}_S : X \rightarrow \{0, 1\}$ given by $\mathbb{I}_S(x) = 1$ if $x \in S$ and $\mathbb{I}_S(x) = 0$ otherwise. Then we define:

$$\int_X \mathbb{I}_S d\mu := \mu(S) \quad (4.7)$$

We define a simple function as a function which maps to a finite set.

Theorem 4.1.1 *If φ is a simple function, then we it can be represented as a finite sum of indicator functions:*

$$\varphi := \sum_{k=0}^n a_k \mathbb{I}_{E_k} \quad (4.8)$$

Where $\{a_k\}_{k=0}^n$ is the set of values in the range of φ and $E_k = \varphi^{-1}(\{a_k\})$, the set of all points which map to a_k .

This allows us to define the integral of a simple function.

Definition Let $\varphi = \sum_{k=0}^n a_k \mathbb{I}_{E_k}$ be a simple function. We define the integral:

$$\int_X \varphi d\mu = \int_X \sum_{k=0}^n a_k \mathbb{I}_{E_k} d\mu = \sum_{k=0}^n a_k \int_X \mathbb{I}_{E_k} d\mu \quad (4.9)$$

Next we define integral of a non negative functions in $\bar{\mathbb{R}}$

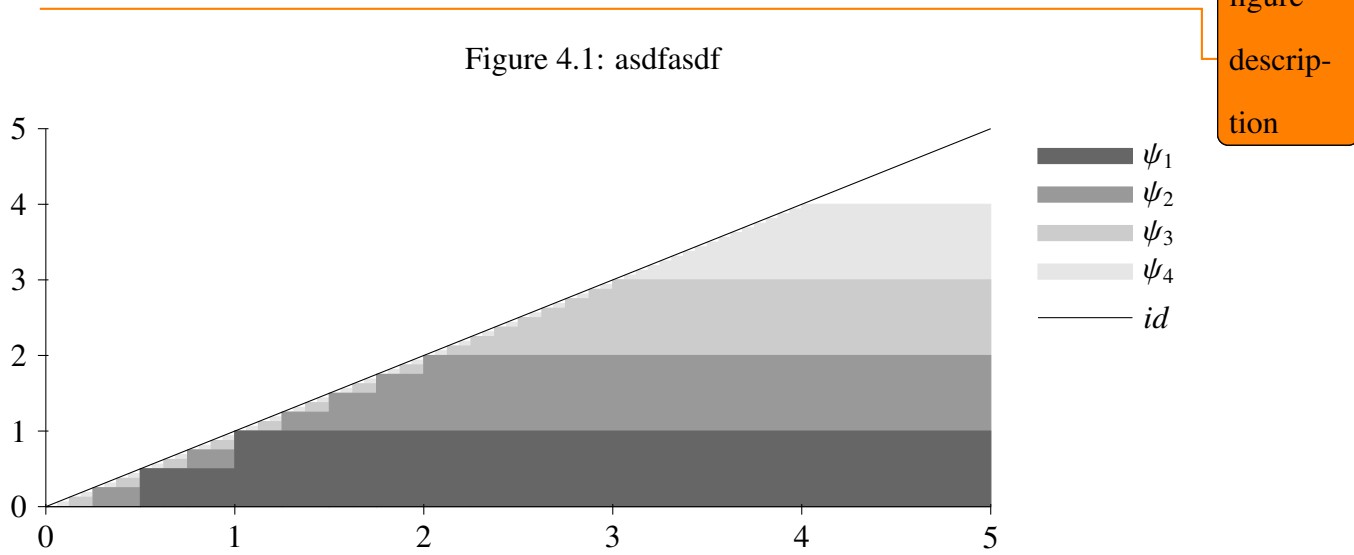
Definition Let (X, μ) be a measure space and $f : X \rightarrow [0, \infty]$.

$$\int_X f d\mu := \sup \left\{ \int_X \varphi : \varphi \text{ simple, and } 0 \leq \varphi \leq f \right\} \quad (4.10)$$

We use the simple function ψ_n defined as:

$$\psi_n = \sum_{k=0}^{n2^n-1} \left[\left(\frac{k}{2^n} \right)^{\left[\frac{k}{2^n}, \frac{k+1}{2^n} \right)} \right] + n^{[n, \infty]} \quad (4.11)$$

Notationally, this definition is rather heavy but is easily understood geometrically as seen in Figure 4.1



Which for any non-negative f allows us to define $\varphi_n = \psi_n \circ f$ a simple function.

Obviously we have φ_n simple since ψ_n is simple.

Since $\psi_n(x) \leq x$ for all x then we also have $0 \leq \varphi_n \leq f$.

Most importantly we have $0 \leq f(x) - \varphi_n(x) \leq 2^{-n}$ and so φ_n , uniformly approaches f as n approaches infinity.

Theorem 4.1.2 *Let f be a function mapping to \mathbb{R} . Then $f = f^+ - f^-$ where f^+ and f^- are non-negative functions given by:*

$$f^+(x) := \max(0, f(x)) \quad (4.12)$$

$$f^-(x) := \max(0, -f(x)) \quad (4.13)$$

Now we have everything we need to define integrals for arbitrary functions mapping to \mathbb{R} :

Definition asdfasd

$$\int_X f d\mu = \int_X f^+ d\mu - \int_X f^- d\mu \quad (4.14)$$

This was a standard treatment of Lebesgue integration.

Mathematically it is very simple to extend this to integrals of hybrid sets over $L(X, \mu)$ by linearity. That is, given $\Gamma \in \mathbb{Z}^{\mathcal{L}(X, \mu)}$:

$$\int_{\Gamma} f d\mu = \sum_{\sigma \in \Gamma} \Gamma(\sigma) \int_{\sigma} f d\mu \quad (4.15)$$

4.2 Riemann Integral on n -cubes

Now that we have oriented n -dimensional cubes, we would like to define the integral over one. For now, we will content ourselves with the Riemann integral and Euclidean volume. More complex domains and other metrics will be handled in later chapters with push-backs and the Lebesgue integral. The volume of an oriented n -cube in \mathbb{R}^n we define to be the product of its side lengths. Formally,

Definition Let $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ be a k -cube in \mathbb{R}^n again with $\mathbf{a} = (a_1, \dots, a_n)$ and $\mathbf{b} = (b_1, \dots, b_n)$. We denote the **volume of $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$** with vol and define it as:

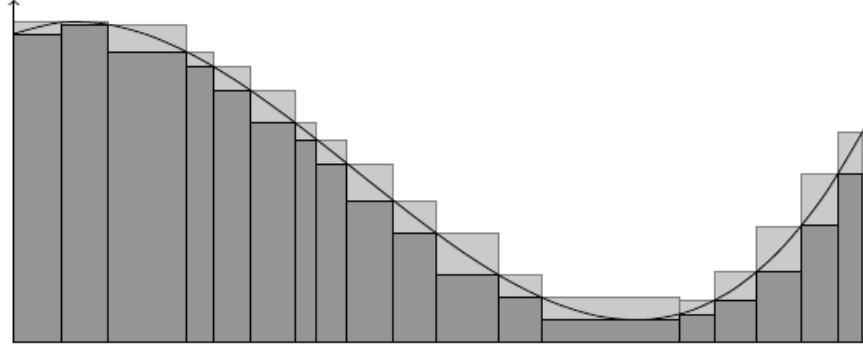
$$\text{vol}(\llbracket \mathbf{a}, \mathbf{b} \rrbracket) = (b_1 - a_1) \cdot (b_2 - a_2) \cdot \dots \cdot (b_n - a_n) \quad (4.16)$$

For any $k < n$, a k -cube will have volume zero. In at least one dimension, the cube will be degenerate (i.e. $a_i = b_i$) and so will contribute zero to the product. Additionally, one can also observe that $\text{vol}(\ominus \llbracket \mathbf{a}, \mathbf{b} \rrbracket) = -\text{vol}(\llbracket \mathbf{a}, \mathbf{b} \rrbracket)$.

Given an n -cube $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ we must cut each $\llbracket a_i, b_i \rrbracket$ into partitions. Previously we used generalized partitions and did not mind if pieces overlapped or exceeded the original range. However, for building our Riemann sums, we are only interested in partitions in the traditional, non-intersecting sense.

Definition Given an oriented interval $\llbracket a, b \rrbracket$ of the real line, we say that a partition of $\llbracket a, b \rrbracket$, $\{P_i\}_{i=1}^n$ is an **interval partition of $\llbracket a, b \rrbracket$** if its pieces are:

Figure 4.2: Upper and lower Riemann sums shown for the same sequence shown with light and dark rectangles respectively. A function over an oriented interval is Riemann integrable if the two sums converge.



1. *Oriented intervals*: P_i is an oriented interval of the real line for all i .
2. *Disjoint*: $P_i \otimes P_j = \emptyset$ for all i, j

We denote the set of all such partition as $\mathcal{P}[[a, b]]$.

This greatly restricts the types of partitions we have access to. Every interval partition will be — up to substitution of “ $]$,” “ $($ ” in place of “ $]$,” “ $($ ” — of the form:

$$\left\{ [[a, x_1)), [[x_1, x_2)), [[x_2, x_3)), \dots, [[x_{n-1}, b]] \right\} \quad (4.17)$$

where x_i is a monotone sequence (that is, either non-increasing or non-decreasing). This is not to say that $P_i = [[x_{i-1}, x_i))$ as the pieces of P may not be given in this order. Regardless of the ordering, we select partitions $P^j \in \mathcal{P}[[a^j, b^j))$ for each dimension of $[[a, b))$. To build our mesh, we construct smaller n -cubes I_{i_1, \dots, i_n} using the Cartesian product of pieces:

$$I_{i_1, \dots, i_n} = i_1 \times \dots \times i_n \quad (4.18)$$

where each i_j is taken from P^j . We are now ready to construct Riemann sums.

Definition Given $P = \{P_j\}_{j=1}^n$ where $P_j \in \mathcal{P}[[a_j, b_j]]$, and $f : [[a, b]] \rightarrow \mathcal{R}$ then we define a

Riemann sum f_P to be:

$$f_P = \sum_{i_1 \in P_1} \cdots \sum_{i_n \in P_n} f(x_{i_1, \dots, i_n}) \text{vol}(I_{i_1, \dots, i_n}) \quad (4.19)$$

where x_{i_1, \dots, i_n} is any point chosen from I_{i_1, \dots, i_n} .

Note that we specify *a* Riemann sum, not *the* Riemann sum. There are several ways to choose $x_{i_1, \dots, i_n} \in I_{i_1, \dots, i_n}$ and different samplings can lead to different Riemann sums for the same partition and same function. In \mathbb{R}^1 , several common ways to sample include the left and right Riemann sums (which sample at the left and right boundaries), the trapezoidal Riemann sum (which averages the left and right) the upper Riemann sum (which samples at $\max(f(x_{i_1, \dots, i_n}))$) and the lower Riemann sum (which samples at $\min(f(x_{i_1, \dots, i_n}))$).

Recall our discussion of refinements from Chapter 2. Given 2 partitions P and P' of the same set then we say $P \leq P'$ if P' is a refinement of P . In this way we can induce a partial ordering on $\mathcal{P}[[a, b]]$. There is a unique smallest element in this partial ordering which is $[[a, b]]$ itself; every partition by definition will refine $[[a, b]]$. Additionally, given $P, P' \in \mathcal{P}[[a, b]]$ then propose that we can always find Q that simultaneously refines both. If $P = P'$ then trivially $Q = P$ is a common refinement. Otherwise, we take:

$$Q = [[a, b]] \otimes \left(\bigoplus_{p \in P} \bigoplus_{p' \in P'} p \otimes p' \right) \quad (4.20)$$

The heavy lifting here is done by the fact that every point in $[[a, b]]$ is covered by exactly one $p \in P$ and exactly one $p' \in P$. The product of any two partition pieces will then be the smallest common interval with multiplicity 1. Multiplying the entire thing by $[[a, b]]$ is done to correct the sign. As we go up in our ordering, our mesh becomes increasingly fine. Taking the Riemann sum of the suprema of this poset gives us the Riemann integral.

