The algebraic specification of spatial data types with applications to constructive volume geometry.

Johnson, Kenneth Harold Anthony

The Algebraic Specification of Spatial Data Types with Applications to Constructive Volume Geometry

Kenneth Harold Anthony Johnson

A thesis submitted to the University of Wales in fulfilment of the requirements for the Degree of Doctor of Philosophy.

Department of Computer Science
Swansea University

Spring 2007

To my parents

Abstract

Spatial objects are modelled as total functions, mapping a topological space of points to a topological algebra of data attributes. High-level operations on these spatial objects form algebras of spatial objects, which model spatial data types. This thesis presents a comprehensive account of the theory of spatial data types.

The motivation behind the general theory is Constructive Volume Geometry (CVG). CVG is an algebraic framework for the specification, representation and manipulation of graphics objects in 3D. By using scalar fields as the basic building blocks, CVG gives an abstract representation of spatial objects, with the goal of unifying the many representations of objects used in 3D computer graphics today.

The general theory developed in this thesis unifies discrete and continuous spatial data, and the many examples where such data is used - from computer graphics to hardware design.

Such a theory is built from the algebraic and topological properties of spatial data types. We examine algebraic laws, approximation methods, and finiteness and computability for general spatial data types. We show how to apply the general theory to modelling (i) hardware and (ii) CVG.

We pose the question *"Which spatial objects can be represented in the algebraic framework developed for spatial data types?"*. To answer such a question, we analyse the expressive power of our algebraic framework. Applying our results to the CVG framework yields a new result: We show any CVG spatial object can be approximated by way of CVG terms, to arbitrary accuracy.

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed [illegible] (candidate)

Date March 13th 2007

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed [illegible] (candidate)

Date March 13th 2007

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed [illegible] (candidate)

Date March 13th 2007

Acknowledgments

I have relied on many people for help and support while preparing this thesis, and I would like to take this opportunity to thank them.

My mother and father deserve the most thanks. Without their love and support I simply could not have completed this thesis. Extra special thanks goes to Susanne Linz and her family for making me feel welcome in their home. I owe a great deal to my extended family both in Canada and here in the UK: Beverly Haley and Andrea Cassidy, The Cabin Boys Ed and Tom Coules and Dean O'Grady, Harry and Colleen Johnson, Dennis and Marian O'Grady. Thank you to Ian and Brian Johnson for always making me feel welcome in Birmingham, and giving me a "home away from home". To all my friends in Canada and in the UK: Nasim Akhtar, Ian Armstrong, Steven Ballard, Scott Brown, Dan Byers, Marc Cezer, Curtis Cram, Zoë Fisher, Zelin Li, Liam Mccarthy, Katie Milosek, Tara and the Neville family, Pamela Schmidt, Christine Stender, Christal Stewart, Derek and the Visutski family. Thanks to The Catholic Society at UWS and the Catholic Community at St. David's Church, Swansea for their support throughout the years. Thanks to the staff at the accommodation office at the university, particularly Nik Harding and Debbie Saunders. Thank you to the Department of Computer Science at the University of Wales, Swansea for providing an excellent environment in which to study, research and be a student. I would especially like to thank Alfie Abdul-Rahman, Ulrich Berger, Jens Blanck, Min Chen, Lyndsey Clarke, Jill Edwards, Will Harwood, Markus Michelbrink, and Jeffery Zucker for some insightful conversations throughout the years here in Swansea. Thank you to the Engineering and Physical Sciences Research Council (EPSRC) for funding my research via the grant *Constructive Volume Geometry - a Fundamental Modelling Methodology For Volume Graphics* Reference: GR/R25286/01 held by Professors Min Chen and John Tucker. Thank you to Min Chen who provided the graphics for the Constructive Volume Geometry chapter.

Thank you to my supervisor John V Tucker for continued support and friendship throughout the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What Are Spatial Objects?

Spatial objects represent data distributed over a space. They are found almost everywhere in Computer Science. For example, in

- *discrete space* we have computer memory, where the space is the set of all memory addresses, each storing exactly one element of data, and in
- *continuous space* we have computer graphics, where objects are represented in 3D space consisting of data from medical scanning devices.

Clearly one can see the diversity of data in space and think of examples of spatial objects defined over different types of space comprised of data from a variety of sources. In each application the user must choose the space and data from which the objects will be made. In order to compute with these objects the user must, in addition, choose some operations to manipulate them. The choices of data and operations on spatial objects are infinite and depend primarily on the application area. For instance, in the context of computer graphics, one may consider operations of scaling, rotation or translation on 3D graphical objects in the RGB graphics model.

The aim of this thesis is to create a general theory of spatial objects and their operations. We will do this using algebra and topology.

1.2 Constructive Volume Geometry

Computer graphics is our primary motivation for the general theory of spatial objects, and we focus particularly on *volume graphics.* This area of computer graphics is concerned with the representation of three dimensional data and methods in which such 3D data is rendered on a two dimensional medium, for example, a computer monitor. An excellent overview of the technology and methods currently used in volume graphics is [14] while [16] gives some practical methods of rendering volumetric data.

In volume graphics, data may be represented in many ways. We may have discrete volume data gathered from a medical scanning device or our data may be the values of a continuous function. Thus when defining graphical objects in this application area, we can expect their data to be from a variety of sources. Now we would like to perform operations such as rotations or translations, on these objects regardless of the type of data that forms our objects.

To achieve a uniform way of manipulating objects, we use Constructive Volume Geometry (CVG) and this serves as the motivation for our work and the main application for our general theory. CVG was developed by Chen and Tucker in [19] as an algebraic framework for the specification, representation and manipulation of *volume data types.* It gives an *abstract* representation of spatial objects and of the operations performed on these objects. Thus, users of CVG perform operations on objects using high-level CVG operations. This allows objects containing data from various sources, discrete or continuous data, to be handled in a uniform manner. Such representations of data are important, as they have a considerable effect on the computational properties of CVG [61], [63].

One key idea is that objects are represented in the CVG framework using *finite* syntax, called CVG *terms*, representing complex spatial objects built from operations on simpler objects. This finiteness is essential in the computation of spatial objects in CVG. One could envision a program which, given a complex syntactic term representing some sequence of operations on CVG objects, works out the resulting object and *renders* it so that it can be viewed by the user as a graphical *scene* on the display. We could also consider a program to transform CVG terms into some normal form which would reduce the computation time needed for rendering.

We pose the new question:

Can CVG terms describe all possible CVG spatial objects?

The choice of operations available to the user of CVG are limitless, but generally are based on the needs of a particular application. We would like some operations and some *basic* CVG spatial objects that we could use to generate all the CVG spatial objects we are interested in. In this way, we would have a sequence of CVG spatial objects and the operations performed on them to create more and more complex objects. This leads to the formulation of *The Completeness Problem*, for CVG and for spatial objects in general.

1.3 The Completeness Problem

At the heart of the theory of abstract data types lies data and operations on data. A natural question to consider is the following.

Are there some operations that are, in a sense, necessary or adequate to construct complicated objects from simple ones?

In other words, we would like some operations and some *basic* spatial objects that we can use to generate all the spatial objects we are interested in. In this way, we would have a composition of spatial objects and the operations performed on them to create more and more complex objects.

The primary obstacle to this approach is that this set of generated sequences of operations on spatial objects, which we call *terms* (Section 3.3), is countable. The set of spatial objects is commonly uncountable, therefore we cannot represent every spatial object as a term. We can however produce a set to generate all spatial objects possibly by *approximation.*

Theories of approximation require topological notions, and for this reason we introduce topologies on the space and data. Our interest therefore is narrowed to the set of *continuous* spatial objects, where such objects and their approximation behaves and is well understood.

We formulate the *Completeness Problem* as follows: For a set C of *continuous* spatial objects can we find

(i) a finite collection Σ of operations on spatial objects,

(ii) a subset $B \subset C$ of spatial objects (called *basic spatial objects*),

such that when we apply the operations of Σ to the generators in B repeatedly the set $\langle B\rangle_{\Sigma_{SW}}$ generated is *adequate* to approximates all spatial objects. Specifically, factors we also need to consider

(iii) a topology on C to formulate the method of approximation.

In this case, the requirement of adequacy based on approximation is simply

$$\langle B\rangle_{\Sigma_{SW}} \text{ is dense in } C.$$

1.4 Aim of Thesis

The primary aim of this thesis is to study a data type of spatial objects and provide a high-level framework of objects and operations on objects that will be useful in developing a theory general enough to handle spatial objects in a uniform manner, without having to consider the data it is made from. To meet this aim, we:

1. Give an account of the algebraic properties of spatial objects, including their laws for specification,
2. Unify different examples from graphics, hardware, etc.,
3. Unify discrete and continuous spatial data types,
4. Give an account of approximation properties of spatial objects.

1.5 Thesis Outline

In Chapter 2 we introduce Constructive Volume Geometry where we give an informal specification of the data (scalars) used to define our CVG objects. In CVG, data is represented by a set S of scalars and we show how operations on total maps $\phi : E^3 \rightarrow S$ called *scalar fields* can be defined. We develop some algebraic properties and show equational laws on our data lift to equational laws on scalar fields. We show how operations on objects are defined using operations on the scalar fields.

Chen and Tucker have asserted in [19] that Constructive Volume Geometry is a general algebraic framework which encapsulates all the representations of computer graphics in use today. Examples of CVG classes of spatial objects are given: Opacity-Only and the Four-Colour Channel Model, chosen for their widespread use in practical applications of computer graphics. We show informally how *Constructive Solid Geometry* as described in the early papers of Requicha [49], [50] is embedded in CVG. The actual mathematical proof is deferred to Chapter 4 Section 4.5. CVG acts as a running application of the general theory throughout the thesis.

Our aim is to give a general answer to the completeness problem, which will require theory from several mathematical fields. We give some preliminary definitions in Chapter 3, and set the notation used throughout the thesis. The reader is expected to have some background in both the theory of abstract data types and topology and a knowledge of computer graphics is also helpful, but not essential.

Universal algebra will play an important role in our work as spatial objects are modelled by algebras of total functions. The basic definitions and notation for many-sorted algebras, terms and term evaluation are given. Some algebraic constructions (products, inverse limits), and structures (commutative rings, vector spaces) are used in the thesis, thus basic definitions and results are included in this chapter. The need for topological results is apparent by our remarks on the completeness problem: we are interested in approximating *continuous* spatial objects, as such objects are well behaved. In a language more suited to topology, the term "spatial object" is often replaced with "function". That is, we are dealing with the class of all continuous functions. A stable theory exists for this important class of functions. It is treated in advanced topological textbooks such as [26], [27], [36] to name a few. In the preliminaries we give some of the basic terminology and definitions from topology and metric spaces. It is also useful to define here various types of orderings such as preorders, partial orders and directed sets. Such notions are needed for the approximation of spatial objects.

Chapter 4 is a study of the algebraic properties of spatial objects. Starting with a set of data and operations on the data, we give a general method of defining operations on spatial objects that perform the operation on the data at each point in space. This method is called *pointwise lifting* or *pointwise extension* and is also used in [41]. Such a construction has many nice properties. Equational Validity is preserved (Theorem 4.3.1) which we use to show equational axiomatic specifications of the data also hold when pointwise lifted to spatial objects (Corollary 4.3.1).

In the general theory of spatial data types we study data distributed throughout an arbitrary topological space. It is useful to make a distinction between *discrete space* and *continuous space*. Their application areas are different and we would like to study both in some detail. Thus we have treated each type of space in a separate chapter.

Chapter 5 gives an account of how one may build a topological space over the set of spatial objects $F(X, A)$ where X is a countable set, for example the integers, and the data set A is some topological algebra. We look at finite approximations which restricts a spatial object to a finite subset X_{fin} of X and consider how $\phi : X_{fin} \rightarrow A$ can approximate $\phi : X \rightarrow A$. Sets of finite restrictions of spatial objects form the basic open sets of the topology on $F(X, A)$. We give many examples of data distributed over discrete space. A larger case study is done with the *Spatially Extended Register Machine* or SERM. The SERM is a computational model which we use to run *programs* over data in an algebra A. We analyse both the operational semantics and the input/output of the model and give some results such as the Local Computation Theorem 5.4.1. We show how the algebraic results of Chapter 4 can give us an easy way of defining machines which compute over the spatial objects in $F(X, A)$. We interpret this construction as a method of modelling parallel computing. That is, a SERM executing a program on the data at each point in the space X simultaneously.

In Chapter 6, the data is a topological algebra A distributed over an arbitrary topological space X. We study the class $C(X, A)$ of all continuous total functions mapping points in the space X to the data in A. The topology we gave for spatial objects in discrete space is actually a special case of the *compact-open topology*, and we give its general construction in detail as well as investigate some useful properties. By generalising finiteness to *compactness*, we use compact sets as a way of approximating a spatial object locally, and these sets of approximating spatial objects form the basic open sets of the compact-open topology on $C(X, A)$. Of particular importance in our work is when the data A is the real numbers $\mathbb{R}$. Section 6.6 studies this case and we show how the metric space $\mathbb{R}$ defines the uniform convergence norm. For such a norm to be meaningful, approximation is limited to spaces that are compact. This does limit our applications, as the space used by CVG is $\mathbb{R}^3$. We resolve this issue starting in Section 6.8 by defining a topology on the inverse limit over the family of compact subsets of the topological space X. This topology is shown to be equivalent to the compact-open topology over $C(X, A)$ in Theorem 6.8.1.