Definition The Riemann integral of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ over a k -cube $[[a, b]]$ where $P = \sup \{ \mathcal{P}[[a, b]] \}$

$$\int_{[[a, b]]} f(x) dx = f_P \quad (4.21)$$

We say that a function is **Riemann integrable** if the upper and lower Riemann sums converge.

4.3 Boundary Operator

In one dimension, the boundary of an interval was quite straight-forward. For a positively oriented interval, the boundary was composed of two points; the right end-point was positive and the left end-point was negative. From the perspective of k -rectangles, the ∂ operator has mapped an oriented 1-rectangle to a set of oriented 0-rectangles. We will now generalize the boundary to map an oriented n -rectangle to an $(n - 1)$ -rectangle.

Definition Let $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ be a k -rectangle in \mathbb{R}^n . Additionally, let i_1, i_2, \dots, i_k be the unique non-decreasing sequence of indices such that $a_{i_j} \neq b_{i_j}$. The **boundary of $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$** , denoted the operator ∂ is given by

$$\begin{aligned} \partial(\llbracket \mathbf{a}, \mathbf{b} \rrbracket) = \bigoplus_{j=1}^k (-1)^j \left(\llbracket (\mathbf{a}^{\llbracket 1, n \rrbracket}), \quad (\mathbf{b}^{\llbracket 1, i_j \rrbracket}) \oplus \mathbf{a}^{\llbracket i_j \rrbracket} \oplus \mathbf{b}^{\llbracket (i_j, n \rrbracket)} \rrbracket \right. \\ \left. \ominus \llbracket (\mathbf{a}^{\llbracket 1, i_j \rrbracket}) \oplus \mathbf{b}^{\llbracket i_j \rrbracket} \oplus \mathbf{a}^{\llbracket (i_j, n \rrbracket)}, \quad (\mathbf{b}^{\llbracket 1, i_j \rrbracket}) \rrbracket \right) \end{aligned} \quad (4.22)$$

The above equation will require a bit of unpacking to digest featuring oriented intervals in two different contexts. The first appears in the superscripts of \mathbf{a} and \mathbf{b} . The intervals $\llbracket 1, i_j \rrbracket$ and $\llbracket (i_j, n \rrbracket$ are and is an interval over vector indices just as in Chapter 3. Thus, the term $\mathbf{a}^{\llbracket 1, i_j \rrbracket}$ refers to the vector $(a_1, a_2, \dots, a_{i_j-1})$ while the term $\mathbf{b}^{\llbracket (i_j, n \rrbracket}$ refers to $(b_{i_j+1}, b_{i_j+2}, \dots, b_n)$. This provides a compact notation to partition the original range of indices into 3 pieces: $\llbracket 1, i_j \rrbracket$, $\llbracket i_j \rrbracket$, and $\llbracket (i_j, n \rrbracket$. Formally, we are actually using the hybrid sets $\llbracket (i_j)^1 \rrbracket$ but we omit multiplicity of one (and will continue to do so through out the section) to lighten notation.

Next we use the pointwise sum \oplus we reconstruct n -dimensional vectors from our pieces. We then construct a $(k - 1)$ -rectangle using these vectors as in (4.8). Hence we will have terms

of the forms:

$$[[a_1, b_1]] \times \dots \times [[a_{i_{j-1}}, b_{i_{j-1}}]] \times [[a_{i_j}, a_{i_j}]] \times [[a_{i_{j+1}}, b_{i_{j+1}}]] \times \dots \times [[a_n, b_n]] \quad (4.23)$$

$$[[a_1, b_1]] \times \dots \times [[a_{i_{j-1}}, b_{i_{j-1}}]] \times [[b_{i_j}, b_{i_j}]] \times [[a_{i_{j+1}}, b_{i_{j+1}}]] \times \dots \times [[a_n, b_n]] \quad (4.24)$$

In each Cartesian product, the terms at i_j : $[[a_{i_j}, a_{i_j}]]$ and $[[b_{i_j}, b_{i_j}]]$ are both 0-cubes. Since we defined the sequence i_j by $a_{i_j} \neq b_{i_j}$, these 0-rectangles are replacing 1-cubes in $[[a, b]]$. Hence we are indeed left with a $(k - 1)$ -cube.

4.3.1 Example: Boundary of a 1-rectangle

Let $a = (a_1)$ and $b = (b_1)$ be trivial 1-tuples. Then $[[a, b]] = [[a_1, b_1]]$. It follows that:

$$\begin{aligned} \partial([a, b]) &= (-1)^i ([[a^{(1,1)}, b^{(1,1)}]] \times \{a_1\} \times [[a^{(1,1)}, b^{(1,1)}]]) \\ &\quad \ominus [[a^{(1,1)}, b^{(1,1)}]] \times \{b_1\} \times [[a^{(1,1)}, b^{(1,1)}]]) \\ &= \ominus [[a^0, b^0]] \times \{a_1\} \times [[a^0, b^0]] \\ &\quad \oplus [[a^0, b^0]] \times \{b_1\} \times [[a^0, b^0]] \\ &= \{b_1\} \ominus \{a_1\} \\ &= \{a^{-1}, b^1\} \end{aligned}$$

One may notice a relationship between this result and the fundamental theorem of calculus:

$$\int_a^b F'(x) dx = F(b) - F(a) \quad (4.25)$$

Which one could easily rewrite as $\int_{[[a, b]]} F'(x) dx = \sum(\partial([a, b]))$. Indeed, this is why we have defined the boundary function as such, but more general statements await. We defined the boundary for not just intervals on \mathbb{R} but k -cubes in \mathbb{R}^n .

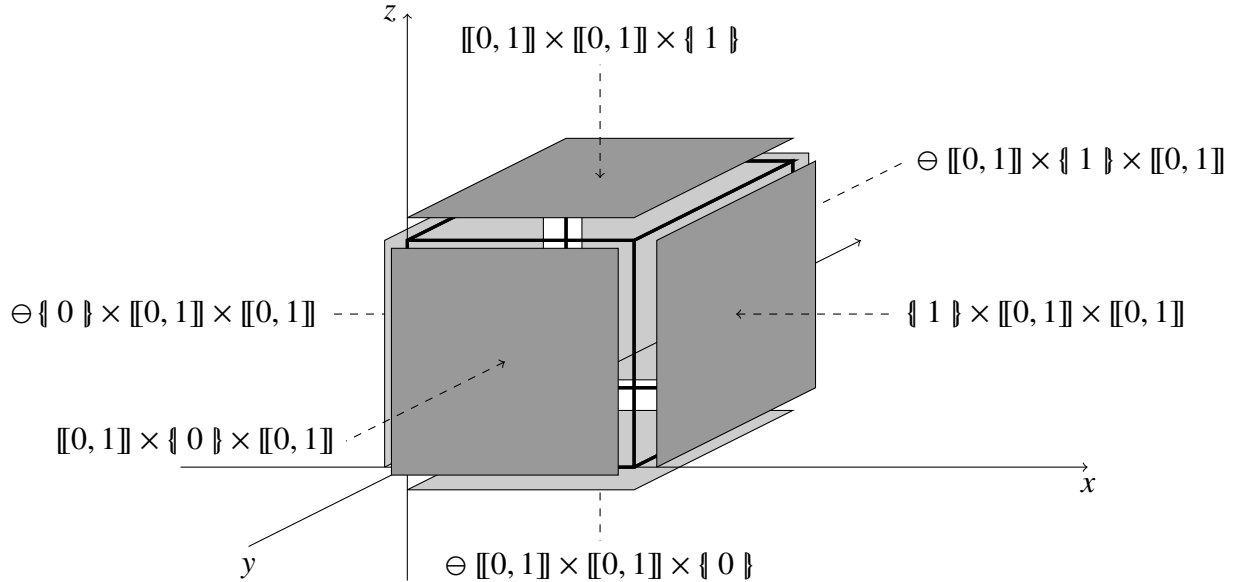
4.3.2 Example: *Boundary of a 3-rectangle*

Let $\mathbf{a} = (0, 0, 0)$ and $\mathbf{b} = (1, 1, 1)$. Omitting the intermediate step, we find the boundary of $[[\mathbf{a}, \mathbf{b}]]$ to be:

$$\begin{aligned} \partial([[\mathbf{a}, \mathbf{b}]]) = & \ominus (\{0\} \times [0, 1] \times [0, 1]) \oplus (\{1\} \times [0, 1] \times [0, 1]) \\ & \oplus ([0, 1] \times \{0\} \times [0, 1]) \ominus ([0, 1] \times \{1\} \times [0, 1]) \\ & \ominus ([0, 1] \times [0, 1] \times \{0\}) \oplus ([0, 1] \times [0, 1] \times \{1\}) \end{aligned}$$

This may not be the most enlightening expression on its own. In Figure 4.3 below, the 3-rectangle given by $[[\mathbf{a}, \mathbf{b}]]$ can be seen as a cube in three dimensions. Physically, the 3-rectangle is a solid cube and includes all interior points. The boundary meanwhile are just the rectangular outer faces, which conveniently, there are also six to match the six terms of $\partial[[\mathbf{a}, \mathbf{b}]]$.

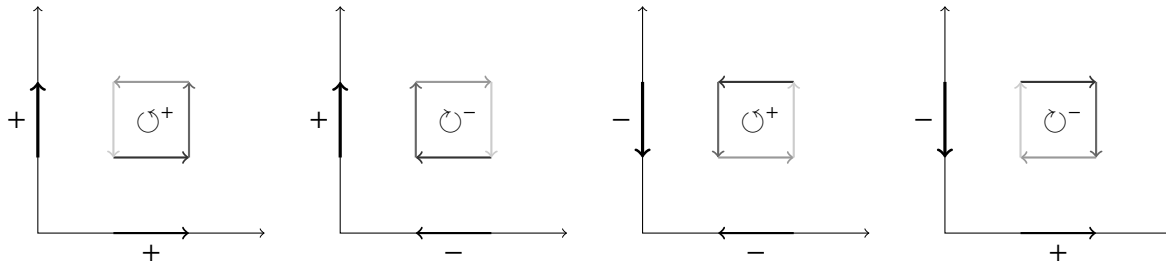
Figure 4.3: The unit cube in \mathbb{R}^3 with positive orientation can be represented as the 3-rectangle: $[(0, 0, 0), (1, 1, 1)]$ is shown as a wire-frame. The six faces that make up its boundary are shaded and labeled with their corresponding terms.



There are several ways to interpret and visualize the \oplus and \ominus sign associated with each face. Most naturally in \mathbb{R}^3 for 2-rectangles is to give each a front and back side with the sign

determining which to use. Alternatively, a 2-rectangle has a boundary formed by 1-rectangles which when drawn as arrows, will all meet head-to-tail. This induces a clockwise or counter-clockwise cycle around the edge of the rectangle and so \circlearrowright and \circlearrowleft are also commonly used. This can be seen in Figure 4.4. One may even notice that the normals produced by both are the same and choose to use that. These are all conceptual tools, which are convenient to use particularly in \mathbb{R}^2 and \mathbb{R}^3 . There may not be such a nice physical interpretation in other spaces.

Figure 4.4: One way of visualizing the orientation of 2-rectangles using clockwise and counter-clockwise cycles of arrows for 1-rectangles. The boundary of $\llbracket a, b \rrbracket \times \llbracket c, d \rrbracket$ becomes the cycle: $(a, c) \rightarrow (b, c) \rightarrow (b, d) \rightarrow (a, d) \rightarrow (a, c)$. Showing the relationship between $\llbracket a, b \rrbracket \times \llbracket c, d \rrbracket$ and $\llbracket b, a \rrbracket \times \llbracket d, c \rrbracket$



4.4 Chains

In fact, we have already seen k -chains without mentioning them explicitly. The boundary of a k -cube was the sum \oplus , of $2k(k-1)$ cubes. Chains do not have to be boundaries however, any linear combination of k -cubes will do.

Definition We denote the Abelian group of all k -cubes in X as $C_k(X)$ (omitting X when obvious by context). An element $c \in C_k(X)$ is called a **k -chain on X** and is of the form:

$$c = \bigoplus_{c_i \in X} \lambda_i c_i \quad (4.26)$$

with integer coefficients λ_i and k -cubes in c_i . If coefficients λ_i are ± 1 and c is *locally finite* (i.e. each c_i intersects with only finitely many c_j that have non-zero coefficients) then we say that c is a **domain of integration**.

Since k -chains are just linear combinations of k -cubes, we naturally extend many of our definitions linearly as well. The integral $\int_c f$ of a k -chain $c = \bigoplus_i \lambda_i c_i$ is defined as $\lambda_1 \int_{c_1} f + \lambda_2 \int_{c_2} f + \dots$. Doing the same for the boundary operator ∂ we have:

$$\begin{aligned}\partial_k : C_k &\rightarrow C_{k-1} \\ \partial_k(c) &= \bigoplus_{i=1}^k \lambda_i \partial_k(c_i)\end{aligned}$$

Elegantly, the boundary operator now maps k -chains to $(k - 1)$ -chains!

$$\dots \xleftarrow{\partial_{k-1}} C_{k-1} \xleftarrow{\partial_k} C_k \xleftarrow{\partial_{k+1}} C_{k+1} \xleftarrow{\partial_{k+2}} \dots \quad (4.27)$$

The most natural next question becomes “*What does $\partial_{k-1}(\partial_k(c))$ look like?*”

4.4.1 Example: *Boundary of a boundary (of a 2-cube)*

Let $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$. We wish to compute $\partial_1(\partial_2(\llbracket \mathbf{a}, \mathbf{b} \rrbracket))$

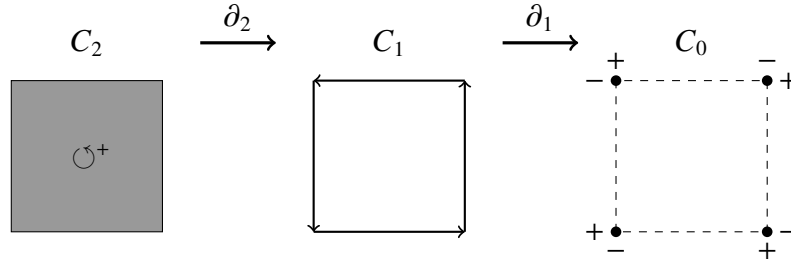
$$\begin{aligned}\partial_1(\partial_2(\llbracket a_1, b_1 \rrbracket \times \llbracket a_2, b_2 \rrbracket)) &= \ominus \partial_1(\llbracket 0 \rrbracket \times \llbracket 0, 1 \rrbracket) \oplus \partial_1(\llbracket 1 \rrbracket \times \llbracket 0, 1 \rrbracket) \\ &\quad \oplus \partial_1(\llbracket 0, 1 \rrbracket \times \llbracket 0 \rrbracket) \ominus \partial_1(\llbracket 0, 1 \rrbracket \times \llbracket 1 \rrbracket)\end{aligned} \quad (4.28)$$

$$\begin{aligned}&= \ominus (\ominus \llbracket (0, 0) \rrbracket \oplus \llbracket (0, 1) \rrbracket) \oplus (\ominus \llbracket (1, 0) \rrbracket \oplus \llbracket (1, 1) \rrbracket) \\ &\quad \oplus (\ominus \llbracket (0, 0) \rrbracket \oplus \llbracket (1, 0) \rrbracket) \ominus (\ominus \llbracket (0, 1) \rrbracket \oplus \llbracket (1, 1) \rrbracket)\end{aligned} \quad (4.29)$$

$$= \emptyset \quad (4.30)$$

When moving from (4.21) to (4.22), in addition to applying ∂_1 we also simplify, $\llbracket x \rrbracket \times \llbracket y \rrbracket = \llbracket (x, y) \rrbracket$. The identity “ $\partial\partial = 0$ ” is not unique to 2-cubes but holds for higher dimensions as well.

Figure 4.5: The boundary of 2-cube gives a cycle of oriented edges. Taking the boundary of again, at each corner, the negative boundary of one edge will be canceled by the positive boundary of the preceding edge.



Let $\llbracket \mathbf{a}, \mathbf{b} \rrbracket$ be a k -rectangle in \mathbb{R}^n . Then we have:

$$\partial_k \partial_{k-1} (\llbracket \mathbf{a}, \mathbf{b} \rrbracket) = \bigoplus_{j=1}^k (-1)^j \left(\partial_{n-1} (\llbracket \mathbf{a}^{\llbracket 1, n \rrbracket}, \mathbf{b}^{\llbracket 1, i_j \rrbracket} \oplus \mathbf{a}^{\llbracket i_j \rrbracket} \oplus \mathbf{b}^{\llbracket i_j, n \rrbracket} \rrbracket) \right. \\ \left. \ominus \partial_{n-1} (\llbracket \mathbf{a}^{\llbracket 1, i_j \rrbracket} \oplus \mathbf{b}^{\llbracket i_j \rrbracket} \oplus \mathbf{a}^{\llbracket i_j, n \rrbracket}, \mathbf{b}^{\llbracket 1, n \rrbracket} \rrbracket) \right) \quad (4.31)$$

$$= \bigoplus_{j=1}^k \bigoplus_{\ell=1}^{k-1} (-1)^{j+\ell} \llbracket (\mathbf{a}^{\llbracket 1, n \rrbracket}), (\mathbf{b}^{\llbracket 1, i_j \rrbracket} \oplus \llbracket i_j, i_{j, \ell} \rrbracket \oplus \llbracket i_{j, \ell}, n \rrbracket \oplus \mathbf{a}^{\llbracket i_j \rrbracket} \oplus \llbracket i_{j, \ell} \rrbracket) \rrbracket \\ \ominus \llbracket (\mathbf{a}^{\llbracket 1, i_{j, \ell} \rrbracket} \oplus \llbracket i_{j, \ell}, n \rrbracket \oplus \mathbf{b}^{\llbracket i_{j, \ell} \rrbracket}), (\mathbf{b}^{\llbracket 1, i_j \rrbracket} \oplus \llbracket i_j, n \rrbracket \oplus \mathbf{a}^{\llbracket i_j \rrbracket}) \rrbracket \\ \ominus \llbracket (\mathbf{a}^{\llbracket 1, i_j \rrbracket} \oplus \llbracket i_j, n \rrbracket \oplus \mathbf{b}^{\llbracket i_j \rrbracket}), (\mathbf{b}^{\llbracket 1, i_{j, \ell} \rrbracket} \oplus \llbracket i_{j, \ell}, n \rrbracket \oplus \mathbf{a}^{\llbracket i_{j, \ell} \rrbracket}) \rrbracket \\ \oplus \llbracket (\mathbf{a}^{\llbracket 1, i_j \rrbracket} \oplus \llbracket i_j, i_{j, \ell} \rrbracket \oplus \llbracket i_{j, \ell}, n \rrbracket \oplus \mathbf{b}^{\llbracket i_j \rrbracket} \oplus \llbracket i_{j, \ell} \rrbracket), (\mathbf{b}^{\llbracket 1, n \rrbracket}) \rrbracket \quad (4.32)$$

Note that we have i_j and i'_ℓ ; after applying the first boundary operator, one dimension of the k -cube is degenerate. Hence for each sequence: $\{i_j\}_{j=1}^k$ we construct $\{i_{j, \ell}\}_{\ell=1}^{k-1}$ given by:

$$i_{j, 1}, \dots, i_{j, k-1} = i_1, \dots, \widehat{i_j}, \dots, i_k \quad (4.33)$$

The double sum iterates over all pairs but \oplus commutes so the $(k-2)$ -cube with degenerate dimensions $\llbracket i_j \rrbracket \oplus \llbracket i_{j, \ell} \rrbracket$ will be iterated over twice. The sequences depend on one another so it

is not as simple as simply swapping ℓ and j :

$$\llbracket i_j \rrbracket \oplus \llbracket i_{j,\ell} \rrbracket = \begin{cases} \llbracket i_\ell \rrbracket \oplus \llbracket i_{\ell,j-1} \rrbracket & j > \ell \\ \llbracket i_{\ell+1} \rrbracket \oplus \llbracket i_{\ell+1,j} \rrbracket & j \leq \ell \end{cases} \quad (4.34)$$

So each term representing a $(k-2)$ -cube will occur twice in the sum. Once with the iteration (j, ℓ) and once with $(\ell, j-1)$ or $(\ell+1, j)$. In either case, $(-1)^{j+\ell}$ is inverted meaning the two cubes will cancel. Leaving us with the boundary of a boundary being empty. By linearity this extends to all chains as well as the sum of empty sets is of course still empty.

Definition A **chain complex** is a sequence of Abelian groups $\dots, A_2, A_1, A_0, A_{-1}, A_{-2}, \dots$ which are connected by homomorphisms $d_n : A_n \rightarrow A_{n-1}$ such that $d_n \circ d_{n+1} = 0$ for all n . Typically written out as:

$$\dots \xleftarrow{d_{k-1}} A_{k-1} \xleftarrow{d_k} A_k \xleftarrow{d_{k+1}} A_{k+1} \xleftarrow{d_{k+2}} \dots \quad (4.35)$$

A **cochain complex** is a sequence of Abelian groups $\dots, A^{-2}, A^{-1}, A^0, A^1, A^2, \dots$ which are connected by homomorphisms $d^n : A^n \rightarrow A^{n+1}$ such that $d^n \circ d^{n-1} = 0$ for all n . Typically written out as:

$$\dots \xrightarrow{d^{k-1}} A^{k-1} \xrightarrow{d^k} A^k \xrightarrow{d^{k+1}} A^{k+1} \xrightarrow{d^{k+2}} \dots \quad (4.36)$$

$(C_\bullet, \partial_\bullet)$ is just one instance of a chain complex known as the “*cubic homology*”. In the next chapter we will look at the more general “*cubic singular homology*”. As well as the related cochain complex: the “*De Rham cohomology*” and how the two relate.

4.5 Differential Forms

Definition A **(differential) 0-form** β on \mathbb{R}^n is a function $\beta : \mathbb{R}^n \rightarrow \mathbb{R}$. A **(differential) 1-form** ω on \mathbb{R}^n is an expression of the form:

$$\omega = f_1(x) dx_1 + f_2(x) dx_2 + \dots + f_n(x) dx_n \quad (4.37)$$

There is very little to say about 0-forms, they are just functions on \mathbb{R}^n . 1-forms look very much like something we're used to integrating with. Of course if all but one of f_i are zero then we have a *basic 1-form*, $\omega = f_i dx_i$ which is nothing new to integrate over. Using the linearity of integration, general 1-forms can easily be separated into a sum of integrals on basic 1-forms. Typically, 1-forms are encountered inexplicitly in a multivariate calculus class with Green's theorem:

$$\int_D \left(\frac{\partial f_2}{\partial x} - \frac{\partial f_1}{\partial y} \right) dx dy = \int_{\partial D} f_1(x, y) dx + f_2(x, y) dy \quad (\text{Green's Theorem})$$

Formally, $T_x(\mathbb{R}^n) \rightarrow \mathbb{R}$.

We can add differential forms but rather than multiplication we use the \wedge product.

To create higher order p -forms we use the wedge operator \wedge .

First of all, the wedge product is primarily defined by being *anti-commutative* or *skew-symmetric*. That is, $dx \wedge dy = -dy \wedge dx$. Several results immediately follow from this. When used on two identical dx , we have $dx \wedge dx = -dx \wedge dx$ and so the result must be zero. Additionally, for any permutation σ of $[p]$:

$$dx_1 \wedge \dots \wedge dx_p = \text{sgn}(\sigma) dx_{\sigma(1)} \wedge \dots \wedge dx_{\sigma(p)} \quad (4.38)$$

Definition Given a k -rectangle $\Omega \in \mathbb{R}^n$ with coordinates $x = (x_1, x_2, \dots, x_n)$ A **differential**

p -form β over Ω has the form:

$$\beta = \sum_{j_1 \in [n]} \dots \sum_{j_p \in [n]} b_{(j_1, \dots, j_p)}(x) dx_{j_1} \wedge \dots \wedge dx_{j_p} \quad (4.39)$$

Typically, we will take $j = (j_1, \dots, j_p)$ and express β as, $\sum_j b_j(x) dx_{j_1} \wedge \dots \wedge dx_{j_p}$. We denote the space of all p -forms on Ω by $\Lambda^p(\Omega)$.

dx^i and dx^j are trivial differential 1-forms by taking $f_i = 1$ and $f_j = 1$ respectively. When we take the wedge product of these, we end up with the 2-form $dx^i \wedge dx^j$. It should then seem natural to think of \wedge as an operator which maps a k -form and an ℓ -form to a $(k + \ell)$ -form. For general p and q -forms we define the wedge product.