Chapter 7 gives a detailed account of the expressiveness and completeness problem, and the role of approximation. We want to find *operations* such that when applied to some basic spatial objects, we can generate *all* spatial objects we are interested in, possibly by approximation. The result which we use is the Stone-Weierstrass Theorem 7.3.2. This is a well known topological result which gives conditions that the operations and basic spatial objects must satisfy to produce a dense subset of $C(X, \mathbb{R})$. We begin in Section 7.3.6 by proving the theorem for the case of spatial objects $C(K, \mathbb{R})$ mapping from a compact space K to the real numbers. The theorem is extended to handle spatial objects whose data is the unit interval $[0, 1]$ and the n-tuple $\mathbb{R}^n$. Such a result is useful when applied to CVG spatial objects and we look at such an application in Section 7.5 for the Four-Colour Channel model. Using the inverse limit construction we prove Theorem 7.7.1 which approximates spatial objects over a non-compact space ($\mathbb{R}^3$ for example). Thus give a spatial object in $C(X, \mathbb{R})$, this theorem finds a suitable spatial object that approximates it locally on a basic open set of the compact-open topology.

Lastly, Chapter 8 reviews the results we have given in this work. Suggestions for further research are give and we discuss some possible extensions of the work. We discuss how a theory of computation may be developed in the future for programming with spatial data types.

Chapter 2

Volume Graphics and Constructive Volume Geometry

2.1 Introduction

Computer graphics have been used extensively in practical computing for well over 30 years. The users of computer graphics have developed many methods to model graphics objects. These representations, or schemes, are varied and depend on the needs of the user and the type of objects that are being represented. Each scheme has a collection of operations that are used to manipulate the objects represented. For instance, 3D volumes have originally been stored as a simple three dimensional arrays. Such a representation is low level. As applications such as medical scanning devices develop, a more general and abstract understanding develops.

Constructive Volume Geometry (CVG) is an algebraic framework that works towards a unification of such representations. Using the theory of abstract data types [43] and [55], CVG was proposed by Chen and Tucker and first published in [19] for the high-level specification, manipulation and rendering of spatial objects.

The goal of CVG is to develop a flexible framework that is independent of any particular concrete representation. This framework is based on *scalar fields*: mappings of the form

$$\phi : E^3 \to S$$

from Euclidean space E^3 to scalar data in a set S, such as the unit interval $[0, 1]$. The user may choose which data is used in the objects that are being modelled, as well as the operations that will be performed on these objects.

2.2 Scalars and Scalar Fields

Graphics objects in any scheme rely on some type of data which describes the object. This data could be of any type, largely dependent on the particular application area. For example the Constructive Solid Geometry scheme [49], [50] represents solid objects and surfaces in three dimensional space but offers no description of the "internal" contents of an object. The data used here is limited to only the set $\{0, 1\}$ but is enough to describe the shape of the object.

In this section we describe the mathematical idea of scalar fields, which will give us a method of describing the data at all points of an object, particularly of its internal composition.

2.2.1 Operations on Scalars

We begin by mathematically describing the data itself, which are commonly referred to as *scalars.*

The scalars S will be the real numbers or a subset of the real numbers, such as the unit interval $[0, 1]$. Usually, there will be some operations on the scalars whose input are $n \geq 1$ values of S and $m \geq 0$ real numbers and output is a value in S. A typical scalar operation f is of the form

$$f : S^n \times \mathbb{R}^m \to S.$$

Let $S = [0, 1]$ denote a set of scalars. There can be many such operations and the user can pick whichever ones are preferred, based on the needs of the application. For instance, we can define some useful operations such as normal arithmetic operations one would expect on numbers, as well as min and max:

$$\begin{aligned}
\max(s_1, s_2) &= \mathit{maximum}_{\mathbb{R}}(s_1, s_2) \\
\min(s_1, s_2) &= \mathit{minimum}_{\mathbb{R}}(s_1, s_2) \\
s_1 + s_2 &= \mathit{minimum}_{\mathbb{R}}(1, s_1 +_{\mathbb{R}} s_2) \\
s_1 - s_2 &= \mathit{maximum}_{\mathbb{R}}(0, s_1 -_{\mathbb{R}} s_2) \\
s_1 \cdot s_2 &= s_1 \cdot_{\mathbb{R}} s_2.
\end{aligned}$$

for scalars $s_1, s_2 \in [0, 1]$.

2.2.2 Equational Laws on Scalars

These operations satisfy some basic laws, each of which may be an *equation*, and so we call them *equational laws.* We list them as follows:

Table 2.1: Equational Laws for Operations on Scalars

Commutativity	$\max(s_1, s_2)$	$=$	$\max(s_2, s_1)$
	$\min(s_1, s_2)$	$=$	$\min(s_2, s_1)$
	$s_1 + s_2$	$=$	$s_2 + s_1$
	$s_1 \cdot s_2$	$=$	$s_2 \cdot s_1$
Associative Laws	$\max(s_1, \max(s_2, s_3))$	$=$	$\max(\max(s_1, s_2), s_3)$
	$\min(s_1, \min(s_2, s_3))$	$=$	$\min(\min(s_1, s_2), s_3)$
	$s_1 + (s_2 + s_3)$	$=$	$(s_1 + s_2) + s_3$
	$s_1 \cdot (s_2 \cdot s_3)$	$=$	$(s_1 \cdot s_2) \cdot s_3$
Distributive Laws	$s_1 + \max(s_2, s_3)$	$=$	$\max(s_1 + s_2, s_1 + s_3)$
	$s_1 + \min(s_2, s_3)$	$=$	$\min(s_1 + s_2, s_1 + s_3)$
	$r \cdot \max(s_1, s_2)$	$=$	$\max(r \cdot s_1, r \cdot s_2)$
	$r \cdot \min(s_1, s_2)$	$=$	$\min(r \cdot s_1, r \cdot s_2)$
	$r \cdot (s_1 - s_2)$	$=$	$(r \cdot s_1) - (r \cdot s_2)$
Idempotent Laws	$\max(s, s)$	$=$	s
	$\min(s, s)$	$=$	s
Identity Laws	$\max(s, 0)$	$=$	s
	$\min(s, 1)$	$=$	s
	$s + 0$	$=$	s
	$s - 0$	$=$	s
	$1 \cdot s$	$=$	s
Dominance Laws	$\max(s, 1)$	$=$	1
	$\min(s, 0)$	$=$	0
	$s + 1$	$=$	1
	$0 - s$	$=$	0
	$0 \cdot s$	$=$	0
Absorption Laws	$\max(s_1, \min(s_1, s_2))$	$=$	s_1
	$\max(s_1, s_1 - s_2)$	$=$	s_1
	$\min(s_1, \max(s_1, s_2))$	$=$	s_1

$$\min(s_1, s_1 + s_2) = s_1$$

Other Useful Laws	$s - 1$	$=$	0
	$s_1 - max(s_2, s_3)$	$=$	$min(s_1 - s_2, s_1 - s_3)$
	$s_1 - min(s_2, s_3)$	$=$	$max(s_1 - s_2, s_1 - s_3)$
	$s_1 - (s_2 + s_3)$	$=$	$((s_1 - s_2) - s_3)$

2.2.3 Additional Operations

Let S and T be sets of scalars, and $s_1, s_2 \in S$, $t_1, t_2 \in T$. We present some useful operations on scalars. They shall be used later in the chapter when we consider particular CVG models (cf. Section 2.4.3).

$$combine((s_1, t_1), (s_2, t_2)) = \begin{cases} \dfrac{t_1 \cdot s_1 + t_2 \cdot s_2}{s_1 + s_2}, & \text{if } s_1 \neq 0 \text{ or } s_2 \neq 0; \\ \dfrac{t_1 + t_2}{2}, & \text{if } s_1 = s_2 = 0. \end{cases}$$

$$select((s_1, t_1), (s_2, t_2)) = \begin{cases} t_1 & \text{if } s_1 \geq s_2; \\ t_2 & \text{if } s_1 < s_2. \end{cases}$$

$$cap((s_1, t_1), (s_2, t_2)) = \begin{cases} t_1 & \text{if } s_1 > s_2; \\ 0 & \text{if } s_1 \leq s_2. \end{cases}$$

2.2.4 Scalar Fields

Scalar fields are the foundation of the CVG framework. They are used to define CVG spatial objects in E^3 space and provide the basis for a mathematical description of the *data attributes* that CVG spatial objects are made of.

Scalar fields are defined by total functions of the form

$$\phi : E^3 \rightarrow S$$

mapping all points in 3D space to a set of scalars. Thus set of all scalar fields is defined as

$$F(E^3, S) = \{\phi \mid \phi : E^3 \rightarrow S \text{ is total}\}.$$

At the heart of CVG are operations which take scalar fields as input and returns a scalar field as output. Such operations will be derived from the operations on the set of scalars.

Using the scalar operations listed in Section 2.2.1, we define some operations over scalar fields in $F(E^3, S)$:

$$\begin{aligned} Max(\phi_1, \phi_2)(x) &= \max(\phi_1(x), \phi_2(x)) \\ Min(\phi_1, \phi_2)(x) &= \min(\phi_1(x), \phi_2(x)) \\ (\phi_1 + \phi_2)(x) &= \phi_1(x) + \phi_2(x) \\ (\phi_1 - \phi_2)(x) &= \phi_1(x) - \phi_2(x) \\ (\phi_1 \times \phi_2)(x) &= \phi_1(x) \cdot \phi_2(x) \end{aligned}$$

for scalar fields $\phi_1, \phi_2 \in F(E^3, S)$ and a point $x \in E^3$.

One may interpret these scalar field operations as performing a scalar operation at each point in E^3 simultaneously. This general method of defining scalar field operations is commonly known as *pointwise lifting*, discussed in Chapter 4.

Additionally, we pointwise lift the scalar operations defined in Section 2.2.3 to give the following operations on scalar fields:

$$\begin{aligned} Combine((\phi_1, \tau_1), (\phi_2, \tau_2))(x) &= combine((\phi_1(x), \tau_1(x)), (\phi_2(x), \tau_2(x))) \\ Select((\phi_1, \tau_1), (\phi_2, \tau_2))(x) &= select((\phi_1(x), \tau_1(x)), (\phi_2(x), \tau_2(x))) \\ Cap((\phi_1, \tau_1), (\phi_2, \tau_2))(x) &= cap((\phi_1(x), \tau_1(x)), (\phi_2(x), \tau_2(x))) \end{aligned}$$

for scalar fields ϕ_1, ϕ_2 in $F(E^3, S)$ and τ_1, τ_2 in $F(E^3, T)$ and a point $x \in E^3$.

2.2.5 Equational Laws on Scalar Fields

Equational laws that are true on sets of scalar values are also true on scalar fields.

We consider an example:

Example 2.2.1 The scalar equational law

$$s_1 - \max(s_2, s_3) = \min(s_1 - s_2, s_1 - s_3)$$

for scalars in S is automatically true on the set $F(E^3, S)$ of scalar fields:

$$\phi_1 - Max(\phi_2, \phi_3) = Min(\phi_1 - \phi_2, \phi_1 - \phi_3).$$

The operations satisfy the scalar equational laws given in Table 2.1. In fact we can make the statement:

Equational laws on scalars extend to equational laws on scalar fields.

This statement will be proved in a more general setting in Theorem 4.3.1 and applied to CVG in Section 4.5 of Chapter 4.

Thus for scalar fields $\phi_1, \phi_2, \phi_3, \phi \in F(E^3, S)$ and $\rho, \rho_1, \rho_2 \in F(E^3, \mathbb{R})$ we have the following table of equational laws:

Table 2.2: Equational Laws for Operations on Scalar Fields

Commutativity	$Min(\phi_1, \phi_2)$	$=$	$Min(\phi_2, \phi_1)$
	$\phi_1 + \phi_2$	$=$	$\phi_2 + \phi_1$
	$\phi_1 \cdot \phi_2$	$=$	$\phi_2 \cdot \phi_1$
Associative Laws	$Max(\phi_1, Max(\phi_2, \phi_3))$	$=$	$Max(Max(\phi_1, \phi_2), \phi_3)$
	$Min(\phi_1, Min(\phi_2, \phi_3))$	$=$	$Min(Min(\phi_1, \phi_2), \phi_3)$
	$(\phi_1 + (\phi_2 + \phi_3))$	$=$	$((\phi_1 + \phi_2) + \phi_3)$
Distributive Laws	$\phi_1 + Max(\phi_2, \phi_3)$	$=$	$Max(\phi_1 + \phi_2, \phi_1 + \phi_3)$
	$\phi_1 + Min(\phi_2, \phi_3)$	$=$	$Min(\phi_1 + \phi_2, \phi_1 + \phi_3)$
	$\rho \cdot Max(\phi_1, \phi_2)$	$=$	$Max(\rho \cdot \phi_1, \rho \cdot \phi_2)$
	$\rho \cdot Min(\phi_1, \phi_2)$	$=$	$Min(\rho \cdot \phi_1, \rho \cdot \phi_2)$
	$\rho \cdot (\phi_1 - \phi_2)$	$=$	$(\rho \cdot \phi_1) - (\rho \cdot \phi_2)$
Idempotent Laws	$Max(\phi, \phi)$	$=$	$Min(\phi, \phi)$
		$=$	ϕ
Identity Laws	$Max(\phi, 0)$	$=$	ϕ
	$Min(\phi, 1)$	$=$	ϕ
	$\phi + 0$	$=$	ϕ
	$\phi - 0$	$=$	ϕ
	$1 \cdot \phi$	$=$	ϕ
Dominance Laws	$Max(\phi, 1)$	$=$	1
	$Min(\phi, 0)$	$=$	0
	$\phi + 1$	$=$	1
	$0 - \phi$	$=$	0
	$0 \cdot \phi$	$=$	0
Absorption Laws	$Max(\phi_1, Min(\phi_1, \phi_2))$	$=$	ϕ_1
	$Max(\phi_1, \phi_1 - \phi_2)$	$=$	ϕ_1
	$Min(\phi_1, Max(\phi_1, \phi_2))$	$=$	ϕ_1
	$Min(\phi_1, \phi_1 + \phi_2)$	$=$	ϕ_1

Other Useful Laws	$\phi - 1$	$=$	0
	$\phi_1 - Max(\phi_2, \phi_3)$	$=$	$Min(\phi_1 - \phi_2, \phi_1 - \phi_3)$
	$\phi_1 - Min(\phi_2, \phi_3)$	$=$	$Max(\phi_1 - \phi_2, \phi_1 - \phi_3)$
	$\phi_1 - (\phi_2 + \phi_3)$	$=$	$(\phi_1 - \phi_2) - \phi_3$
	$(\rho_1 + \rho_2) \cdot \phi$	$=$	$(\rho_1 \cdot \phi) + (\rho_2 \cdot \phi)$

2.3 Spatial Objects and Their Operations

We now have some mathematical models of the data and operations on the data for which our CVG spatial objects are comprised of. With these definitions and examples in place we are now in a position to define a CVG spatial object from [19].

Definition 2.3.1 *A CVG spatial object is a tuple $\mathbf{o} = (O, A_1, \ldots, A_k)$ of scalar fields defined in E^3, where $O : E^3 \rightarrow [0,1]$ is an opacity field, with*

$$Attr_1 : E^3 \rightarrow A_1, \ldots, Attr_k : E^3 \rightarrow A_k$$

as k other possible attributes fields.

From the definition, we note that all CVG spatial objects have an opacity field. This field specifies the visibility of the object at each point in E^3. Such data is used by rendering algorithms to display the visible geometry of an object on a user's screen.

Given an opacity field $O : E^3 \rightarrow [0,1]$, a point x in E^3 is said to be opaque if $O(x) = 1$. If $O(x) = 0$ then the point is transparent. When $O(x)$ is any other value then the point is semi-transparent. Consider the situation when the geometry we are describing is that of an amorphous phenomenon [31], for example, fog. Not every point in such an object is opaque, or completely transparent. We shall denote a completely transparent spatial object by the symbol □ and a opaque spatial object by the symbol ■.

Spatial objects will also contain k attributes

$$A_1, \ldots, A_k,$$

which are used to define properties such as colour, reflection or any other data useful to the rendering process.

We may also have non-visual data in the attributes such as magnetic fields or information about the presence of some type of chemicals at a certain point in space. Different applications need different sets of attributes for the spatial objects they specify and CVG allows the user to define attributes that describe whatever data is needed by the particular application they are working in.

CVG uses the theory of abstract data types [43] for naming scalar fields which provides a way to work with all the spatial objects that have the same attributes.

A spatial object *declaration* Σ is defined in [19](p.284) as a collection of names for spaces, attributes and scalar fields. Often times it is useful to display this information:

Object Declaration	Σ
Space	sp
Attributes	$Opacity, A_1, \ldots, A_k$
Fields	$O : sp \rightarrow Opacity$ $Attr_1 : sp \rightarrow A_1, \ldots, Attr_k : sp \rightarrow A_k$

We denote the family of all spatial objects which share the same space, attributes and fields specified in the declaration Σ as

$$O(\Sigma).$$[1]

We define a *CVG class* to be an algebraic structure A over the set of spatial objects $O(\Sigma)$ of the declaration Σ and operations $\Phi_1, \ldots, \Phi_m$ which operate on these spatial objects. We display a CVG class as

CVG Class	A
Spatial Objects	$O(\Sigma)$
Operations	$\Phi_1 : O(\Sigma)^{n_1} \rightarrow O(\Sigma), \ldots, \Phi_m : O(\Sigma)^{n_m} \rightarrow O(\Sigma)$

Given some objects in $O(\Sigma)$ we can apply some of the operations to *construct* more and more complex scenes of spatial objects in our CVG class. This sequence of operation applications on spatial objects are represented by *CVG terms.* We define the set of CVG terms inductively by the following rules

(i) the spatial objects $\mathbf{o}_1, \ldots$ in the set $O(\Sigma)$ are CVG terms,

[1]In fact, $O(\Sigma)$ is the class $Alg(\Sigma)$. We introduce algebraic concepts in the next chapter.

(ii) $\Phi_1(t_1, \ldots, t_{n_1}), \ldots, \Phi_m(t_1, \ldots, t_{n_m})$ are CVG terms for operations $\Phi_1, \ldots, \Phi_m$ performed on the CVG terms t_{i_j} where $1 \leq i \leq n$ and $1 \leq j \leq m$, and

(iii) nothing else is a CVG term.

CVG terms represent compositions of operations on spatial objects and can be expressed using trees, where non-terminal nodes represent CVG operators and terminal nodes represent a spatial object. In this sense, we have a finite representation of a rather complicated spatial object. For example, the term

$$t = \mathbf{o}_1 \boxed{\cup} (\mathbf{o}_4 \boxed{-} \mathbf{o}_5) \boxed{\cup} \mathbf{o}_2 \boxed{\cup} \mathbf{o}_3 \boxed{\cup} ((\mathbf{o}_9 \boxed{\cup} \mathbf{o}_{10} \boxed{-} \mathbf{o}_{11}) \boxed{-} \mathbf{o}_6 \boxed{-} \mathbf{o}_7 \boxed{-} \mathbf{o}_8)$$

represents the CVG scene displayed in Figure 2.1, where $\mathbf{o}_1, \ldots, \mathbf{o}_{11}$ are spatial objects.

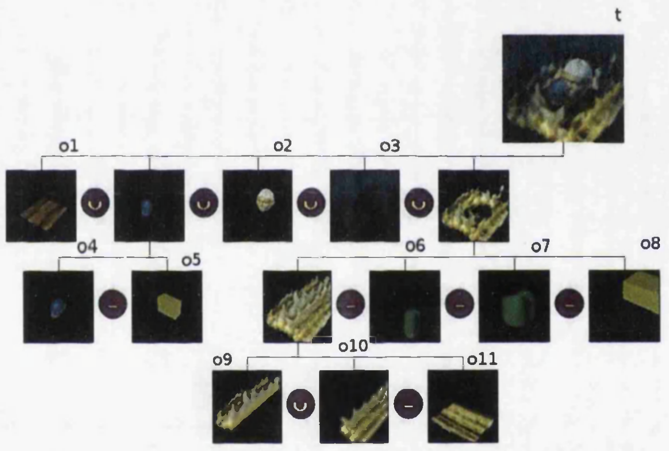

Figure 2.1: Tree Representation of CVG Term t

2.3.1 Operations on Spatial Objects

Operations on the set of spatial objects in $O(\Sigma)$ can be defined from the operations on scalar fields, which in turn are defined from the scalars themselves. Let $G_0, G_1, \ldots, G_k$ be scalar field operations of the form

$$G_i : (O \times A_1 \times \ldots \times A_k) \times (O \times A_1 \times \ldots \times A_k) \rightarrow A_i$$

which takes two $(k+1)$-tuples of attributes and returns an attribute.

General binary operations $\Phi : O(\Sigma) \times O(\Sigma) \to O(\Sigma)$ for spatial objects $\mathbf{o}_1 = (O_1, A_{1,1}, \ldots, A_{1,k})$ and $\mathbf{o}_2 = (O_2, A_{2,1}, \ldots, A_{2,k})$ are defined as

$$\Phi(\mathbf{o}_1, \mathbf{o}_2) = (G_o, G_1, \ldots, G_k)$$

where $G_0, G_1, \ldots, G_k$ are abbreviations of

$$\begin{array}{c} G_o(O_1, A_{1,1}, \ldots, A_{1,k}, O_2, A_{2,1}, \ldots, A_{2,k}) \\ G_1(O_1, A_{1,1}, \ldots, A_{1,k}, O_2, A_{2,1}, \ldots, A_{2,k}) \\ \vdots \\ G_k(O_1, A_{1,1}, \ldots, A_{1,k}, O_2, A_{2,1}, \ldots, A_{2,k}) \end{array}$$

respectively.

2.3.2 Data Representation in CVG

The data in the Constructive Volume Geometry framework is organized into two parts: the *Object level*, and the *Field level.*

The object level is represented by a CVG tree which corresponds to a CVG term, where the root of such a tree represents the final composed spatial object, an example of which is given in Figure 2.2. The field level specifies the scalar fields. These may be defined mathematically as a function or specify only a single constant value or be a very complex volumetric data set taken from a medical scanning device. Clearly this data is varied, and one of the features of

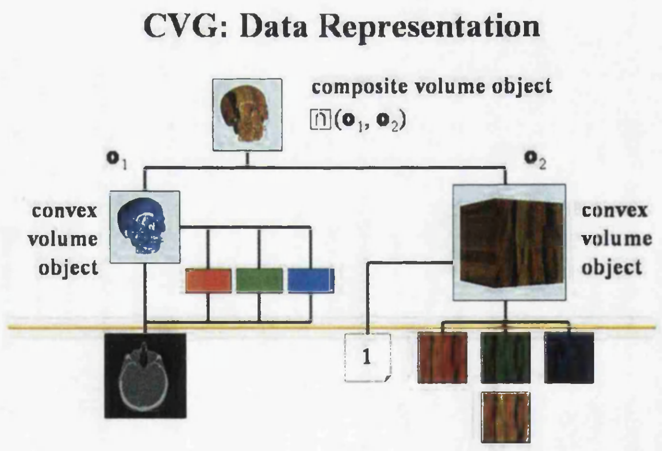

Figure 2.2: Composition of objects with data from a variety of sources

CVG is that spatial objects are separated from the data they are comprised of.

In this next section we will give examples of such operations when defining some particularly useful examples of CVG models: the Opacity-Only model and the closely related Boolean-Only model, and the 4-Colour Channel model.

2.4 Classes of CVG Spatial Objects

We have developed a general framework in which the user can select some data from which to form spatial objects, and define operations on these objects.[19] In this section, we highlight this usage in practical terms by showing how some commonly used schemes can be specified by the CVG framework.

2.4.1 Opacity-Only Model

The simplest scheme we can consider is that where spatial objects have only one attribute: opacity. We declare these types of spatial objects as follows:

Object Declaration	Σ_{1cc}
Space	E^3
Attributes	$[0,1]$
Fields	$O : E^3 \rightarrow [0,1]$

Thus the Opacity-Only declaration of spatial objects is

$$O(\Sigma_{1cc}) = \{\mathbf{o} \mid \mathbf{o} : E^3 \rightarrow [0,1]\}.$$

We define some useful operations on the spatial objects in $O(\Sigma_{1cc})$. We collect these operations and their definitions in the following CVG class:

CVG Class	A_{1cc}
Spatial Objects	$O(\Sigma_{1cc})$
Operations	$\boxed{\cup} : O(\Sigma_{1cc}) \times O(\Sigma_{1cc}) \rightarrow O(\Sigma_{1cc})$ $\boxed{\cap} : O(\Sigma_{1cc}) \times O(\Sigma_{1cc}) \rightarrow O(\Sigma_{1cc})$ $\boxed{-} : O(\Sigma_{1cc}) \times O(\Sigma_{1cc}) \rightarrow O(\Sigma_{1cc})$
Definitions	$\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2) = Max(O_1, O_2)$ $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2) = Min(O_1, O_2)$ $\boxed{-}(\mathbf{o}_1, \mathbf{o}_2) = O_1 - O_2$

We note that the spatial objects in the opacity-only class of CVG contain one attribute: opacity. Thus spatial objects are simply $\mathbf{o}_1 = O_1$, where $O_1 : E^3 \rightarrow [0,1]$ is a scalar field.