Definition Let $\alpha = \sum_i a_i(x) dx_{i_1} \wedge \dots \wedge dx_{i_p} \in \Lambda^p(\Omega)$ and $\beta = \sum_j b_j(x) dx_{j_1} \wedge \dots \wedge dx_{j_q} \in \Lambda^q(\Omega)$.

We extend the wedge product $\wedge : \Lambda^p(\Omega) \times \Lambda^q(\Omega) \rightarrow \Lambda^{p+q}(\Omega)$ by:

$$\alpha \wedge \beta = \sum_{i,j} a_i(x) b_j(x) dx_{i_1} \wedge \dots \wedge dx_{i_p} \wedge dx_{j_1} \wedge \dots \wedge dx_{j_q} \quad (4.40)$$

Although we take all possible $\binom{n}{q} \cdot \binom{n}{p}$ pairs of $a_i(x) dx_{i_1} \wedge \dots \wedge dx_{i_p}$ and $b_i(x) dx_{i_1} \wedge \dots \wedge dx_{i_p}$, most of the possible terms will end up being zero. If *any* of the terms in dx_{i_1} appears in dx_{j_1} , then the wedge product will be zero and no term will be contributed. As such, if $q + p > n$, there will be a duplicate in every term and so the entire sum will be zero. When all is said and done, at most we will have $\binom{n}{p+q}$ terms. Rather than the skew-symmetry we had when commuting $dx \wedge dy$, in higher dimensions the sign depends on $p \cdot q$ of the p -form and q -form we are commuting. Specifically,

$$\alpha \wedge \beta = (-1)^{pq} \beta \wedge \alpha \quad (4.41)$$

This can be easily seen by commuting each of $dx_{j_1}, \dots, dx_{j_q}$ terms each past $dx_{i_1} \wedge \dots \wedge dx_{i_p}$. So we are commuting q terms each past p terms, reversing the sign each time for a net $(-1)^{pq}$.

The wedge product is only one part of our algebra of differential forms. We have several other nice identities for its behaviour with addition and multiplication. For the following, we consider f to be a function on \mathbb{R}^n . Additionally we consider the differential forms ω_1 and ω_2 to be k -forms, α to be a p -form and β to be an q -form. Then we have the following:

$$(\omega_1 + \omega_2) \wedge \alpha = \omega_1 \wedge \alpha + \omega_2 \wedge \alpha \quad (4.42)$$

$$(\omega_1 \wedge \alpha) \wedge \beta = \omega_1 \wedge (\alpha \wedge \beta) \quad (4.43)$$

$$(f \cdot \omega_1) \wedge \alpha = f \cdot (\omega_1 \wedge \alpha) = \omega_1 \wedge (f \cdot \alpha) \quad (4.44)$$

These should all be quite obvious from definitions. We should also note the identities which are *not* present. We have defined the sum of ω_1 and ω_2 : two differential forms which are the same dimension but not the sum of α and β : differential forms with different dimension. It is clear how one would add two differential forms of the same dimension as both were defined as sums to begin with. We also do not define the multiplication of \cdot two differential forms but we multiplying a form by a function is simply:

$$(f \cdot \alpha)(x) = \sum_i f(x) \cdot a_i(x) dx_{i_1} \wedge \dots \wedge dx_{i_p} \quad (4.45)$$

Integrating over a k -form is quite simple. First we shall consider integrating a k -form over a k -rectangle in \mathbb{R}^k . Such a k -form is also known as a *top-dimensional form*. As we saw previously, any form of higher degree must be zero. If ω is such a top form then we can always write

$$\omega = f dx_1 \wedge \dots \wedge dx_k \quad (4.46)$$

for some function f . Other presentations of ω exist, but we can always achieve such a presentation by commuting over \wedge to the canonical ordering x_1, \dots, x_n . Once we have the k -form in this presentation, we just remove the wedges and evaluate the integral using the integrand $f dx_1 dx_2 \dots dx_k$.

Definition Let α be a k -form on $\Omega \subset \mathbb{R}^n$ of the form $\alpha = A(x) dx_1 \wedge \dots \wedge dx_n$. If $A \in \mathcal{L}^1(\Omega, dx)$ then we define:

$$\int_{\Omega} \alpha = \int_{\Omega} A(x) dx \quad (4.47)$$

Where the left-hand side is the integral of a k -form and the right-hand side is a Lebesgue integral. For any $\beta \in \Lambda^k(\Omega)$ we extend this definition linearly as the sum of integrals.

4.6 Stokes' Theorem

$$\int_{\partial M} \omega = \int_M d\omega \quad (4.48)$$

4.7 Example

4.8 Manifolds

4.8.1 Singular Cubes

Up until now we have been dealing with the very small set of axis aligned cubes.

Definition Manifold

Definition Given a map $\Phi : X \rightarrow Y$, $x \in X$ pushes forward to $\Phi(x) \in Y$

Forms pull back from Y to X

Benefit of differential forms is how cleanly they handle changes in coordinates.

This is generally done through the use of pull-backs.

A pullback $\varphi^* f$ can be thought of as *functional precomposition*: $(\varphi^* f)(x) = f(\varphi(x))$

Gets its name from pulling F back through ω

Definition $F : X \rightarrow \Omega$ Define the pullback $F^*\beta$

$$F^*\beta = \sum_j b_j(F(x))(F^*dx_{j_1}) \wedge \dots \wedge (F^*dx_{j_k}) \quad (4.49)$$

and

$$F^*dx_j = \sum_\ell \frac{\partial F^j}{\partial x_\ell} dx_\ell \quad (4.50)$$

Which can be reduced by:

$$F^*\beta = \sum_j b_j(F(x))(F^*dx_{j_1}) \wedge \dots \wedge (F^*dx_{j_k}) \quad (4.51)$$

$$= \sum_j b_j(F(x)) \left(\sum_\ell \frac{\partial F^{j_1}}{\partial x_\ell} dx_\ell \right) \wedge \dots \wedge \left(\sum_\ell \frac{\partial F^{j_k}}{\partial x_\ell} dx_\ell \right) \quad (4.52)$$

$$= \dots \quad (4.53)$$

$$= \sum_j b_j(F(x)) \det(J_F) dx_{j_1} \wedge \dots \wedge dx_{j_k} \quad (4.54)$$

Which is significant given the change of variable formula for integration:

$$\int_{\phi(U)} f(v) dv = \int_U f(\phi(u)) |\det \phi'(u)| du \quad (4.55)$$

Theorem 4.8.1 *Let $F : X \rightarrow \Omega$ be an (orientation-preserving diffeomorphism) and α an integrable n -form on Ω then*

$$\int_\Omega \alpha = \int_X F^*\alpha \quad (4.56)$$

More algebra of differential forms

$$F^*(\alpha \wedge \beta) = (F^*\alpha) \wedge (F^*\beta) \quad (4.57)$$

Definition Exterior derivative

...

$$d(\alpha \wedge \beta) = (d\alpha) \wedge \beta + (-1)^j \alpha \wedge (d\beta) \quad (4.58)$$

...

$$F^*(d\beta) = dF^*\beta \quad (4.59)$$

We have already seen standard n -cubes in the previous chapter. Now that we have pullbacks we can define integration on the more general *singular cube*. But first, what is a singular cube.

Definition Singular cube: differentiable map c from standard k -cube to k -dimensional manifold.

It is a common abuse of notation to use c to also refer to the image $c(\llbracket 0, 1 \rrbracket^k) \subseteq M$. The choice of using specifically the standard k -cube is arbitrary. A differentiable map f from $\llbracket a, b \rrbracket$ can always be composed with $g : t \mapsto ta + (1 - t)b$ to get singular cube $c = f \circ g$. Gives us innumerable more shapes to work with. Hemisphere for example is the singular 2-cube given by $(r, \varphi) \rightarrow r \cos(\pi\varphi)x + r \sin(\pi\varphi)y$

Definition Let c be a singular k -cube and ω a k -form on the image of c . Then we define the integral using $c^*(\omega)$ the pull-back of ω along c :

$$\int_c \omega = \int_{c(\llbracket 0, 1 \rrbracket^k)} \omega = \int_{\llbracket 0, 1 \rrbracket^k} c^* \omega \quad (4.60)$$

4.8.2 Manifold Integration

A manifold is nothing more than a collection of local charts diffeomorphic to \mathbb{R}^n

So we use pullbacks to turn a set of rectangles into a manifold while simultaneously applying the manifold's pullback.

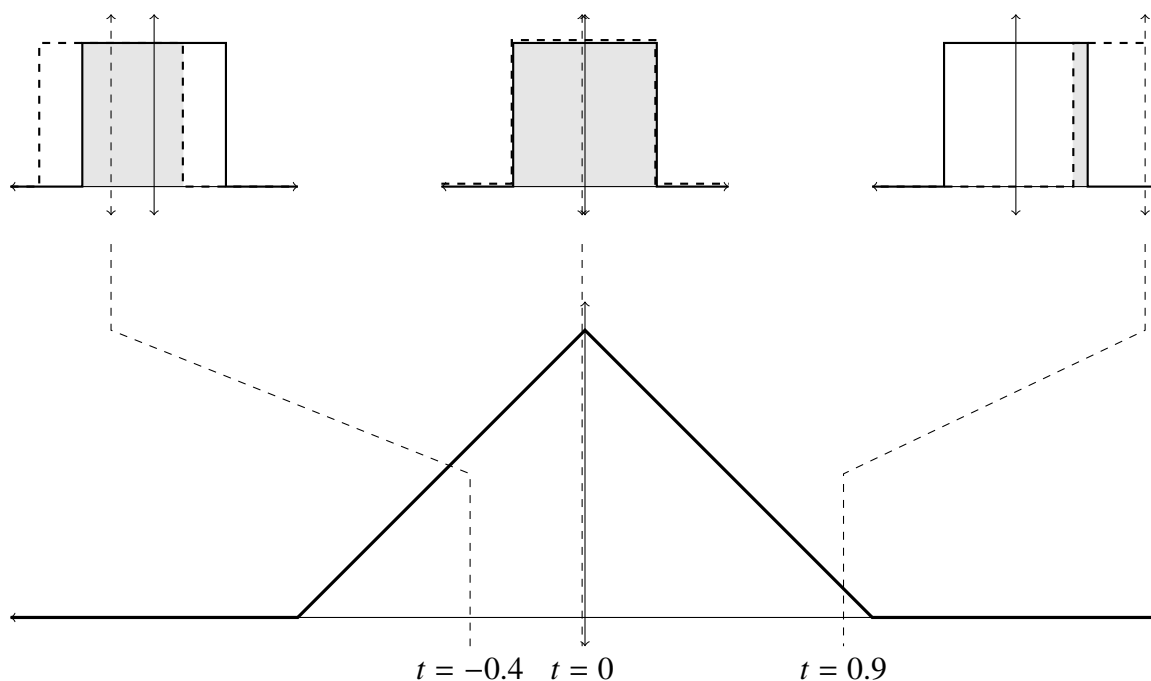
Integration on oriented manifolds is just a careful application of Theorem 5.2.1

Chapter 5

Convolution

Convolution is an operation which takes two functions and produces a third. It takes one of the two input functions, and modifies one by mirroring and translating. The resulting function is then the overlap between one of these functions as a function of the translation. Visually, this can be seen below in Figure 5.1.

Figure 5.1: The convolution of the box signal $f(t) = g(t) = (0^{((-\infty, -0.5))} \oplus 1^{[-0.5, 0.5]} \oplus 0^{((0.5, \infty))})$ with itself.



Formally this, equates to the following definition for convolution over continuous domains:

Definition The **convolution** $*$, of two functions F and G is defined as:

$$(F * G)(t) = \int_{-\infty}^{\infty} F(\tau) G(t - \tau) d\tau \quad (5.1)$$

and in the case of discrete linear convolution, summation would replace integration. In this equation, t represents the translation of G as well as the input for $(F * G)$ while τ is internal to the integral and varies over the real line.

Figure 5.2: 512x512px “Lena”(a) with a 1px (b) and 5px (c) Gaussian blur applied. Gaussian blurring is accomplished by convolving an image with a Gaussian kernel and is commonly used in image processing to reduce noise prior to edge detection.



Convolution has applications in many areas of mathematics and engineering. One very common use in image processing is in blurring. *Gaussian blurring* is the result of a 2-dimensional convolution of an image with the Gaussian distribution function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (5.2)$$

Blurring an image in this way reduces noise and greatly increases the efficacy of subsequent edge detection. In statistics, a (simple) *moving average* can be represented as a convolution by a rectangular pulse while more generally, weighted moving averages can be made by convolving with other functions.

5.1 Convolution of Piecewise Functions

CAS such as Maple and Mathematica are quite adept at solving integrals. Convolution of elementary functions generally poses no problem. When convolving two piecewise continuous functions, many possible intervals arise and the conditionals that arise can quickly overwhelm them unaided.

We are interested in *Symbolic Linear Convolution* (of piecewise continuous functions). The typical approach is to first consider for convolution of “one piece” functions [12, 23]. By “one-piece” functions we mean functions which are restricted to a single interval and zero everywhere else. We will consider two functions, F and G defined as:

$$F(x) = f^{[a_f, b_f)}(x) = \begin{cases} f(x) & a_f \leq x < b_f \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

$$G(x) = g^{[a_g, b_g)}(x) = \begin{cases} g(x) & a_g \leq x < b_g \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

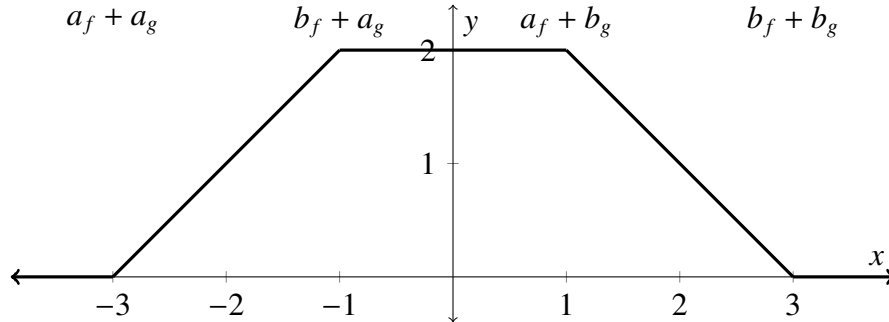
for which we would like to compute the convolution ($F * G$). To reduce the total number of cases generated, it is generally also assumed that $b_f - a_f \leq b_g - a_g$. Assuming that F is non-zero over a shorter interval is not that strong an assumption as convolution is commutative; if it is not the case we can rearrange $F * G$ to $G * F$. To see this, simply apply the substitute $\tau' = t - \tau$ in equation (7.1):

$$\begin{aligned} (F * G)(t) &= \int_{-\infty}^{\infty} F(\tau)G(t - \tau) d\tau \\ &= \int_{\infty}^{-\infty} F(t - \tau')G(\tau') (-1)d\tau' \\ &= \int_{-\infty}^{\infty} G(\tau')F(t - \tau') d\tau' \\ &= (G * F)(t) \end{aligned} \quad (5.5)$$

Thus we can assume that our static function is also the function with the shorter interval. Since F and G are zero outside of their respective intervals, we do not need to integrate over the entire real line. F is our static function, so $[a_f, b_f)$ would be sufficient. For a tight boundary, we have the following:

$$\begin{aligned}
 (F * G)(t) &= \int_{-\infty}^{\infty} F(\tau) G(t - \tau) d\tau \\
 &= \int_{a_f}^{b_f} f(\tau) G(t - \tau) d\tau \\
 &= \begin{cases} \int_{a_f}^{t-a_g} f(\tau) g(t - \tau) d\tau & (a_f + a_g) \leq t < (b_f + a_g) \\ \int_{a_f}^{b_f} f(\tau) g(t - \tau) d\tau & (b_f + a_g) \leq t < (a_f + b_g) \\ \int_{t-b_g}^{b_f} f(\tau) g(t - \tau) d\tau & (a_f + b_g) \leq t < (b_f + b_g) \\ 0 & \text{otherwise} \end{cases} \quad (5.6)
 \end{aligned}$$

Figure 5.3: Convolution of length 1 and 2 rectangular pulses. Given the functions $F = 1^{[-1,1]}$ and $G = 1^{[-2,2]}$, there are three non-zero regions in $(F * G)$.



These regions can be visualized as above in Figure 5.3 where two rectangular pulses are convolved. If both functions had equal length non-zero intervals (i.e. $b_f - a_f = b_g - a_g$), then the central plateau would be empty (as in Figure 5.1). Another formulation presented by Cîrnu [10] and Cavicchi [9] is to use:

$$(F * G)(t) = \begin{cases} \int_{\max(a_f, t-b_g)}^{\min(b_f, t-a_g)} f(\tau) \cdot g(t - \tau) d\tau & (a_f + a_g) \leq t < (b_f + b_g) \\ 0 & \text{otherwise} \end{cases} \quad (5.7)$$

Although this may appear to reduce the number of cases, expanding the min and max will cause just as many cases to return.

To extend this to piecewise continuous function, we simply treat each piecewise function as the sum of “one-piece” functions. Given functions, $F = \sum_i f_i^{P_i}$ and $G = \sum_j g_j^{Q_j}$ where $\{P_i\}, \{Q_j\}$ are each sets of disjoint intervals, and $f_i^{P_i}, g_j^{Q_j}$ are all “one-piece” functions. The convolution of $F * G$ is the sum of pairwise convolution:

$$\begin{aligned}
 \left(\left(\sum_i f_i^{P_i} \right) * \left(\sum_j g_j^{Q_j} \right) \right) (t) &= \int_{-\infty}^{\infty} \left(\sum_i f_i^{P_i} \right) (\tau) \cdot \left(\sum_j g_j^{Q_j} \right) (t - \tau) d\tau \\
 &= \sum_i \sum_j \int_{-\infty}^{\infty} f_i^{P_i}(\tau) \cdot g_j^{Q_j}(t - \tau) d\tau \\
 &= \sum_i \sum_j (f_i^{P_i} * g_j^{Q_j})
 \end{aligned} \tag{5.8}$$

To summarize, the algorithm for convolution of two piecewise functions is as follows:

1. Each function is converted into a sum of disjoint function intervals.
2. Each function interval in one function is convolved with each function interval in the other.
3. The final result is the sum of all function interval convolutions.

This is the typical approach to convolution of piece-wise functions as presented by West and McClellan [23]. When the boundaries between regions is symbolic, then we may not be able to determine which interval is longer. Another approach involving hybrid functions will be presented in Section 5.2. Concerns with intervals where one boundary point is at infinity have also been raised [12]. Techniques to handle this will be investigated in Section 5.3.

5.2 Hybrid Function Convolution

Our exposition for hybrid set convolution will appear very similar to that from the previous section and can be seen as a replacement for step 2 in the algorithm. Again we will be interested in the convolution of “one-piece” functions which we will use to build up piece-wise continuous functions. Assuming two hybrid one-piece functions $f^{[a_f, b_f)}$ and $g^{[a_g, b_g)}$ which are 0 outside of the intervals $[a_f, b_f)$ and $[a_g, b_g)$ respectively. Then the **hybrid convolution** of $f^{[a_f, b_f)}$ and $g^{[a_g, b_g)}$ is:

$$\begin{aligned} (f^{[a_f, b_f)} * g^{[a_g, b_g)})(t) = \mathcal{R}_+ \left(\left(\int_{\llbracket a_f, t-a_g \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket a_f+a_g, b_f+a_g \rrbracket} \right. \\ \oplus \left(\int_{\llbracket a_f, b_f \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket b_f+a_g, a_f+b_g \rrbracket} \\ \left. \oplus \left(\int_{\llbracket t-b_g, b_f \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket a_f+b_g, b_f+b_g \rrbracket} \right)(t) \quad (5.9) \end{aligned}$$

The first thing one should note is the similarity between this expression and (5.6). But, *we do not enforce* $b_f - a_f \leq b_g - a_g$ as we did in Section 5.1. Instead both cases will be handled by our generalized partition structure. If the integral $f(t)g(t-\tau)$ is easier to compute than $g(t)f(t-\tau)$ then the convolution can, of course, still be commuted. This could be due to the nature of functions for f and g or if f and g are a part of a larger sum with identical sub-functions on different regions. Then these integrals could potentially be combined, provided they have the same integrand, resulting in fewer overall integrals to compute. The ordering of f and g is no longer dictated by the relative length of their respective intervals.

When $b_f - a_f \leq b_g - a_g$ then the three oriented intervals will be disjoint and the two equations are identical. Otherwise, if $b_f - a_f > b_g - a_g$, then the interval $\llbracket b_f + a_g, a_f + b_g \rrbracket$ will have a negative orientation. The intervals $\llbracket a_f + a_g, b_f + a_g \rrbracket$, $\llbracket b_f + a_g, a_f + b_g \rrbracket$ and $\llbracket a_f + b_g, b_f + b_g \rrbracket$ still forms a reducible (i.e. everywhere multiplicity one) generalized partition over $[a_f + a_g, b_f + b_g)$. Outside this region the function is zero as expected.

Suppose we are in this second case and we wish to evaluate the convolution at a point t which is in all three intervals. This occurs when $(a_f + b_g) \leq t < (b_f + a_g)$ and we have:

$$\llbracket a_f + a_g, b_f + a_g \rrbracket(t) = 1$$

$$\llbracket b_f + a_g, a_f + b_g \rrbracket(t) = -1$$

$$\llbracket a_f + b_g, b_f + b_g \rrbracket(t) = 1$$

Simplifying the +-reduction we then get:

$$\begin{aligned} (f^{[a_f, b_f]} * g^{[a_g, b_g]})(t) &= \left(\int_{\llbracket a_f, t-a_g \rrbracket} f(\tau) g(t-\tau) d\tau \right) \\ &\quad - \left(\int_{\llbracket a_f, b_f \rrbracket} f(\tau) g(t-\tau) d\tau \right) \\ &\quad + \left(\int_{\llbracket t-b_g, b_f \rrbracket} f(\tau) g(t-\tau) d\tau \right) \end{aligned} \quad (5.10)$$

All three of these integrals have the same integrand so we can use bi-linearity to move the sum to be over the domains of integration. These domains then cancel nicely to leave us with:

$$\begin{aligned} (f^{[a_f, b_f]} * g^{[a_g, b_g]})(t) &= \int_{\llbracket a_f, t-a_g \rrbracket \ominus \llbracket a_f, b_f \rrbracket \oplus \llbracket t-b_g, b_f \rrbracket} f(\tau) g(t-\tau) d\tau \\ &= \int_{\llbracket t-b_g, t-a_g \rrbracket} f(\tau) g(t-\tau) d\tau \end{aligned} \quad (5.11)$$

Let us now look at a concrete example with some actual numbers.

5.2.1 Example: *Hybrid Convolution*

In Figure 5.3 we saw the convolution of $1^{[-1,1]}$ with $1^{[-2,2]}$. We know that convolution is commutative so computing $1^{[-2,2]} * 1^{[-1,1]}$ we already know what to expect. We will label these as 1_f and 1_g to differentiate and to prevent confusion by reminding us that the object we are dealing with is $x \mapsto 1$ rather than the number 1 itself.

$$\begin{aligned}
(1_f^{[-2,2]} * 1_g^{[-1,1]})(t) = \mathcal{R}_+ \left(\left(\int_{\llbracket -2, t-1 \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket -3, 1 \rrbracket} \right. \\
\oplus \left(\int_{\llbracket -2, 1 \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket 1, -1 \rrbracket} \\
\left. \oplus \left(\int_{\llbracket t-1, 2 \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket -1, 3 \rrbracket} \right)(t) \quad (5.12)
\end{aligned}$$

Already this is promising as we can see the set of end-points: $\{-3, -1, 1, 3\}$ agrees with our previous example. Let us consider three points $t_1 \in [-3, -1)$, $t_2 \in [-1, 1)$ and $t_3 \in [1, 3)$. We omit the derivations but encourage the reader to convince themselves that each is correct.

$$(1_f^{[-2,2]} * 1_g^{[-1,1]})(t_1) = \int_{\llbracket -2, t_1 - (-1) \rrbracket} 1_f(\tau) 1_g(t_1 - \tau) d\tau = t_1 + 3$$

First we should note that at no point in the integral do we attempt to evaluate 1_f or 1_g outside of their original domains $[-2, 2]$ and $[-1, 1]$ respectively. Thus it is safe to replace $1_f(\tau) \cdot 1_g(t - \tau)$ with 1 inside the integral. From here, the integral is trivially evaluated and is as expected.