We can extend the equational laws of scalar fields that the spatial objects are composed of to the actual spatial objects in $O(\Sigma_{1cc})$. Continuing with Example 2.2.1 we have

Example 2.4.1 The equation

$$\phi_1 - Max(\phi_2, \phi_3) = Min(\phi_1 - \phi_2, \phi_1 - \phi_3)$$

on the scalar fields $\phi_1, \phi_2, \phi_3 : E^3 \rightarrow [0,1]$ lifts to the equational law

$$\boxed{-}(\mathbf{o}_1, \boxed{\cup}(\mathbf{o}_2, \mathbf{o}_3)) = \boxed{\cap}(\boxed{-}(\mathbf{o}_1, \mathbf{o}_2), \boxed{-}(\mathbf{o}_1, \mathbf{o}_3))$$

on spatial objects $\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3$ in $O(\Sigma_{1cc})$.

We list additional equational laws for Opacity-Only spatial objects in Table 2.3:

Table 2.3: Scalar Operation Laws

Equational Laws for A_{1cc}			
Commutativity	$\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2)$	$=$	$\boxed{\cap}(\mathbf{o}_2, \mathbf{o}_1)$
Associative Laws	$\boxed{\cup}(\mathbf{o}_1, \boxed{\cup}(\mathbf{o}_2, \mathbf{o}_3))$	$=$	$\boxed{\cup}(\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2), \mathbf{o}_3)$

$$\begin{array}{lrcl}
 & \boxed{\cap}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_2,\mathbf{o}_3)) & = & \boxed{\cap}(\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2),\mathbf{o}_3) \\
\text{Idempotent Laws} & \boxed{\cup}(\mathbf{o},\mathbf{o}) & = & \mathbf{o} \\
 & \boxed{\cap}(\mathbf{o},\mathbf{o}) & = & \mathbf{o} \\
\text{Identity Laws} & \boxed{\cup}(\mathbf{o},\square) & = & \mathbf{o} \\
 & \boxed{\cap}(\mathbf{o},\blacksquare) & = & \mathbf{o} \\
 & \boxed{-}(\mathbf{o},\square) & = & \mathbf{o} \\
\text{Dominance Laws} & \boxed{\cup}(\mathbf{o},\blacksquare) & = & \blacksquare \\
 & \boxed{\cap}(\mathbf{o},\square) & = & \square \\
 & \boxed{-}(\square,\mathbf{o}) & = & \square \\
\text{Absorption Laws} & \boxed{\cup}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2)) & = & \mathbf{o}_1 \\
 & \boxed{\cap}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2)) & = & \mathbf{o}_1 \\
 & \boxed{\cup}(\mathbf{o}_1,\boxed{-}(\mathbf{o}_1,\mathbf{o}_2)) & = & \mathbf{o}_1 \\
\text{Other Useful Laws} & \boxed{-}(\mathbf{o},\blacksquare) & = & \square \\
 & \boxed{-}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_2,\mathbf{o}_3)) & = & \boxed{\cap}(\boxed{-}(\mathbf{o}_1,\mathbf{o}_2),\boxed{-}(\mathbf{o}_1,\mathbf{o}_3)) \\
 & \boxed{-}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_2,\mathbf{o}_3)) & = & \boxed{\cup}(\boxed{-}(\mathbf{o}_1,\mathbf{o}_2),\boxed{-}(\mathbf{o}_1,\mathbf{o}_3))
\end{array}$$

2.4.2 The Boolean-Only Opacity Model and CSG

Closely related to the Opacity Model, is Boolean-Only Opacity model. Instead of using the entire interval $[0, 1]$, we only consider the subset $\mathbb{B} = \{0, 1\} \subset [0, 1]$ consisting of two values. This allows us to specify CVG objects that map points in E^3 to the set $\mathbb{B}$, thereby specifying the *surface* of a solid object. The Boolean-Only Opacity type of spatial object are declared as

Object Declaration	Σ_B
Space	E^3
Attributes	$\mathbb{B}$
Fields	$O : E^3 \rightarrow \mathbb{B}$

We denote the set of Boolean-Only CVG objects as

$$O(\Sigma_B) = \{\mathbf{o} \mid \mathbf{o} : E^3 \rightarrow \mathbb{B}\}$$

An object $\mathbf{o}$ in $O(\Sigma_B)$ that occupies space in a subset $A \subset E^3$ is defined by

$$\mathbf{o}(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

where a value $\mathbf{o}(x) = 1$ means that the point x is in the object defined by the subset A, and a value $\mathbf{o}(x) = 0$ means the point is outside the object.

We define the operations $\boxed{\cup}$, $\boxed{\cap}$ and $\boxed{-}$ for Boolean-Only Opacity CVG objects as follows:

CVG Class	$A_{\mathbb{B}}$
Spatial Objects	$O(\Sigma_B)$
Operations	$\boxed{\cup} : O(\Sigma_B) \times O(\Sigma_B) \to O(\Sigma_B)$ $\boxed{\cap} : O(\Sigma_B) \times O(\Sigma_B) \to O(\Sigma_B)$ $\boxed{-} : O(\Sigma_B) \times O(\Sigma_B) \to O(\Sigma_B)$
Definitions	$\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2) = Max(O_1, O_2)$ $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2) = Min(O_1, O_2)$ $\boxed{-}(\mathbf{o}_1, \mathbf{o}_2) = O_1 - O_2$

Constructive Solid Geometry

Constructive Solid Geometry (CSG) has been used to model solid objects in computer graphics. Examples of such modeling abound in civil and mechanical engineering.

Roughly speaking, in CSG an object is specified by a subset of E^3. A user has several primitive objects that can be combined to construct more complicated objects by way of operations of union, intersection and difference. The reader interested in the mathematical details which underly the theory of Constructive Solid Geometry are directed to some early papers such as [49] and [50].

To specify objects in CSG we use the set of all subsets of E^3, which we denote by $\mathscr{P}(E^3)$. The CSG operations $\cup$, $\cap$ and $-$ then have the usual set-theoretic definitions. We list the operations in the class

CVG Class	A_{CSG}
Spatial Objects	$\mathscr{P}(E^3)$
Operations	$\cup : \mathscr{P}(E^3) \times \mathscr{P}(E^3) \rightarrow \mathscr{P}(E^3)$ $\cap : \mathscr{P}(E^3) \times \mathscr{P}(E^3) \rightarrow \mathscr{P}(E^3)$ $- : \mathscr{P}(E^3) \times \mathscr{P}(E^3) \rightarrow \mathscr{P}(E^3)$

We can make an observation here that the class A_{CSG} of Constructive Solid Geometry and the Boolean-Only Opacity Model A_{1cc} have the same algebraic structure, that is, they are *isomorphic*. This proof involves defining a mapping $\epsilon : \mathscr{P}(E^3) \rightarrow F(E^3, \mathbb{B})$ from the family of all sets of E^3 to the set of boolean-valued scalar fields. To show that the operations behave in the correct manner, we would need to verify the following equations for subsets A and B of E^3.

$$\begin{aligned}
\epsilon(A \cup B) &= \epsilon(A)\boxed{\cup}\epsilon(B) \\
\epsilon(A \cap B) &= \epsilon(A)\boxed{\cap}\epsilon(B) \\
\epsilon(A - B) &= \epsilon(A)\boxed{-}\epsilon(B).
\end{aligned}$$

This result is proved in Theorem 4.5.1 of Chapter 4, after more sophisticated mathematical tools have been introduced.

2.4.3 The 4-Colour Channel Model

A particularly useful example of a CVG model is the 4-colour channel model, which consists of data attributes for opacity, red, green and blue. This model used in [48] and later modified by [37] for use in volume visualisation. We declare the 4-colour channel spatial objects as follows:

Object Declaration	Σ_{4cc}
Space	E^3
Attributes	$[0,1]$, $\mathbb{R}$
Fields	$O : E^3 \rightarrow [0,1]$ $R : E^3 \rightarrow \mathbb{R}$ $G : E^3 \rightarrow \mathbb{R}$ $B : E^3 \rightarrow \mathbb{R}$

We define the set $O(\Sigma_{4cc})$ of all four-colour channel model spatial objects as

$$O(\Sigma_{4cc}) \quad = \quad \{\mathbf{o} \mid \mathbf{o} : E^3 \rightarrow [0,1] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}\}.$$

Furthermore, we define some operations on these objects in the class

CVG Class	A_{4cc}
Spatial Objects	$O(\Sigma_{4cc})$
Operations	$\boxed{\cup} : O(\Sigma_{4cc}) \times O(\Sigma_{4cc}) \rightarrow O(\Sigma_{4cc})$ $\boxed{\cap} : O(\Sigma_{4cc}) \times O(\Sigma_{4cc}) \rightarrow O(\Sigma_{4cc})$ $\boxed{-} : O(\Sigma_{4cc}) \times O(\Sigma_{4cc}) \rightarrow O(\Sigma_{4cc})$

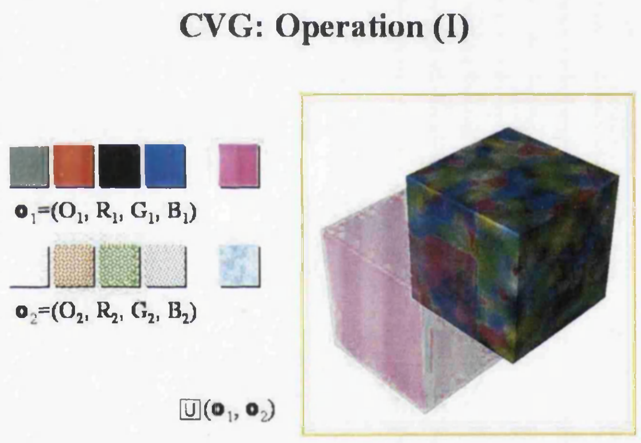

Figure 2.3: The union of two Σ_{4cc} spatial objects $\mathbf{o}_1$ and $\mathbf{o}_2$

Let $\mathbf{o}_1$ and $\mathbf{o}_2$ with scalar fields O_1, R_1, G_1, B_1 and O_2, R_2, G_2, B_2 respectively. The operations for CVG spatial objects in the 4-colour channel model are displayed in Figures 2.3 and 2.4 and are defined as follows:

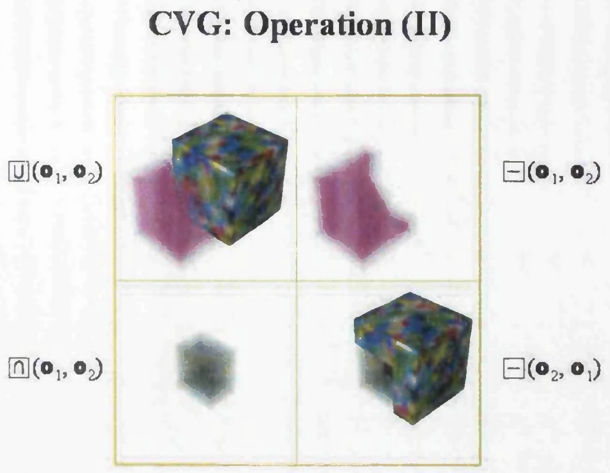

Figure 2.4: 4_{cc} Operations with two Σ_{4cc} spatial objects $\mathbf{o}_1$ and $\mathbf{o}_2$

$$\begin{aligned}
\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2) &= \big(Max(O_1, O_2), Select\big((O_1, R_1), (O_2, R_2)\big),\\
&\quad Select\big((O_1, G_1), (O_2, G_2)\big), Select\big((O_1, B_1), (O_2, B_2)\big)\big)\\
\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2) &= \big(Min(O_1, O_2), Select\big((O_1, R_1), (O_2, R_2)\big),\\
&\quad Select\big((O_1, G_1), (O_2, G_2)\big), Select\big((O_1, B_1), (O_2, B_2)\big)\big)\\
\boxed{-}(\mathbf{o}_1, \mathbf{o}_2) &= \big(O_1 - O_2, R_1, G_1, B_1\big)
\end{aligned}$$

2.5 Notes and Sources

This chapter is based on the paper [19] and Technical Report [18], both by Chen and Tucker.

A basic rendering algorithm developed for the CVG framework can be found in the Chen and Tucker paper [19]. Given a CVG term we aim to render each component of the term (or equivalently, each subtree in the CVG tree) as independently as possible using structural induction. However, we do not study the problem in this thesis, but leave it as an open problem for future research.

A brief overview of Constructive Volume Geometry and its relation to spatial data types was presented at the British Colloquium for Theoretical Computer Science (BCTCS) in 2006, and parts of the chapter are based on that presentation [35].

Another PhD which applies constructive volume geometry is [1]. In that work, algebraic properties of abstract operations on volume objects are studied so that one can design them to be more consistent and useful, and discover general laws for their behavior. An algebraic framework with operations specially for volume objects with complex physical properties is designed, and a physically-based rendering algorithm based on the Kubelka-Munk Theory of diffuse reflectance is developed.

Constructive Volume Geometry is being developed largely through practical work in graphics. The CVG framework has been implemented in the *vlib* API library by Andrew S. Winter and Min Chen [21].

Chapter 3

Algebraic and Topological Preliminaries

3.1 Introduction

The scientific study of spatial data types will require concepts and results from many areas of mathematics.