For t_3 , only the third term has non-zero multiplicity and by an identical argument as for t_1 we have:

$$(1_f^{[-2,2]} * 1_g^{[-1,1]})(t_3) = \int_{\llbracket t-1, 2 \rrbracket} 1_f(\tau) 1_g(t_3 - \tau) d\tau = 3 - t_3$$

Finally, t_2 deviates from this pattern slightly as t_2 is in *all three* oriented intervals. By the same derivation as we used in the previous section we can use equation (5.11):

$$(1_f^{[a_f, b_f]} * 1_g^{[a_g, b_g]})(t_2) = \int_{\llbracket t_2-1, t_2 - (-1) \rrbracket} f(\tau) g(t_2 - \tau) d\tau = 2$$

For any other point t which is not in $[-3, 3)$, then all three oriented intervals will have multiplicity zero. Simplifying the $+$ -reduction we have, $\mathcal{R}_+(\emptyset) = e_+ = 0$ and so (5.12) evaluates correctly everywhere.

5.3 Infinite Intervals

The method presented in Section 5.1 behaves correctly assuming that all the end points are finite. But Evans and McClellan [12] raise two issues that arise when we allow for interval end points at infinity. Namely,

1. Interval lengths can no longer be compared to ensure the first interval is shorter than the second.
2. Indeterminant arithmetic may occur in computing new endpoints of the form $+\infty - \infty$

We have already shown that hybrid function convolution is not concerned with the relative length of functions. Clearly, the first point should not be a concern for hybrid convolution. Less clear is that, invalid arithmetic on interval endpoints can be ignored as well! Unlike the 16 different cases used by Evans and McClellan, we can extend hybrid convolution from finite-only to mixed finite and infinite end points with no additional logic.

The first important observation is that indeterminate arithmetic can only occur in *internal* end-points. By this we mean that of the four end-points: $\{a_f + a_g, b_f + a_g, a_f + b_g, b_f + b_g\}$, the points $a_f + a_g$ and $b_f + b_g$ can always safely be evaluated. If b_f were $-\infty$ or a_f were ∞ then the function interval $f^{[a_f, b_f]}$ is actually f^0 and can be ignored (similarly $b_g \neq -\infty$ and $a_g \neq \infty$). Now, recall that the three hybrid set intervals that occurred in equation (5.9) were:

$$\llbracket a_f + a_g, b_f + a_g \rrbracket, \quad \llbracket b_f + a_g, a_f + b_g \rrbracket, \quad \text{and} \quad \llbracket a_f + b_g, b_f + b_g \rrbracket$$

So if indeterminate arithmetic does occur among end-points it would occur at least twice. This will prove useful in allowing us to have these points cancel each other out. For example, suppose $b_f + a_g = \infty + (-\infty) = \perp$ is undefined. Even though, $\llbracket a_f + a_g, b_f + a_g \rrbracket$ and $\llbracket b_f + a_g, a_f + b_g \rrbracket$ may separately be undefined, their sum is not:

$$\llbracket a_f + a_g, b_f + a_g \rrbracket \oplus \llbracket b_f + a_g, a_f + b_g \rrbracket = \llbracket a_f + a_g, a_f + b_g \rrbracket$$

Before we can just add these intervals, each is of course attached to a function so we return to the discussion of compatibility from section 2.3. After substituting the previously assumed $b_f = \infty$ and $a_g = -\infty$, we can see that both the integrands are identical and the domains nearly so as well:

$$\left(\int_{\llbracket a_f, t+\infty \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket -\infty, \perp \rrbracket} \quad \text{and} \quad \left(\int_{\llbracket a_f, \infty \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket \perp, a_f+b_g \rrbracket}$$

For $t \neq -\infty$, the domains of integration match as well. By theorem 2.3.1, these two terms are compatible. Putting this all together:

$$\begin{aligned} (f^{\llbracket a_f, \infty \rrbracket} * g^{\llbracket -\infty, b_g \rrbracket})(t) = \mathcal{R}_+ \left(\left(\int_{\llbracket a_f, \infty \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket -\infty, a_f+b_g \rrbracket} \right. \\ \left. \oplus \left(\int_{\llbracket t-b_g, \infty \rrbracket} f(\tau) g(t-\tau) d\tau \right)^{\llbracket a_f+b_g, \infty \rrbracket} \right)(t) \end{aligned} \quad (5.13)$$

Each end-point can be either finite or infinite; left end-points are either finite or $-\infty$ while right end-points are finite or $+\infty$. So with 4 end-points and 2 possible cases for each, this leads to 16 possible cases to convolve one-piece functions. It can be shown that equation (5.9) without modification is correct in all cases. A full enumeration of this can be found in Appendix A.

5.4 Discrete Convolution

Until now we have assumed F and G are continuous functions. While the continuous is of more historical interest, the discrete case is more widely used in digital signal processing. From a theoretical perspective, very little is different between the continuous and discrete case; it is primarily a swap from integrals \int to sums \sum and evaluating functions at points $f(x)$ to indexing in an array $f[x]$.

Definition The **discrete convolution** of two sequences F and G defined as:

$$(F * G)[t] = \sum_{\tau=-\infty}^{\infty} F[\tau] \cdot G[t - \tau] \quad (5.14)$$

or for sequences with non-negative indexing:

$$(F * G)[t] = \sum_{\tau=0}^{\infty} F[\tau] \cdot G[t - \tau] \quad (5.15)$$

When convolving sequences, limiting the boundaries which need to be summed over is just as important as in the continuous case. In particular, when working with digital inputs represented by arrays, it is very important to not attempt to index outside of the bounds of the arrays. Therefore it is important to get tight bounds on the range of τ . Again, this is a simple swap as well:

$$\begin{aligned} (f^{[a_f, b_f]} * g^{[a_g, b_g]})(t) = \mathcal{R}_+ \left(\left(\sum_{\tau \in \llbracket a_f, t - a_g \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket a_f + a_g, b_f + a_g \rrbracket} \right. \\ \oplus \left(\sum_{\tau \in \llbracket a_f, b_f \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket b_f + a_g, a_f + b_g \rrbracket} \\ \left. \oplus \left(\sum_{\tau \in \llbracket t - b_g, b_f \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket a_f + b_g, b_f + b_g \rrbracket} \right) [t] \quad (5.16) \end{aligned}$$

One must be even more careful with bounds in the discrete case compared to the continuous. Excepting the Dirac function and similar functions, generally including or excluding the endpoints will not affect the evaluation of an integral:

$$\int_{(a,b)} f(x) dx = \int_{[a,b]} f(x) dx = \int_{[a,b)} f(x) dx = \int_{(a,b]} f(x) dx$$

The difference between each interval is measure 0 and so excepting pathological functions, the integrals should be equal. The same does not hold for summation; the openness or “closed-

ness” of an interval matters even in the typical use case.

That being said, the boundary points between intervals are safe to fall in either direction.

For $t = b_f + a_g$, evaluating either the sum

$$\sum_{\tau \in \llbracket a_f, t - a_g \rrbracket} f[\tau] g[t - \tau] \quad \text{or} \quad \sum_{\tau \in \llbracket a_f, b_f \rrbracket} f[\tau] g[t - \tau]$$

equates to the same thing. So we can have the intervals $\llbracket a_f + a_g, b_f + a_g \rrbracket$ and $\llbracket b_f + a_g, a_f + b_g \rrbracket$ or the intervals $\llbracket a_f + a_g, b_f + a_g \rrbracket$ and $\llbracket b_f + a_g, a_f + b_g \rrbracket$; either would be correct. Similarly we can also move the point at $a_f + b_g$ from the third term or the second term.

Here we are using closed intervals for f and g . This is at odds with typical array iteration, like Python’s `range(...)` function which tend to use *closed-open* intervals. Unlike the continuous case, there is no structural difference between an open or closed interval; the choice of using one over the other is usually a matter of whichever gives the nicest indices by avoiding $+1$ ’s or -1 ’s. So we can handle the difference by converting $[a, b) = [a, b - 1]$

$$\begin{aligned} (f^{[a_f, b_f]} * g^{[a_g, b_g]})(t) = \mathcal{R}_+ \left(\left(\sum_{\tau \in \llbracket a_f, t - a_g \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket a_f + a_g, b_f + a_g \rrbracket} \right. \\ \oplus \left(\sum_{\tau \in \llbracket a_f, b_f \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket b_f + a_g, a_f + b_g \rrbracket} \\ \left. \oplus \left(\sum_{\tau \in \llbracket t - b_g, b_f \rrbracket} f[\tau] g[t - \tau] \right)^{\llbracket a_f + b_g, b_f + b_g - 1 \rrbracket} \right) [t] \quad (5.17) \end{aligned}$$

5.5 Implementation

Implementation for continuous symbolic convolution was done in *Maple*. Maple is able to correctly determine many cancellations but requires assistance for a few cases with symbolic and infinite end-points. For the most part, the cases enumerated in Appendix A follow a few patterns:

1. Merging terms with indeterminate end-points. (Cases 6, 7, 9, 11, 13, 14, and 15)
2. Remove any terms over empty intervals. (Cases 1, 2, 3, 5, 7, 10, 11, 13, and 14)
3. Manipulate intervals to find a disjoint partition and combine integrals using linearity of domains (Cases 4, 8, and 12)

To merge indeterminate end-points, we use the local variables `afbg` and `bfag` to represent the sums $a_f + b_g$ and $b_f + a_g$ respectively. If the sums are well defined, then we can apply the substitution, otherwise `afbg` and `bfag` are left symbolic.

```

if (af+bg != undefined) then afbg := af+bg; fi ;
if (bf+ag != undefined) then bfag := bf+ag; fi ;
...

```

In cases where these sums are undefined, these terms will disappear (cancellation shown in Appendix A). However, to assist Maple in finding this cancellation we must leave the sum as a symbolic term. We can avoid any potentially undefined integrals by delaying the evaluation of the functions inside the integral. We use the symbolic `fg(x)` to represent $f(x)*g(t-x)$ to prevent Maple from unwrapping the integrals until we have determined ranges which the integrals will be evaluated much like pseudo-functions from section 2.3:

```

...
out := (int (fg (x) , x=af .. t-ag ))*( OrientedInterval (af+ag , bfag ))( t )
      +(int (fg (x) , x=af .. bf ))*( OrientedInterval (bfag , afbg ))( t )
      +(int (fg (x) , x=t-bg .. bf ))*( OrientedInterval (afbg , bf+bg ))( t );
...

```

To ensure that future steps function smoothly, we remove any $t - \infty$ or $t + \infty$ terms that can arise in the bounds of integration. This is just a simple substitution:

```

...
out := subs (t-infinity=-infinity , out );

```

```

out:=subs(t+infinity=infinity , out);
...

```

If `afbg` or `bfbg` are symbolic ($a_f + b_g$ or $b_f + a_g$ are undefined respectively) then these symbolic terms then disappear when the piecewise `OrientedIntervals` are converted to Heaviside functions: `convert(%, Heaviside, t)`. Converting this back to a piecewise function with `convert(%, piecewise, t)` results in a more readable expression.

However, since the Heaviside function is undefined at 0, this can lead to point discontinuities in cases where $a_f + b_g$ and $b_f + a_g$ are perfectly well defined. To remedy this, before we convert to Heaviside and back, we should first attempt to convert it to a piecewise function. Converting to piecewise will fail if `bfbg` or `afbg` are incomparable (undefined):

```

...
try temp := convert(out, piecewise, t);
catch:
    try temp:=convert(convert(out, Heaviside), piecewise, t);
    catch: temp := out;
end try;
finally out := temp;
end try;
...

```

By this point, Maple has already removed any terms with empty intervals so all that remains is combining integrals by linearity. In most cases this can be handled by `Combine` in the `IntegrationTools` package. `Combine` is unable to combine when end-points are infinite so again we will substitute a symbolic term:

```

...
try temp:=Combine(subs(infinity=infty , out));
catch temp:=out;

```

```
finally out:=subs(infty=infinity,temp); end try;
...
```

The final step is to replace $fg(x)$ with $f(x)*g(t-x)$

```
...
out:=subs(fg(x)=f(x)*g(t-x), out)
```

Hopefully this process strikes the reader as being quite *simple* as most of the functionality to reduce the hybrid function equation is already provided by Maple. This is intentional, not only is it easier to implement but it illustrates the point that hybrid convolution can in fact be a simpler framework with less case-based logic. Admittedly, some strange looking hacks are required to “make it work”, such as the conversion to Heaviside and back to piecewise. With the exception of deciding to leave `afb` and `bfa` symbolic, many of these steps are entirely optional. They result in a simplified, nicer looking expression but the equation can be correctly interpreted without them.

Chapter 6

Conclusions

The primary objective of this thesis was to extend [8] by investigating further applications of hybrid sets and functions. Although this paper was focused on results for integration and Petri net graphs, this is not to take away from the “smaller” results shown along the way. As a first example, we showed that, (notational choice: $\mathbb{Z}^{\mathbb{P}}$, $h(\mathbb{P})$, $\mathcal{H}(\mathbb{P})?$), the hybrid sets over prime numbers is equivalent to \mathbb{Q}_+ .

Hybrid sets come into their own within the context of hybrid functions as we saw with arithmetic on piecewise functions and symbolic matrices. Combined with tricks from linear algebra, the usage of hybrid functions allowed for large decreases in both cases. Hybrid pseudo-functions leave a function associated with an element unevaluated and allow for algebra to be performed on the domains before requesting any functions be evaluated.

Hybrid functions were shown to be a good model for domains of integration. An atlas can succinctly be defined in terms of a set of hybrid relation over a universe of Euclidean rectangles mapping to a common manifold. Principle of Inclusion-Exclusion was used instead of the typical *partitions of unity* to reduce the atlas to its support. Unlike some of the previous examples, any leftover negative terms produced by PIE are completely well-founded and have natural geometric interpretations. Moreover, ∂ , the boundary operator on a k -chain explicitly constructs them. The beauty (and usefulness) of ∂ was then shown with a proof of the gener-

alized Stokes' theorem. Which in turn we used in conjunction with generalized partitions to transform an otherwise difficult to compute integral.

Finally, we showed a novel formulation of Petri net graphs. Instead of considering transitions as a special type of node, we represented transitions along with corresponding arc weights as a single hybrid set. Conditions for liveness and coverability were also discussed. Relaxing the condition of non-negative markings gives way to *lending Petri nets* [3] [4] for which hybrid sets were even better suited. Unfortunately, I just discovered the Bartoletti papers and have not had a chance to read more than the abstract.

Bibliography

- [1] Colin G Bailey, Dean W Gull, and Joseph S Oliveira. Hypergraphic oriented matroid relational dependency flow models of chemical reaction networks. *arXiv preprint arXiv:0902.0847*, 2009.
- [2] Jean-Pierre Banâtre, Pascal Fradet, Yann Radenac, et al. Generalised multisets for chemical programming. *Mathematical Structures in Computer Science*, 16(4):557–580, 2006.
- [3] Massimo Bartoletti, Tiziana Cimoli, and G Michele Pinna. Lending petri nets.
- [4] Massimo Bartoletti, Tiziana Cimoli, and G Michele Pinna. Lending petri nets and contracts. In *Fundamentals of Software Engineering*, pages 66–82. Springer, 2013.
- [5] Wayne D. Blizard. Multiset theory. *Notre Dame Journal of Formal Logic*, 30(1):36–66, 12 1988.
- [6] Wayne D. Blizard. Negative membership. *Notre Dame Journal of Formal Logic*, 31(3):346–368, 06 1990.
- [7] George Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*, volume 2. Walton and Maberly, 1854.
- [8] Jacques Carette, Alan P. Sexton, Volker Sorge, and Stephen M. Watt. Symbolic domain decomposition. In *Intelligent Computer Mathematics*, pages 172–188. Springer, 2010.
- [9] Thomas J Cavicchi. Simplified method for analytical evaluation of convolution integrals. *Education, IEEE Transactions on*, 45(2):142–144, 2002.

- [10] Mircea I Cîrnu. Calculation of convolution products of piecewise defined functions and some applications. *Journal of Information Systems and Operations Management*, 6:41–52, 2012.
- [11] E Damiani, O D’Antona, and D Loeb. Getting results with negative thinking. In *Proceedings of the Conference of Formal Series and Algebraic Combinatorics, Bordeaux*, 1991.
- [12] Brian L Evans and James H McClellan. Algorithms for symbolic linear convolution. In *Asilomar Conference on Signals, Systems, and Computers*, pages 948–948. Computer Society Press, 1994.
- [13] Theodore Hailperin. *Boole’s logic and probability: a critical exposition from the standpoint of contemporary algebra, logic and probability theory*. Elsevier, 1986.
- [14] Donald E Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms, The*. Addison-Wesley Professional, 2014.
- [15] Monica D Lam, Edward E Rothberg, and Michael E Wolf. The cache performance and optimizations of blocked algorithms. *ACM SIGOPS Operating Systems Review*, 25(Special Issue):63–74, 1991.
- [16] Daniel Loeb. Sets with a negative number of elements. *Advances in Mathematics*, 91(1):64–74, 1992.
- [17] Wolfgang Reisig. *Petri nets: an introduction*. Springer-Verlag New York, Inc., 1985.
- [18] Karsten Schmidt. Parameterized reachability trees for algebraic petri nets. In *Application and Theory of Petri Nets 1995*, pages 392–411. Springer, 1995.
- [19] Alan P Sexton, Volker Sorge, and Stephen M Watt. Abstract matrix arithmetic. In *Symbolic and Numeric Algorithms for Scientific Computing, 2008. SYNASC’08. 10th International Symposium on*, pages 61–68. IEEE, 2008.

- [20] D Singh, A Ibrahim, T Yohanna, and J Singh. An overview of the applications of multisets. *Novi Sad Journal of Mathematics*, 37(3):73–92, 2007.
- [21] D Singh, AM Ibrahim, T Yohanna, and JN Singh. A systematization of fundamentals of multisets. *Lecturas Matematicas*, 29:33–48, 2008.
- [22] T Tao. Differential forms and integration. Technical report, Tech. Rep., Department of Mathematics, UCLA, 2007.
- [23] Kevin A West and JH McClellan. Symbolic convolution. *Education, IEEE Transactions on*, 36(4):386–393, 1993.
- [24] Hassler Whitney. Characteristic functions and the algebra of logic. *Annals of Mathematics*, pages 405–414, 1933.
- [25] NJ Wildberger. A new look at multisets. *University of New South Wales, Sydney*, 2003.

Appendix A

Convolution with Infinite End-Points

Table A.1: All possible cases of finite and infinite end-points. Finite end-points are denoted with F . Infinite left end-points are denoted $-\infty$ and infinite right end-points are denoted ∞ .

	a_f	b_f	a_g	b_g			a_f	b_f	a_g	b_g
Case 0:	F	F	F	F		Case 8:	$-\infty$	F	F	F
Case 1:	F	F	F	∞		Case 9:	$-\infty$	F	F	∞
Case 2:	F	F	$-\infty$	F		Case 10:	$-\infty$	F	$-\infty$	F
Case 3:	F	F	$-\infty$	∞		Case 11:	$-\infty$	F	$-\infty$	∞
Case 4:	F	∞	F	F		Case 12:	$-\infty$	∞	F	F
Case 5:	F	∞	F	∞		Case 13:	$-\infty$	∞	F	∞
Case 6:	F	∞	$-\infty$	F		Case 14:	$-\infty$	∞	$-\infty$	F
Case 7:	F	∞	$-\infty$	∞		Case 15:	$-\infty$	∞	$-\infty$	∞

When convolving one-piece functions with infinite end-points, there are 4 end-points which can each be either finite or infinite. As such there are 2^4 possible combinations of end-point types shown in the table above. Throughout all calculations in this section, the integrands will not change, only the domains. So we define the function C as a sort of restricted convolution:

$$C_{\llbracket x, y \rrbracket}(t) = \int_{\llbracket x, y \rrbracket} f(\tau) g(t - \tau) d\tau \quad (\text{A.1})$$

Which can be used to condense equation (5.9), the definition for hybrid convolution, into:

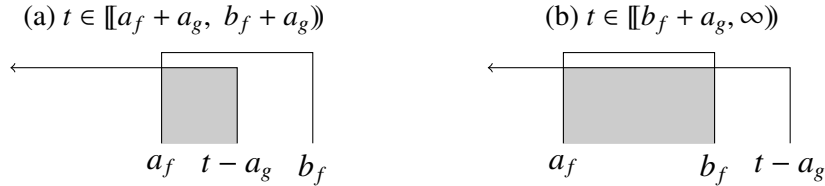
$$(f^{[a_f, b_f]} * g^{[a_g, b_g]}) = \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, b_f+a_g \rrbracket} \oplus C_{\llbracket a_f, b_f \rrbracket}^{\llbracket b_f+a_g, a_f+b_g \rrbracket} \oplus C_{\llbracket t-b_g, b_f \rrbracket}^{\llbracket a_f+b_g, b_f+b_g \rrbracket} \right)$$

Case 0 has all finite points and was already shown to be correct in Section 5.2 but is listed for completeness. The exposition will not be repeated here.

Case 1 has one infinite point, $b_g = \infty$ and results in an empty interval for the third term. Since it is impossible for t to be in $\llbracket \infty, \infty \rrbracket$, we can safely remove this term altogether.

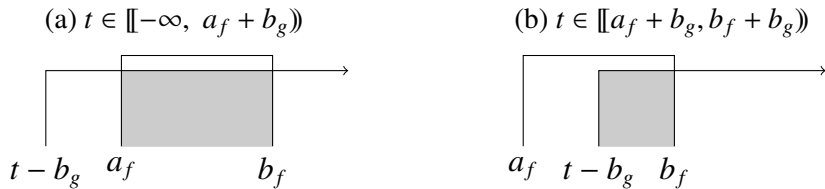
$$\begin{aligned} (f^{[a_f, b_f]} * g^{[a_g, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, b_f+a_g \rrbracket} \oplus C_{\llbracket a_f, b_f \rrbracket}^{\llbracket b_f+a_g, \infty \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket \infty, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, b_f+a_g \rrbracket} \oplus C_{\llbracket a_f, b_f \rrbracket}^{\llbracket b_f+a_g, \infty \rrbracket} \right) \end{aligned}$$

Throughout this section we will use sets of diagrams like the one below to verify that our expressions are sensible. The region where the intervals, $[a_f, b_f)$ and $[t - b_g, t - a_g)$ overlap, (shaded in gray) is where the convolution will be non-zero. The bounds of this intersection may depend on t ; each case that results in a non-empty intersection will be shown separately. To verify an expression, each diagram should correspond to a term in the convolution and the bounds of the shaded region should correspond to the domain on that term's integral.



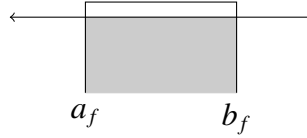
Case 2 also has only one infinite point, $a_g = -\infty$ and also yields an empty interval:

$$\begin{aligned} (f^{[a_f, b_f]} * g^{[-\infty, b_g]}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket a_f, b_f \rrbracket}^{\llbracket -\infty, a_f+b_g \rrbracket} \oplus C_{\llbracket t-b_g, b_f \rrbracket}^{\llbracket a_f+b_g, b_f+b_g \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket a_f, b_f \rrbracket}^{\llbracket -\infty, a_f+b_g \rrbracket} \oplus C_{\llbracket t-b_g, b_f \rrbracket}^{\llbracket a_f+b_g, b_f+b_g \rrbracket} \right) \end{aligned}$$



Case 3 is a combination of both case 1 and 2, resulting in only a single term for the entire real line since both the first and third terms are over empty intervals.