In these preliminaries we will list some basic definitions, lemmas and theorems from areas of universal algebra, topology and analysis with more advanced material introduced when needed. We establish notation that will be used throughout this thesis.

3.2 Many-Sorted Algebras

We model abstract data types with many-sorted algebras. Tucker and Stephenson give a complete account of their role in the study of data types in their textbook [55]. They are also listed in several other papers [43], [59] and [62] to name a few. We give only the basic definitions needed in our study, beginning with many-sorted signatures and settle on the notation of [59].

Definition 3.2.1 (Many-Sorted Signatures)
A signature Σ (for a many-sorted algebra) is a pair consisting of

1. *a finite set S of sorts, and*
2. *a finite set $Func(\Sigma)$ of (primitive or basic) function symbols, each symbol f having a type $s_1 \times \cdots \times s_n \longrightarrow s$ where $n \geq 0$ is the arity of f and $s_1, \ldots, s_n \in S$ are the domain sorts and $s \in S$ is the range sort; in such a case we write*
$$f : s_1 \times \cdots \times s_n \longrightarrow s.$$

The case $n = 0$ corresponds to constant symbols; we then write $f : \longrightarrow s$.

We will sometimes display the signature Σ as

Signature	Σ
Sorts	$\ldots, s, \ldots$
Constants	$\ldots, c :\rightarrow s, \ldots$
Operations	$\ldots, f : s_1 \times \cdots \times s_n \rightarrow s, \ldots$

Definition 3.2.2 (Product Types Over Σ) *A product type over Σ, or Σ-product type is a symbol of the form $s_1 \times \cdots \times s_n$ $(n \geq 0)$, where $s_1, \ldots, s_n$ are sorts of Σ called its component sorts. We define ProdType(Σ) to be the set of Σ-product types.*

For a Σ-product type $u = s_1 \times \cdots \times s_n$, we define $A^u = A_{s_1} \times \cdots \times A_{s_n}$. Thus x is an element of the product A^u if, and only if, $x = (x_1, \ldots, x_n)$, where $x_i \in A_{s_i}$ for $1 \leq i \leq n$. Thus each Σ-function symbol $f : u \rightarrow s$ has an interpretation in the Σ-algebra A as $f_A : A^u \rightarrow A_s$. If u is empty, denoted by the symbol λ, then $f :\rightarrow s$ is a constant symbol and f_A is an element of A_s.

For a Σ-product type u and Σ-sort s, let Σ_s^u denote the set of all Σ-function symbols of type $u \rightarrow s$. The set of all constants of type s is denoted Σ_s^λ.

Definition 3.2.3 (Many-Sorted Algebras) *A Σ-algebra A has, for each sort $s \in \Sigma$, a non-empty set A_s called the carrier of sort s, and for each Σ-function symbol $f : s_1 \times \cdots \times s_n \rightarrow s$, a function $f_A : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s$.*

We can display the Σ-algebra A as follows

Algebra	A
Carriers	$\ldots, A_s, \ldots$
Constants	$\ldots, c_A :\rightarrow A_s, \ldots$
Operations	$\ldots, f_A : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s, \ldots$

3.2.1 The Import Construct

An additional construction commonly used in the definition of signatures and algebras is the *import* construct. This construction allows common data types with their data and operations, to be reused in multiple data types. It is useful when we wish to describe precisely the addition of new sets and operations to a signature or algebra. We give an example where a signature Σ is imported by another signature Σ'. The two signatures are displayed as follows:

Signature	Σ
Sorts	S
Constants	$c :\to s$
Operations	$f : s_1 \times \cdots \times s_n \to s$

Signature	Σ'
Import	Σ
Sorts	S'
Constants	$c' :\to s'$
Operations	$f' : s'_1 \times \cdots \times s'_n \to s'$

From [55](p.121), an intuitive idea of the import notation is given by:

- substituting all the components of the signature Σ' named in **Import** into the **Sorts**, **Constants** and **Operations** declared after **Import** in Σ; and
- allowing any sort to be included in the type of any new operation.

The import construct does not add anymore expressive power to many-sorted signatures, we use it for convenience. We can *flatten* Σ' by removing the **Import** declaration and making the necessary substitutions:

Signature	Σ'
Sorts	$S \cup S'$
Constants	$c :\rightarrow s,\ c' :\rightarrow s'$
Operations	$f : s_1 \times \cdots \times s_n \rightarrow s$ $f' : s'_1 \times \cdots \times s'_n \rightarrow s'$

Similar remarks hold for the import construct of algebras.

3.2.2 Subalgebras

Now for a signature Σ and an interpreting Σ-algebra A, we consider subsets of A that are closed under the Σ-operations. We call such subsets Σ-subalgebras, are they are defined as follows:

Definition 3.2.4 (Subalgebras) *[43](p.233) Let A and B be S-sorted Σ-algebras. Then B is said to be a Σ-subalgebra of A if, and only if, for each sort $s \in S$, $B_s \subseteq A_s$; for each constant symbol $c :\rightarrow s$*

$$c_B = c_A;$$

and function symbol $f : u \rightarrow s$ where $u = s_1 \times \cdots \times s_n$ and any $(b_1, \ldots, b_n) \in B^u$,

$$f_B(b_1, \ldots, b_n) = f_A(b_1, \ldots, b_n).$$

We use the notation $B \leq A$ to mean B is a subalgebra of A.

Definition 3.2.5 (Generated Subalgebras) *Let A be any Σ-algebra and $S \subseteq A$ be any subset of A. The subset $\langle S\rangle_A$ of A* generated *by S is the set*

$$\langle S\rangle_A = \bigcap\{B \mid S \subseteq B \text{ and } B \leq A\}.$$

We often refer to the elements of S as the *generators* of the Σ-subalgebra $\langle S\rangle_A$.

3.3 Many-Sorted Terms

By applying the operations of the signature Σ on variables contained in a set Y, we are constructing expressions or *terms* over Σ. In this section we define rules on how we may construct many-sorted terms for a general signature Σ, following [55](p.281).

Definition 3.3.1 (Many-Sorted Terms) *Let $Y = \langle Y_s \mid s \in S \rangle$ be a family of non-empty sets of variables indexed by the sort set S of Σ. We define the set $T(\Sigma, Y)_s$ of all terms of sort s by induction from the following rules*

(i) each constant symbol $c :\rightarrow s$ is a term of sort s,

(ii) each variable symbol $y \in Y_s$ is a term of sort s,

(iii) for each function symbol $f : s_1 \times \cdots \times s_n \rightarrow s$ and any terms t_1 of sort $s_1, \ldots, t_n$ of sort s_n the expression $f(t_1, \ldots, t_n)$ is a term of sort s, and

(iv) nothing else is a term.

We denote

$$T(\Sigma, Y) = \langle T(\Sigma, Y)_s \mid s \in S \rangle$$

to be the family of all Σ-terms indexed by the sort set S.

Let Σ be an S-sorted signature. Roughly speaking, term evaluation over a Σ-algebra A is a mapping from the set of terms $T(\Sigma, Y)$ to A that works out the value of a term in A. We define the collection of *assignment functions* that map variables in Y to data in A_s as $V(Y, A)_s = \{v_s : Y_s \rightarrow A_s\}$, for a sort $s \in S$ and then

$$V(Y, A) = \langle V(Y, A)_s \mid s \in S \rangle.$$

We use these assignment functions to define a function mapping terms in $T(\Sigma, Y)$ to data elements in the algebra A. Such a mapping is a *term evaluation* function. This definition of term evaluation for a Σ-algebra A is a modification of the definition found in [43] and [55]:

Definition 3.3.2 (Many-Sorted Term Evaluation) *Given an S-sorted family $v = \langle V_s : Y_s \rightarrow A_s \mid s \in S \rangle$ of assignments of elements $v_s(y) \in A_s$ to variables $y \in Y_s$, we define the family*

$$[\![-]\!]_A^v = \{[\![-]\!]_s^v : T(\Sigma, Y)_s \times V(Y, A)_s \rightarrow A_s \mid s \in S\}$$

of term evaluation maps derived from v by induction on the structure of terms: for every sort $s, s_1, \ldots, s_n \in S$, constant $c :\rightarrow s \in \Sigma$, variable $y \in Y_s$, operation $f : s_1 \times \cdots \times s_n \rightarrow s$ in Σ and terms $t_1 \in T(\Sigma_Y)_{s_1}, \ldots, t_n \in T(\Sigma_Y)_{s_n}$, we have

$$\begin{aligned} [\![c]\!]_s^v &= c_{A_s} \\ [\![y_i]\!]_s^v &= v(y_i) \\ [\![f(t_1, \ldots, t_n)]\!]_s^v &= f_A([\![t_1]\!]_{s_1}^v, \ldots, [\![t_n]\!]_{s_n}^v). \end{aligned}$$

3.4 Commutative Rings

A commutative ring is an algebraic structure containing a set A, an additive operation $+$ and multiplicative operation $\cdot$ which satisfy certain laws. We specify these operations as follows:

Signature	Σ_{CRing}
Sorts	$CRing$
Constants	$0 :\to CRing$ $1 :\to CRing$
Operations	$+ : CRing \times CRing \to CRing$ $\cdot : CRing \times CRing \to CRing$ $- : CRing \to CRing$

We list the laws, or axioms for a commutative ring. These axioms have been selected since they represent properties we are accustomed to when calculating, for example, with the integers.

Axioms	T_{CRing}
Associativity of addition	$(\forall x)(\forall y)(\forall z)[(x+y)+z = x+(y+z)]$
Identity of addition	$(\forall x)[x+0 = x]$
Inverse of addition	$(\forall x)[x+(-x) = 0]$
Commutivity of addition	$(\forall x)(\forall y)[x+y = y+x]$
Assoc. of multiplication	$(\forall x)(\forall y)(\forall z)[(x\cdot y)\cdot z = x\cdot(y\cdot z)]$
Comm of multiplication	$(\forall x)(x\cdot y = y\cdot x)$
Identity for multiplication	$(\forall x)[x\cdot 1 = x]$
Distributivity	$(\forall x)(\forall y)(\forall z)[x\cdot(y+z) = x\cdot y + x\cdot z]$

Definition 3.4.1 (Commutative Rings) *[39] A Σ_{CRing}-algebra A is a ring if, and only if, it satisfies the axioms of T_{CRing}, i.e., $R \in Alg(\Sigma_{CRing}, T_{CRing})$.*

It is easy to show that the operations of $+$ and $\cdot$ on the real numbers satisfy these equations (as the use of 0 and 1 in the axioms imply). Other classical examples of commutative rings are

- the integers $\mathbb{Z}$,
- the integers mod n where $\mathbb{Z}_n = \{0, \ldots, n-1\}$,
- the rational numbers $\mathbb{Q}$,
- the complex numbers $\mathbb{C}$, and
- the ring of polynomials $R[x]$ where $R \in Alg(\Sigma_{CRing}, T_{CRing})$.

3.5 Vector Spaces

The axioms of a vector space can be found in any book covering algebraic structures, for example [52].

Signature	$Scalar$
Sorts	$Scalar$
Operations	$+ : Vector \times Vector \to Vector$
	$\cdot\ : Scalar \times Vector \to Vector$

Signature	Σ_{Vector}
Import	$Scalar$
Sorts	$Vector$
Operations	$+ : Vector \times Vector \to Vector$
	$\cdot\ : Scalar \times Vector \to Vector$

The axioms which the vector space satisfies are listed as follows for $u, v, w \in Vector$ and $a, b, c \in Scalar$:

Axioms	T_{Vector}
Associativity of addition	$(\forall u, v, w)(u + v) + w = u + (v + w)]$
Identity of addition	$(\forall u)[u + 0 = u]$
Inverse of addition	$(\forall u)[u + (-u) = 0]$
Commutativity of addition	$(\forall u, v)[u + v = v + u]$
Identity	$(\forall u)[1 \cdot u = u]$
Associativity	$(\forall a, b)(\forall u)[(a \cdot b) \cdot u = a \cdot (b \cdot u)]$
Distribution	$(\forall a, b)(\forall u, v)[a \cdot (u + v) = a \cdot u + a \cdot v]$ $[(a + b) \cdot v = a \cdot u + b \cdot u]$

Definition 3.5.1 (Vector Spaces) *A Σ_{Vector}-algebra V is defined to be a vector space if, and only if, it satisfies the axioms listed in T_{Vector}. i.e., $V \in Alg(\Sigma_{Vector}, T_{Vector})$.*

3.6 Ordered Sets

Orderings are binary relations between mathematical objects. The most familiar orderings to us are that of the number systems, such as the binary relation $\leq$ for the natural numbers $\mathbb{N}$ or $>$ on the real numbers $\mathbb{R}$. The following definitions generalise such orderings to situations where we wish to compare, for example, sets using the inclusion relation $\subseteq$ or functions of the real numbers. The source of these definitions and examples are from [26] and [52].