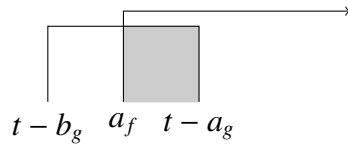
$$\begin{aligned}
 (f^{[a_f, b_f]} * g^{[-\infty, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket a_f, b_f \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket \infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket a_f, b_f \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) = \int_{\llbracket a_f, b_f \rrbracket} f(\tau) g(t - \tau) d\tau \\
 &\quad \text{(a) } t \in \llbracket -\infty, \infty \rrbracket
 \end{aligned}$$



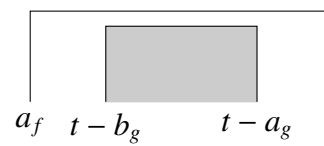
Case 4 (i.e. $b_f = \infty$) is a bit more involved:

$$\begin{aligned}
 (f^{[a_f, \infty)} * g^{[-\infty, b_g]}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, \infty \rrbracket} \oplus C_{\llbracket a_f, \infty \rrbracket}^{\llbracket \infty, a_f+b_g \rrbracket} \oplus C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket a_f+b_g, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, a_f+b_g \rrbracket \oplus \llbracket a_f+b_g, \infty \rrbracket} \ominus C_{\llbracket a_f, \infty \rrbracket}^{\llbracket a_f+b_g, \infty \rrbracket} \oplus C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket a_f+b_g, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, a_f+b_g \rrbracket} \oplus C_{\llbracket a_f, t-a_g \rrbracket \ominus \llbracket a_f, \infty \rrbracket \oplus \llbracket t-b_g, \infty \rrbracket}^{\llbracket a_f+b_g, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, a_f+b_g \rrbracket} \oplus C_{\llbracket t-b_g, t-a_g \rrbracket}^{\llbracket a_f+b_g, \infty \rrbracket} \right)
 \end{aligned}$$

(a) $t \in \llbracket a_f + a_g, a_f + b_g \rrbracket$



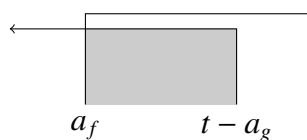
(b) $t \in \llbracket a_f + b_g, \infty \rrbracket$



Case 5 $b_f = \infty, b_g = \infty$

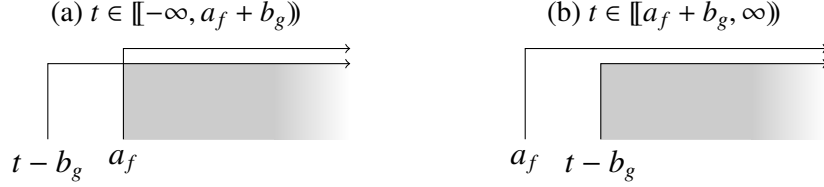
$$\begin{aligned}
 (f^{[a_f, \infty)} * g^{[a_g, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, \infty \rrbracket} \oplus C_{\llbracket a_f, \infty \rrbracket}^{\llbracket \infty, \infty \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket a_f, t-a_g \rrbracket}^{\llbracket a_f+a_g, \infty \rrbracket} \right)
 \end{aligned}$$

(a) $t \in \llbracket a_f + a_g, \infty \rrbracket$



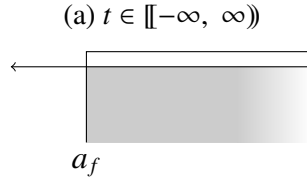
Case 6: $b_f = \infty, a_g = -\infty$

$$\begin{aligned} (f^{[a_f, \infty)} * g^{[-\infty, b_g)}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, \perp \rrbracket} \oplus C_{\llbracket a_f, \infty \rrbracket}^{\llbracket \perp, a_f + b_g \rrbracket} \oplus C_{\llbracket t - b_g, \infty \rrbracket}^{\llbracket a_f + b_g, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, a_f + b_g \rrbracket} \oplus C_{\llbracket t - b_g, \infty \rrbracket}^{\llbracket a_f + b_g, \infty \rrbracket} \right) \end{aligned}$$



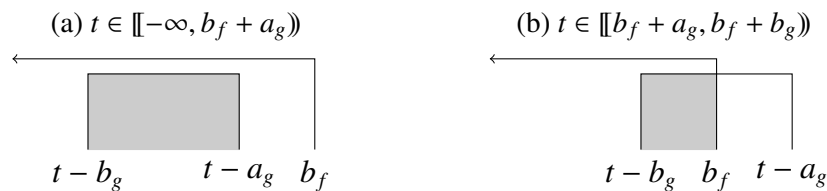
Case 7: $b_f = \infty, a_g = -\infty, b_g = \infty$

$$\begin{aligned} (f^{[a_f, \infty)} * g^{[-\infty, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, \perp \rrbracket} \oplus C_{\llbracket a_f, \infty \rrbracket}^{\llbracket \perp, \infty \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \infty, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket a_f, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \end{aligned}$$



Case 8: $a_f = -\infty$

$$\begin{aligned} (f^{[-\infty, b_f)} * g^{[a_g, b_g)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t - a_g \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket b_f + a_g, -\infty \rrbracket} \oplus C_{\llbracket t - b_g, b_f \rrbracket}^{\llbracket -\infty, b_f + b_g \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t - a_g \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} \oplus C_{\llbracket t - b_g, b_f \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} \oplus \llbracket b_f + a_g, b_f + b_g \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t - a_g \rrbracket}^{\llbracket -\infty, b_f \rrbracket} \oplus \llbracket t - b_g, b_f \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} C_{\llbracket t - b_g, b_f \rrbracket}^{\llbracket b_f + a_g, b_f + b_g \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket t - b_g, t - a_g \rrbracket}^{\llbracket -\infty, b_f + a_g \rrbracket} C_{\llbracket t - b_g, b_f \rrbracket}^{\llbracket b_f + a_g, b_f + b_g \rrbracket} \right) \end{aligned}$$



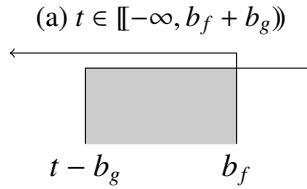
Case 9: $a_f = -\infty, b_g = \infty$

$$\begin{aligned} (f^{[-\infty, b_f]} * g^{[a_g, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, b_f+a_g \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket b_f+a_g, \perp \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket \perp, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, b_f+a_g \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket b_f+a_g, \infty \rrbracket} \right) \end{aligned}$$



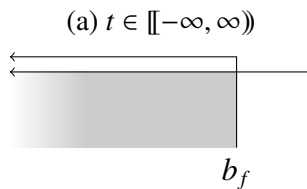
Case 10: $a_f = -\infty, a_g = -\infty$

$$\begin{aligned} (f^{[-\infty, b_f]} * g^{[-\infty, b_g]}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket t-b_g, b_f \rrbracket}^{\llbracket -\infty, b_f+b_g \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket t-b_g, b_f \rrbracket}^{\llbracket -\infty, b_f+b_g \rrbracket} \right) \end{aligned}$$

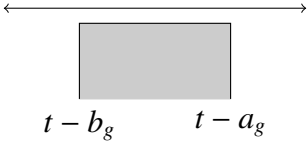


Case 11: $a_f = -\infty, a_g = -\infty, b_g = \infty$

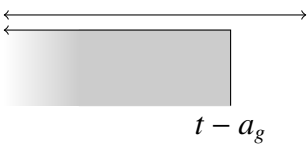
$$\begin{aligned} (f^{[-\infty, b_f]} * g^{[-\infty, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket -\infty, \perp \rrbracket} \oplus C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket \perp, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket -\infty, b_f \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \end{aligned}$$



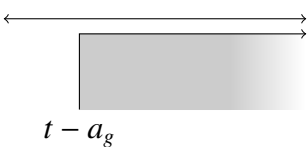
Case 12: $a_f = -\infty, b_f = \infty$

$$\begin{aligned}
 (f^{[-\infty, \infty)} * g^{[a_g, b_g)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \infty, -\infty \rrbracket} \oplus C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket \oplus \llbracket -\infty, \infty \rrbracket \oplus \llbracket t-b_g, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket t-b_g, t-a_g \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &\quad \text{(a) } t \in \llbracket -\infty, \infty \rrbracket
 \end{aligned}$$


Case 13: $a_f = -\infty, b_f = \infty, b_g = \infty$

$$\begin{aligned}
 (f^{[-\infty, \infty)} * g^{[a_g, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \infty, \perp \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \perp, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket -\infty, t-a_g \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &\quad \text{(a) } t \in \llbracket -\infty, \infty \rrbracket
 \end{aligned}$$


Case 14: $a_f = -\infty, b_f = \infty, a_g = -\infty$

$$\begin{aligned}
 (f^{[-\infty, \infty)} * g^{[-\infty, b_g)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, \perp \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \perp, -\infty \rrbracket} \oplus C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, -\infty \rrbracket} \oplus C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &= \mathcal{R}_+ \left(C_{\llbracket t-b_g, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\
 &\quad \text{(a) } t \in \llbracket -\infty, \infty \rrbracket
 \end{aligned}$$


Case 15: $a_f = -\infty, b_f = \infty, a_g = -\infty, b_g = \infty$ has all infinite points and is not really what one would think of as a one-piece function at all. The definition of convolution already holds for such functions. That being said:

$$\begin{aligned} (f^{[-\infty, \infty)} * g^{[-\infty, \infty)}) &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, \perp_1 \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \perp_1, \perp_2 \rrbracket} \oplus C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket \perp_2, \infty \rrbracket} \right) \\ &= \mathcal{R}_+ \left(C_{\llbracket -\infty, \infty \rrbracket}^{\llbracket -\infty, \infty \rrbracket} \right) \\ &= \int_{\llbracket -\infty, \infty \rrbracket} f(\tau) g(t - \tau) d\tau \end{aligned}$$

```

Convolve := proc(f, af, bf, g, ag, bg)
  local afbg, bfag, out, temp;
  if(af+bg != undefined) then afbg := af+bg; fi;
  if(bf+ag != undefined) then bfag := bf+ag; fi;
  out := (int(fg(x), x=af..t-ag))*(OrientedInterval(af+ag, bfag))(t)
    +(int(fg(x), x=af..bf))*(OrientedInterval(bfag, afbg))(t)
    +(int(fg(x), x=t-bg..bf))*(OrientedInterval(afbg, bf+bg))(t);
  out := subs(t-infinity=-infinity, out);
  out := subs(t+infinity=infinity, out);
  try temp := convert(out, piecewise, t);
  catch:
    try temp := convert(convert(out, Heaviside), piecewise, t);
    catch: temp := out; end try;
  finally out := temp; end try;
  try temp := Combine(subs(infinity=infty, out));
  catch temp := out;
  finally out := subs(infty=infinity, temp); end try;
  out := subs(fg(x)=f(x)*g(t-x), out);
end proc;

```



```
> Convolve(sin , 0, Pi , t->exp(-t) , 0, 1);
```

$$\begin{cases} 0 & t \sim < 0 \\ \int_{t \sim}^0 \sin(x) e^{-t \sim + x} dx & t \sim < 1 \\ \int_{-1+t \sim}^{t \sim} \sin(x) e^{-t \sim + x} dx & t \sim < \pi \\ \int_{-1+t \sim}^{t \sim} \sin(x) e^{-t \sim + x} dx & t \sim < \pi + 1 \\ 0 & \pi + 1 \leq t \sim \end{cases}$$

Case 1:

```
> assume(t::real); additionally(af<bf); additionally(ag<bg);
> Convolve(f , af , bf , g , ag , infinity );
```

$$\begin{cases} 0 & t \sim < af \sim + ag \sim \\ \int_{af \sim}^{t \sim - ag \sim} f(x) g(t \sim - x) dx & t \sim < bf \sim + ag \sim \\ \int_{af \sim}^{bf \sim} f(x) g(t \sim - x) dx & bf \sim + ag \sim \leq t \sim \end{cases}$$

Curriculum Vitae

Name: Mike Ghesquiere

**Post-Secondary
Education and
Degrees:** University of Western Ontario
London, ON
2008-2012 B.Sc.

University of Western Ontario
London, ON
2013-2015 M.Sc.

**Related Work
Experience:** Teaching Assistant
The University of Western Ontario
2013 - 2015