Definition 3.6.1 (Preordered sets) *Let A be a non-empty set. A preorder relation in A is a binary relation R satisfying the following properties*

1. *for all $x \in A$: xRx (Reflexivity),*

2. *$x \leq y$ and $y \leq z$ implies $x \leq z$ (Transitivity).*

Definition 3.6.2 (Partially Ordered Sets) *Let A be a preordered set. A partial order relation in A is a binary relation $\leq$ satisfying the following property:*

3. *$x \leq y$ and $y \leq x$ implies $x = y$ (Antisymmetry).*

We usually called partially ordered sets *posets.* Examples of posets appear throughout mathematics, such as the binary relation of set inclusion:

Example 3.6.1 [52] Let P be the power set of some universe U. Then for the binary relation of set inclusion $\subseteq$, we can easily see that for any subsets $A, B, C \subseteq U$ the following properties are satisfied:

(i) $A \subseteq A$,

(ii) $A \subseteq B$ and $B \subseteq C$ implies $A \subseteq C$,

(iii) $A \subseteq B$ and $B \subseteq A$ imply $A = B$.

In addition, we shall need another ordering property for sets:

Definition 3.6.3 (Directed Set) *A directed set D is a preordered set with the following property*

4. for each $x, y \in D$, there exists a $z \in D$ such that $x < z$ and $y < z$.

Definition 3.6.4 *[52](p.44) Let B be a non-empty subset of a partially ordered set A. An element $x \in A$ is called a lower bound of A if $x \leq a$ for each $a \in A$. An element y of A is said to be an upper bound of A if $a \leq y$ for every $a \in A$.*

A lower bound of A is called a greatest lower bound (g.l.b) or the infimum (inf) if it is greater than or equal to every lower bound of A. An upper bound of A is called a least upper bound (l.u.b) or the supremum (sup) if it is less than, or equal to every upper bound of A. It can be shown ([44]) that both the g.l.b and l.u.b are unique, if they exist. Using this we define

Definition 3.6.5 (Lattice) *[52] A lattice is a partially ordered set L in which each pair of elements has a greatest lower bound and a least upper bound. For any $x, y \in L$ we denote the g.l.b and l.u.b of x and y as $x \wedge y$ and $x \vee y$, called the meet and join of x and y respectively.*

Lattices are important in this work, and we look at them in greater detail in Chapter 4 Section 4.4.

3.7 Algebraic Constructions

When we have a family $A = \{A_i \mid i \in I\}$, of Σ-algebras, a natural algebraic structure to define is the *product* $\prod A$. We give this construction in detail, and show how the product of Σ-algebras itself can be made into an Σ-algebra. We also give the construction of an important subalgebra of $\prod A$ called the inverse limit over a family of Σ-algebras.

3.7.1 Products of Σ-Algebras

We use products of Σ-algebras throughout our work, so we list some definitions from [26](p.22 – 24) and [43](p.276 – 277).

Definition 3.7.1 *[43] Let $A = \langle A_i \mid i \in I \rangle$ be an I-indexed family of S-sorted Σ-algebras, for some (possibly empty) indexing set I. The product of A, is the S-sorted Σ-algebra $\prod A = \prod_{i \in I} A_i$ with S-indexed family of carrier sets*

$$\prod A = \langle (\prod A)_s \mid s \in S \rangle,$$

where each carrier set $(\prod A)_s$ is the product of the sets $(A_i)_s$ given by

$$(\prod A)_s = \{ f : I \to \cup_{i \in I} (A_i)_s \mid f(i) \in A_{is} \text{ for all } i \in I \}.$$

For each sort $s \in S$ and each constant symbol $c \in \Sigma_s^{\lambda}$,

$$c_{\prod A}(i) = c_{A_i}.$$

For each non-empty Σ-product type $u = s_1 \times \ldots \times s_n$ each sort $s \in S$ and each function symbol $f \in \Sigma_s^u$ and any $(a_1, \ldots, a_n) \in (\prod A)^u$,

$$f_{\prod A}(a_1, \ldots, a_n)(i) = f_{A_i}(a_1(i), \ldots, a_n(i)).$$

The *projection* of the product $\prod A$ to a single coordinate space is a function $\pi_i : \prod A \to A_i$ where $\pi_i(a) = a_i$ is an element in the i-th coordinate set A_i. For a given $j \in I$ and $C_j \subset A_j$ we write the preimage $\pi_j^{-1}[C_j]$ of the projection function π_j as $\langle C_j \rangle$, where each factor in $\prod_{i \in I} A_i$ is A_i except at j, where the factor is C_j. For finitely many indices $i_1, \ldots, i_n$, and sets $C_{i_1} \subset A_{i_1}, \ldots, C_{i_n} \subset A_{i_n}$, we denote $\pi_{i_1}^{-1}[C_{i_1}] \cap \cdots \cap \pi_{i_n}^{-1}[C_{i_n}]$ by $\langle C_{i_1}, \ldots, C_{i_n} \rangle$.

We can define a Σ-algebra with the product of Σ-algebras as the carrier where the operations on $\prod_{i \in I} A_i$ are pointwise lifted.

Algebra	$\prod A$
Carriers	$\prod A$
Constants	$\ldots, c_{\prod A}(i) = c_{A_i}, \ldots$
Operations	$\ldots, f_{\prod A} : \prod A \times \cdots \times \prod A \to \prod A, \ldots$
Definitions	$\ldots, f_{\prod A}(a_1, \ldots, a_n)(i) = f_{A_i}(a_1(i), \ldots, a_n(i)), \ldots$

3.7.2 General Construction of an Inverse System

Let Σ be a general single sorted signature. The inverse system shall be defined over a family $A = \{A_i \mid i \in I\}$ of Σ-algebras where I is the index set.

Definition 3.7.2 (Inverse Systems) *[43](p.204) An inverse system of Σ-algebras consists of:*

(i) a directed partially ordered set $(I, \leq)$; i.e: for each i, j in I there exists a k such that $i \leq j \leq k$.

(ii) an indexed family $A = \{A_i \mid i \in I\}$ of Σ-algebras; and

(iii) an indexed family of Σ-homomorphisms $\phi^i_j : A_i \to A_j$, for each $i \geq j$ such that

$$\phi^j_k \circ \phi^i_j = \phi^i_k$$

for all $i \geq j \geq k$ and ϕ^i_i is the identity map for each index $i \in I$.

We write this inverse system as $S(A) = \{A, \phi^i_j, I\}$.

The *limit* of an inverse system $S(A)$ is defined to be the set

$$\varprojlim A = \{a \in \prod A \mid \text{ for all } i \geq j \ \ \phi^i_j(a_i) = a_j\}.$$

We denote the projection map restricted to the inverse limit as

$$\phi_i : \pi_i \mid \varprojlim A : \varprojlim A \to A_i$$

which defines a homomorphism from the inverse limit to the coordinate space A_i. [32](p.119)

Lemma 3.7.1 *The limit $\varprojlim A$ of the inverse system $S(A)$ is a Σ-subalgebra of the product $\prod A$.*

Proof The carrier set $\varprojlim A$ is a subset of $\prod A$ by definition. To show $\varprojlim A \leq \prod A$, we note that for each constant $c \in \Sigma^\lambda_s$ we have

$$c_{\prod A} = c_{A_1} \times c_{A_2} \times \cdots$$

as the interpretation in the Σ-algebra $\prod A$. Then for any $i \geq j$ in I and Σ-homomorphism $\phi^i_j : A_i \to A_j$ we have

$$\begin{aligned} \phi^i_j(c_{\prod A}(i)) &= \phi^i_j(c_{A_i}) \\ &= c_{A_j} \\ &= c_{\prod A}(j) \end{aligned}$$

and hence $c_{\prod A} \in \varprojlim A$.

Now consider any Σ-operation $f \in \Sigma_s^{s^n}$ and let $a_1, \ldots, a_n$ be elements of $\varprojlim A$. We want to show that the product

$$f_{A_1}(a_1(1), \ldots, a_n(1)) \times f_{A_2}(a_1(2), \ldots, a_n(2)) \times \cdots$$

is in the carrier set $\varprojlim A$. That is, for a Σ-homomorphism $\phi_j^i : A_i \to A_j$ with $i \geq j \in I$ the condition

$$\phi_j^i(f_{\prod A}(a_1, \ldots, a_n)(i)) = f_{\prod A}(a_1, \ldots, a_n)(j)$$

is true for any elements $a_1, \ldots, a_n$ of $\varprojlim A$. By definition we have

$$\begin{aligned}
\phi_j^i(f_{\prod A}(a_1, \ldots, a_n)(i)) &= \phi_j^i(f_{A_i}(a_1(i), \ldots, a_n(i))) \\
&= f_{A_j}(\phi_j^i(a_1(i)), \ldots, \phi_j^i(a_n(i))) \\
&= f_{A_j}(a_1(j), \ldots, a_n(j)) \\
&= f_{\prod A}(a_1, \ldots, a_n)(j)
\end{aligned}$$

showing that the condition

$$\phi_j^i(f_{\prod A}(a_1, \ldots, a_n)(i)) = f_{\prod A}(a_1, \ldots, a_n)(j)$$

holds, proving the result. ∎

We display the Σ-subalgebra $\varprojlim A$ as follows

Algebra	$\varprojlim A$
Carriers	$\varprojlim A$
Constants	$\ldots, c_{\varprojlim A}, \ldots$
Operations	$\ldots, f_{\varprojlim A} : \varprojlim A \times \cdots \times \varprojlim A \to \varprojlim A, \ldots$
Definitions	$\ldots, f_{\varprojlim A}(a_1, \ldots, a_n)(i) = f_{A_i}(a_1(i), \ldots a_n(i)), \ldots$

3.8 Topology

This section gives several basic definitions and results from topology, taken from the standard texts in topology [26] and [36].

Definition 3.8.1 (Topological Spaces) *Let X be a set. A topology in X is a family $\mathscr{T}$ of subsets of X that satisfies*

1. *Each union of members of $\mathscr{T}$ is also a member of $\mathscr{T}$,*

2. *Each finite intersection of members of $\mathscr{T}$ is also a member of $\mathscr{T}$,*

3. *$\emptyset$ and X are members of $\mathscr{T}$.*

Definition 3.8.2 *A pair $(X, \mathscr{T})$ consisting of a set X and a topology $\mathscr{T}$ in X is called a topological space.*

Definition 3.8.3 (Open Sets) *The members of the topology $\mathscr{T}$ are called open sets.*

Definition 3.8.4 (Closed Sets) *The set $A \subset X$ is called closed if its complement $C_X A$ is an open set*[1]*.*

Definition 3.8.5 *[26](p.63) Let $(X, \mathscr{T})$ be a space. By a neighborhood (written nbhd) of an $x \in X$ is meant any open set (that is, member of $\mathscr{T}$) containing x.*

Definition 3.8.6 (Closure) *[26](p.69) Let $A \subset X$. The set $\overline{A} = \{x \in X \mid \forall U(x) : U(x) \cap A \neq \emptyset\}$ of all points in $x \in X$ such that each nbhd $U(x)$ of x contains at least one point of A (which may be x itself), is called the closure of A.*

Some facts about the closure of a set will be useful:

Lemma 3.8.1 *A is closed if, and only if, $A = \overline{A}$.*

Lemma 3.8.2 *$\overline{\overline{A}} = \overline{A}$. That is, $\overline{A}$ is closed.*

Definition 3.8.7 (Basis for a Topology) *[26](p.64) Let $(X, \mathscr{T})$ be a topological space. A family $\mathscr{B} \subset \mathscr{T}$ is called a basis for $\mathscr{T}$ if each member of $\mathscr{T}$ is the union of members of $\mathscr{B}$.*

From [26](p.64 – 65) we have the result

Theorem 3.8.1 *Let $\mathscr{B} \subset \mathscr{T}$. The following two properties of $\mathscr{B}$ are equivalent:*

1. *$\mathscr{B}$ is a basis for $\mathscr{T}$.*

2. *For each $G \in \mathscr{T}$ and each $x \in G$ there is a $U \in \mathscr{B}$ with $x \in U \subset G$.*

Proof (1) $\Rightarrow$ (2). Let $x \in G$; since $G \in \mathscr{T}$ and $\mathscr{B}$ is a basis, $G = \cup_\alpha U_\alpha$, where each $U_\alpha \in \mathscr{B}$. Thus there is at least one $U_\alpha \in \mathscr{B}$ with $x \in U_\alpha \subset G$. (2) $\Rightarrow$ (1). Let $G \in \mathscr{T}$; for each $x \in G$, find $U_x \in \mathscr{B}$ with $x \in U_x \subset G$; then $G = \cup\{U_x \mid x \in G\}$. ■

[1]If $A \subset X$, the complement $C_X A$ of A with respect to X is $X - A$ [26](p.5)

3.8.1 Separation Axioms

Recall the separation hierarchy from [36]. We use many of the separation properties of certain spaces throughout this work. We list them as:

T_1 **space** A topological space is a T_1-space if, and only if, each set which consists of a single point is closed.

T_2 **space (Hausdorff)** A topological space is a Hausdorff space (T_2-space) if, and only if, whenever x and y are distinct points of the space there exists disjoint nbhds x and y.

T_3 **space (Regular)** A Hausdorff space is regular (or T_3) if each $y \in Y$ and closed set A not containing y have disjoint nbhds; i.e. if A is closed and $y \notin A$ then there is a nbhd U of y and an open $V \supset A$ such that $U \cap V = \emptyset$.

T_4 **space (Normal)** A space is normal if, and only if, for each disjoint pair of closed sets A and B, there are disjoint open sets U and V such that $A \subset U$ and $B \subset V$. A T_4-space is a normal space which is T_1.

3.8.2 Compact Spaces and Dense Sets

This section reviews the notions of covers, subcovers and compactness. A definition of density is given and we list some equivalent statements of this definition. We start with compact spaces and follow the definitions of [52].

Definition 3.8.8 (Open Covers) *Let $T = (\mathscr{T}, X)$ be a topological space. A class $\{G_i\}$ of open subsets in the family $\mathscr{T}$ is said to be an open cover of T if each point in X belongs to at least one G_i. That is,*

$$\bigcup_{i \in I} G_i = X.$$

A subclass of an open cover which is itself an open cover is called a subcover.

Definition 3.8.9 (Compactness) *A topological space $T = (\mathscr{T}, X)$ is compact if, and only if, each open cover has a finite subcover.*

Definition 3.8.10 (Relative Compactness) *[26](p.237) A subset A of X is relatively compact whenever its closure $\bar{A}$ is compact.*

We list some useful definitions and terminology for this section from [51].

Definition 3.8.11 (Local Compactness)
A Hausdorff topological space A is said to be local compact if every point of A has a compact neighborhood.

Definition 3.8.12 (Refinements) *[26](p.161) Let $A = \{A_i \mid i \in I\}$ and $B = \{B_j \mid j \in J\}$ be two coverings of a space Y. The family A is said to refine (or be a refinement of) B if for each A_i there is some B_j with $A_i \subset B_j$.*

Definition 3.8.13 (Paracompact Spaces) *[26](p.162)*
A Hausdorff space Y is paracompact if each open covering of Y has an open nbhd-finite refinement.

Definition 3.8.14 (Density) *Let X be a space. The subset $D \subset X$ is dense in X if $\overline{D} = X$.*

The following theorem gives some useful equivalences of this definition:

Theorem 3.8.2 *[26](p.72) The following two statements are equivalent:*

1. *D is dense in X,*
2. *Each non-empty basic open set in X contains an element of D.*

In fact, we could add more statements to the list in this theorem, but the ones we have stated are suitable for our purposes.

3.8.3 The Product Topology

From [26](p.98) we have the standard definition:

Definition 3.8.15 (Subbasis of Products) *Let $\{A_i \mid i \in I\}$ be a family of topological Σ-algebras. For each $i \in I$ let $\mathscr{T}_i$ be the topology for A_i. The product has a subbase consisting of all sets $\langle U_i \rangle = \pi_i^{-1}[U_i]$ where U_i ranges over all members of $\mathscr{T}_i$ and i over all elements of the index set I.*

Now, from [26](p.98) we have that the basic open sets of $\prod A_i$ are of the form

$$U_{\alpha_1} \times \cdots \times U_{\alpha_n} \times \prod\{A_j \mid j \neq \alpha_1, \ldots, \alpha_n\} = \bigcap_{i=1}^{n} \pi_{\alpha_i}^{-1}[U_{\alpha_i}]$$

where n is finite and U_{α_i} is open in A_{α_i}. We will denote this basic open set of the product topology by

$$\langle U_{\alpha_1}, \ldots, U_{\alpha_n} \rangle.$$

3.8.4 Metrics and Distance

Metrics give a general notion of distance between two points in a set X. We give the following definitions:

Definition 3.8.16 (Metric Spaces) *[52](p.51) Let X be a non-empty set. A metric on X is a function $d : X \times X \rightarrow \mathbb{R}$ which satisfies the following conditions:*

1. *$d(x,y) \geq 0$ and $d(x,y) = 0$ if, and only if, $x = y$,*
2. *$d(x,y) = d(y,x)$,*
3. *$d(x,z) \leq d(x,y) + d(y,z)$.*

For a metric $d : X \times X \rightarrow \mathbb{R}$ we define the set

$$B_d(a,r) = \{x \mid d(x,a) < r\},$$

called the d-sphere of radius r and center a. Using such sets, we can define a basis for a topology:

Lemma 3.8.3 *[26] The family $\{B_d(a,r) \mid x \in X, r > 0\}$ of all d-spheres in X can serve for a basis for a topology.*

In light of the previous lemma, we can make the following definition

Definition 3.8.17 *[26] Let X be a set and d be a metric in X. The topology $\mathscr{T}$ having for basis the family*

$$\{B_d(a,r) \mid x \in X, r > 0\}$$

of all d-spheres in X, is called the topology in X induced by the metric d.

Definition 3.8.18 *[52](p.54), Let X be a non-empty set. A norm is a mapping $\| - \| : X \rightarrow \mathbb{R}^+$ satisfying the following properties:*

1. *$\|x\| \geq 0$ and $\|x\| = 0 \iff x = \mathbf{0}$,*
2. *$\| -x\| = \|x\|$,*
3. *$\|x + y\| \leq \|x\| + \|y\|$.*

where x, y are points in X.

We observe that a metric can be defined on a space X by way of a norm:

$$d(x,y) = \|x - y\|$$

for $x, y \in X$.

Definition 3.8.19 (Normed Vector Spaces) *[5](p.18) If V is a real vector space (i.e. where the scalars are real numbers), a norm on V is a mapping $\| - \| : V \to \mathbb{R}^+$ such that*

1. $\|v\| \geq 0$ *and* $\|v\| = 0 \iff v = \mathbf{0}$,
2. $\|v + w\| \leq \|v\| + \|w\|$,
3. $\|\lambda v\| = |\lambda| \|v\|$

3.8.5 Continuity and Uniform Continuity

We give the definition of continuity and some related terminology. Basic notation and ideas of functions (or mappings) from one set to another are in any topology text, for example [26] or [36] and therefore are not listed here.

Definition 3.8.20 (Continuity) *Let $(X, \mathscr{T}_X)$ and $(Y, \mathscr{T}_Y)$ be spaces. A map $f : X \to Y$ is called continuous if the inverse image of each set open in Y is open in X (that is, f^{-1} maps $\mathscr{T}_Y$ into $\mathscr{T}_X$).*

We usually use this topological notion of continuity, but it is also useful to consider the analytical definition. We give these definitions in terms of general metric spaces.

Definition 3.8.21 *[5](p.79 – 80) Let f be a mapping from a metric space (X, d_1) to a metric space (Y, d_2). f is continuous at the point x in the domain of f dom(f) if for all $\epsilon > 0$ there exists a $\delta = \delta(\epsilon, x)$ such that*

$$d_1(x, a) < \delta \implies d_2(f(x), f(a)) < \epsilon.$$

Definition 3.8.22 (Continuous Functions) *We say that f is continuous if it is continuous at every point in its domain.*

There are many equivalent notions of continuity, and the one we use depends on the situation. We list a theorem which is used often in this work which lists some statements equivalent to the statement of Definition 3.8.22.

Theorem 3.8.3 *Let X, Y be topological spaces, and $f : X \to Y$ a map. The following statements are equivalent:*

1. *f is continuous.*
2. *The inverse image of each closed set in Y is closed in X.*
3. *The inverse image of each member of a subbasis (basis) for Y is open in X (not necessarily a member of a subbasis, or basis for X).*

4. *For each $x \in X$ and each nbhd $W(f(x))$ in Y, there exists a nbhd $V(x)$ in X such that $f[V(c)] \subset W[f(x)]$.*
5. *$f[\overline{A}] \subset \overline{f[A]}$ for every $A \subset X$.*
6. *$\overline{f^{-1}[B]} \subset f^{-1}[\overline{B}]$ for every $B \subset Y$.*

We give the definition of uniform continuity in terms of metric spaces from [5](p.81) and [52](p.77). Uniform continuity is a stronger condition than continuity in the sense that for any ϵ we can find a δ which works *uniformly* throughout the space X. That is, independent of any particular choice of element x in X.

Definition 3.8.23 (Uniform Continuity) *A function f from a metric space (X, d_1) to a metric space (Y, d_2) is said to be uniformly continuous if for all $\epsilon > 0$, there exists a $\delta = \delta(\epsilon)$ depending on ϵ and independent of x such that*

$$d_1(x,y) \leq \delta \implies d_2(f(x), f(y)) < \epsilon.$$

for $x, y \in X$.

3.9 Comparison of Algebras and Topological Spaces

3.9.1 Homomorphisms and Isomorphisms

To compare Σ-algebras we use Σ-homomorphisms and Σ-isomorphisms. We take the following definition from [43](p.257). These definitions are also found in [33] and [59].

Definition 3.9.1 (Σ-Homomorphisms) *[43](p.257)*
Let A and B be any S-sorted Σ-algebras. A Σ-homomorphism $\phi : A \to B$ is an S-indexed family of mappings

$$\phi = \langle \phi_s : A_s \to B_s \mid s \in S \rangle,$$

such that

1. *for each sort $s \in S$ and each constant symbol $c :\to s$*
$$c_B = \phi_s(c_A),$$
2. *for each function symbol $f : u \to s$ of Σ-product type $u = s_1 \times \cdots \times s_n$ and sort $s \in S$,*
$$\phi_s(f_A(a_1, \ldots, a_n)) = f_B(\phi_{s_1}(a_1), \ldots, \phi_{s_n}(a_n)),$$
for any element $(a_1, \ldots, a_n) \in A^u$.

If each mapping $\phi_s : A_s \to B_s$ is bijective, then ϕ is a Σ-isomorphism.

3.9.2 Homeomorphisms

We use the notion of continuity to compare topological spaces in the following definition.

Definition 3.9.2 (Homeomorphism) *[26](p.87)*
A continuous bijective map $f : X \to Y$ such that $f^{-1} : Y \to X$ is also continuous is called a homeomorphism and denoted by $f : X \cong Y$.

Two spaces X, Y are homeomorphic, written $X \cong Y$, if there exists a homeomorphism f between X and Y.

3.9.3 Topological Algebras

Definition 3.9.3 (Topological Σ-Algebras)
A topological Σ-algebra is a Σ-algebra with topologies on the carriers such that each of the basic Σ-operations is continuous.

3.10 Notes and Sources

This chapter gives a summary of terminology and basic results in algebra and topology. This is by no means a comprehensive list of the terminology and results used in my thesis. However, all results are either proved in the thesis or a citation is given.

All topological definitions or results mentioned will be found in any of standard topology textbooks such as Bourbaki [11], Engelking [27], Kelley [36], Simmons [52] and Dugundji [26]. Aubin [5] provides most of the definitions and results for metrics and norms.

The algebraic preliminaries can be found in standard texts [64] or [33]. Meinke and Tucker [43] is an excellent source of information.

The inverse limit plays an important role both algebraically and topologically in Chapters 6 and 7 thus we have given its construction in detail. Sources of this information are [26], [27] and [43].

Chapter 4

The Algebra of Spatial Objects

4.1 Introduction

In this chapter we define a general form of spatial objects and some operations on spatial objects. We consider constructing new spatial objects by the composition of operations and the laws the operations may satisfy. We study the algebraic structures on the data A and on the set of all spatial objects, denoted $F(X, A)$, and how they are related.

In Section 4.2 we formally define spatial objects. We define a Σ-algebra A containing data and operations on that data which is then used to construct a Σ-algebra $F(X, A)$ of spatial objects by the *pointwise extension* of operations on A. To further the usefulness of our algebra on $F(X, A)$, two additional operations on spatial objects are introduced: *evaluation* and *substitution.* Given a spatial object ϕ, and a point x in space X, evaluation works out the data value $\phi(x)$ at x. Substitution replaces the data value $\phi(x)$ at a point x with a given value a from the data set A.

Section 4.3 uses the set $T(\Sigma)$ of syntactic terms formed from the signature Σ. We define term evaluation on A and $F(X, A)$ as mapping terms to the Σ-algebras A and $F(X, A)$ respectively. We show that the data value of terms are equivalent when evaluated on either A or $F(X, A)$ (Lemma 4.3.1, Section 4.3). We prove theorems which show

- equational validity on A is equivalent to equational validity in $F(X, A)$,

and its extension

- conditional equational validity on A is equivalent to conditional equational validity on $F(X, A)$.

Applying this theorem, we can deduce immediately, e.g., if A is a commutative ring, then so is $F(X, A)$. Then, the algebraic specification of integral domains is used to show that valid universal formulae on A are not necessarily valid on $F(X, A)$.

Term definable functions are used to define a new Σ'-algebra A' which consists entirely of operations defined from the evaluation of Σ-terms. The main result shows that for any subset H of the carrier set A the set $\langle H\rangle_{\Sigma'}$ generated by the repeated application of the Σ'-operations is a subset of $\langle H\rangle_{\Sigma}$.

Section 4.4 discusses the topic of lattices, using the real numbers as our primary example. We define the lattice operations *meet* and *join* on a set L and list eight axioms that they satisfy. We deduce a result which shows that the pointwise lifted operations of meet and join also satisfy the lattice axioms on the set $F(X, L)$. We define sublattices, and use the unit interval $[0,1]$ as an important example of a sublattice of the real numbers.

4.2 Algebras of Spatial Objects

In this section we define a many-sorted signature Σ and an interpreting Σ-algebra A that we use throughout this chapter. We define spatial objects and show how the operations on A can be pointwise lifted to form a Σ-algebra over the set of all spatial objects.

4.2.1 Spatial Objects

Let Σ be a general many sorted signature with sorts $\ldots, s, \ldots$ displayed as follows.

Signature	Σ
Sorts	$\ldots, s, \ldots$
Constants	$\ldots, c : \to s, \ldots$
Operations	$\ldots, f : s_1 \times \cdots \times s_n \to s, \ldots$

Consider a general Σ-algebra A which interprets Σ:

Algebra	A
Carriers	$\dots, A_s, \dots$
Constants	$\dots, c_A :\rightarrow A_s, \dots$
Operations	$\dots, f_A : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s, \dots$

Using this Σ-algebra, we define spatial objects as found in [55]:

Definition 4.2.1 (Spatial Objects) *Let X be a non-empty set representing a space and A a Σ-algebra. A spatial object is a total mapping*

$$\phi : X \rightarrow A,$$

which assigns data in A to points in X.

Let

$$F(X, A) = \{\phi : X \rightarrow A\}$$

be the set of all total mappings from the set X to the data in the algebra A; in this case, the data will be tuples $A_{s_1} \times \cdots \times A_{s_n}$ of the carriers of A. We will often use the terms "spatial object" and "function" interchangeably throughout our work.

Examples

We give several examples that motivate the general theory chosen to highlight the wide applicability of spatial objects in Computer Science, particularly in Constructive Volume Geometry and the theory of machine modelling.

Example 4.2.1 (Constructive Volume Geometry)
CVG defines objects in three dimensional space where $X = \mathbb{R}^3$. The data is usually from a Σ-algebra S which contains a subset of the real numbers as the carrier, and contains operations on this data:

Algebra	S
Carriers	$S = [0,1] \subset \mathbb{R}$
Import	$\mathbb{R}$
Constants	$0, 1 \in \mathbb{R}$
Operations	$\max : S \times S \to S$ $\min : S \times S \to S$ $+ : S \times S \to S$ $- : S \times S \to S$ $\cdot : S \times S \to S$
Definitions	$\max(s_1, s_2) = maximum(s_1, s_2)$ $\min(s_1, s_2) = minimum(s_1, s_2)$ $s_1 + s_2 = minimum(1, s_1 +_{\mathbb{R}} s_2)$ $s_1 - s_2 = maximum(0, s_1 -_{\mathbb{R}} s_2)$ $s_1 \cdot s_2 = minimum(1, s_1 \cdot_{\mathbb{R}} s_2)$

Then the set of CVG spatial objects mapping points in $\mathbb{R}^3$ to $[0,1]$ is denoted

$$F(\mathbb{R}^3, S).$$

Closely related to the CVG example are spatial objects in discrete space:

Example 4.2.2 (2D Grids) Consider an example where the space X is the two dimensional discrete grid of integers $\mathbb{Z}^2$, and the data is contained in some Σ-algebra A. Then

$$F(\mathbb{Z}^2, A)$$

is the set of spatial objects that assigns the data in A to each (x, y) coordinate in the space $\mathbb{Z}^2$.

Example 4.2.3 (Finite 2D Grids) In this example the space X is the two dimensional discrete grid of finite integers $\mathbb{Z}_n \times \mathbb{Z}_n$ where $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$.

Example 4.2.4 (Machine Modelling) The memory addresses of a machine can be thought of as a space. We define the space $X = Addr$ to be the set of all memory location addresses in a machine. Let the data be from a Σ-algebra A. Then the spatial object $\phi : Addr \to A$ represents a particular state of the memory in the machine and the set $F(Addr, A)$ contains all possible memory states.

Now consider the case when $A = W_k$, where W_k is the Σ-algebra defined as

Algebra	W_k
Carriers	$W_k = \{0,1\}^k$
Import	$\mathbb{B}$
Constants	$0, 1 :\rightarrow W_k$
Operations	$\wedge, \vee : W_k \times W_k \rightarrow W_k$ $\neg : W_k \rightarrow W_k$ $+1 : W_k \rightarrow W_k$
Definitions	$(b_1, \ldots, b_k) \wedge (b'_1, \ldots, b'_k) = (b_1 \wedge b'_1, \ldots, b_1 \wedge b'_k)$ $(b_1, \ldots, b_k) \vee (b'_1, \ldots, b'_k) = (b_1 \vee b'_1, \ldots, b_1 \vee b'_k)$ $\neg(b_1, \ldots, b_k) = (\neg b_1, \ldots, \neg b_k)$ $(b_1, \ldots, b_k) + 1 = (b_1 + 1, \ldots, b_k + 1)$

which contains k-bit words. We define constants and operations which specify the standard boolean operations of *and*, *or* and *negation* and a "successor" function on the binary numbers. We note the set W_k is cyclic; that is,

$$W_k = \mathbb{Z}_{2^k} = \{0, 1, \ldots, 2^k - 1\}.$$

Thus an overflow error from the application of the successor function is handled simply by returning zero. The set $F(Addr, W_k)$ contains all possible configurations of the address space in the machine. We can set the address space $Addr$ to the natural numbers $\mathbb{N}$ to make $F(\mathbb{N}, W_k)$ the set of all spatial objects mapping a natural number to a k-bit word.

Setting the space $Addr = W_l$ gives us all spatial objects $F(W_l, W_k)$ assigning k-bit words to addresses of length l.

Example 4.2.5 (The Ring of Real Numbers) Let $X = \mathbb{R}$ and define the Σ-algebra R as

Algebra	R
Carriers	$\mathbb{R}$
Constants	$0_R, 1_R :\rightarrow \mathbb{R}$
Operations	$+_R, \cdot_R : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ $-_R : \mathbb{R} \rightarrow \mathbb{R}$

with the operations $+_R$,$-_R$ and $\cdot_R$ defined on $\mathbb{R}$ as usual. Then $F(\mathbb{R}, R)$ denotes the set of all total functions $f : \mathbb{R} \rightarrow R$ mapping points in $\mathbb{R}$ to points in the ring of reals R.

Example 4.2.6 (Sequences) Let $X = \mathbb{N}$ and define the Σ-algebra N

Algebra	N
Carriers	$\mathbb{N}$
Constants	$0_N :\rightarrow \mathbb{N}$
Operations	$+1 : \mathbb{N} \rightarrow \mathbb{N}$ $+ : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ $\times : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

to be the Peano-arithmetic algebra. We can define $F(\mathbb{N}, N)$ to be the set of all sequences of natural numbers. For example, $e \in F(\mathbb{N}, N)$ defined by

$$e(0) = 0,\ e(1) = 2,\ e(2) = 4, \ldots$$

denotes the sequence of the even natural numbers. We can write this sequence in a more traditional notation as

$$(e_i)_{i=0}^{\infty} = (0, 2, 4, \ldots).$$

Example 4.2.7 (Products) Now, consider the product of algebras

$$A^X = \prod_{x \in X} A_x,$$

indexed by an arbitrary set X, where there is a Σ-algebra for each coordinate of the product A^X. In the standard construction of the product, a typical element of A^X has the form

$$(\ldots, a_x, \ldots)$$

where a_x (at the x-coordinate) is an element of the Σ-algebra A_x. In the special case that each $A_x = A$ for all $x \in X$, we actually have the collection of all total functions mapping the set X to A. That is,

$$A^X = \prod\{A : x \in X\} = \{\phi : \phi \text{ is a function on } X \text{ to } A\},$$

which is precisely the definition of $F(X, A)$.

We refer the reader to [43](p.271) where a detailed construction of a Σ-algebra over the direct product $\prod A$ and a formal proof shows that $\prod A = F(X, A)$.

4.2.2 Pointwise Data Operations on Spatial Objects

For this section, let Σ be a general single sorted signature with sort s and A a Σ-algebra. New operations can be constructed on $F(X, A)$ from the Σ-algebra A by way of pointwise lifting. This makes $F(X, A)$ into a Σ-algebra as follows.

We specify a constant function such that for each constant symbol $c \in \Sigma_s^\lambda$ the function

$$c_{F(X,A)} : X \to A \in F(X, A)$$

is defined by

$$c_{F(X,A)}(x) = c_A, \text{ for all } x \in X.$$

For each function symbol $f \in \Sigma_s^{s^n}$ with $f : s^n \to s$, and $\phi_i \in F(X, A)$ for $1 \leq i \leq n$, we define the operation $f_{F(X,A)} : F(X, A)^n \to F(X, A)$ pointwise as

$$f_{F(X,A)}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$$

for all x in X.

These constants and operations on $F(X, A)$ produce a Σ-algebra F which we display as:

Algebra	$F(X,A)$
Carriers	$F(X,A)$
Constants	$\ldots, c_{F(X,A)} : X \to A, \ldots$
Operations	$\ldots, f_{F(X,A)} : F(X,A)^n \to F(X,A), \ldots$
Definitions	$c_{F(X,A)}(x) = c_A$ $f_{F(X,A)}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$

One interpretation of the pointwise operator $f_{F(X,A)}$ is the evaluation in parallel, or the simultaneous application, of the operation f_A on all points x in the space X.

Notice that this algebra $F(X,A)$ is very abstract since one can operate *only* on spatial objects – operations on both space and data are out of bounds! In the next section some operators are collected that allow access to the data of the spatial objects we define. We address the formal status of the definition above in Section 4.2.5. For now, we collect several more examples of pointwise lifted operations.

Examples

Example 4.2.8 (Constructive Volume Geometry, Continued)
From Chapter 2, Section 2.4.1 we have the following algebra from which the operations are lifted pointwise from those in the scalar field $\phi : E^3 \to S$.

Algebra	SF
Carriers	$F(E^3, S)$
Import	S
Constants	$\mathbf{0}, \mathbf{1} :\rightarrow F(E^3, S)$
Operations	$Max : F(E^3,S) \times F(E^3,S) \rightarrow F(E^3,S)$
	$Min : F(E^3,S) \times F(E^3,S) \rightarrow F(E^3,S)$
	$+ : F(E^3,S) \times F(E^3,S) \rightarrow F(E^3,S)$
	$- : F(E^3,S) \times F(E^3,S) \rightarrow F(E^3,S)$
	$\cdot : F(E^3,S) \times F(E^3,S) \rightarrow F(E^3,S)$
Definitions	$Max(\phi_1, \phi_2)(x) = \max(\phi_1(x), \phi_2(x))$
	$Min(\phi_1, \phi_2)(x) = \min(\phi_1(x), \phi_2(x))$
	$(\phi_1 + \phi_2)(x) = \phi_1(x) + \phi_2(x)$
	$(\phi_1 - \phi_2)(x) = \phi_1(x) - \phi_2(x)$
	$(\phi_1 \cdot \phi_2)(x) = \phi_1(x) \cdot \phi_2(x)$

Example 4.2.9 (Machine Modelling, Continued)
Consider the situation where $Addr = \mathbb{N}$ and the set of spatial objects are $F(\mathbb{N}, \{0,1\}^k)$; all spatial objects $\phi : \mathbb{N} \rightarrow \{0,1\}^k$ mapping the natural numbers to k-bit words representing the values 0 to $2^k - 1$.

Hence each ϕ in $F(\mathbb{N}, \{0,1\}^k)$ represents a particular state in a machine, whose memory addresses are indexed by the natural numbers.

We can define the logical operations $\wedge$, $\vee$ and $\neg$ over $F(\mathbb{N}, \{0,1\}^k)$ by performing the operations on the valued stored at each of the memory addresses at two machine states ϕ and ϕ'. For all $n \in \mathbb{N}$ we define

$$\begin{aligned} (\phi \wedge \phi')(n) &= \phi(n) \wedge \phi'(n) \\ (\phi \vee \phi')(n) &= \phi(n) \vee \phi'(n) \\ (\neg\phi)(k) &= \neg(\phi(n)) \end{aligned}$$

We define a successor function for a state ϕ in $F(\mathbb{N}, \{0,1\}^k)$ by adding $+1$ to the value stored at each memory address. For all $n \in \mathbb{N}$ we have

$$\phi + \mathbf{1} = \phi(n) + 1,$$

where $\mathbf{1} : \mathbb{N} \rightarrow \{0,1\}^k$ is the state which has the value 1 stored at every memory location. This operation on the spatial object (or machine state) has the effect of adding 1 to the data at each address space.

Example 4.2.10 (Ring of Real Numbers, Continued)
Recalling the ring $R = \langle \mathbb{R}; 0_R, 1_R; -_R, +_R, \cdot_R \rangle$ of real numbers, we pointwise lift the constants $0, 1$ in R to define the constants $\mathbf{0}, \mathbf{1} \in F(\mathbb{R}, R)$ as

$$\begin{aligned} \mathbf{0}(x) &= 0_R \ \forall x \in X, \\ \mathbf{1}(x) &= 1_R \ \forall x \in X. \end{aligned}$$

The operations $+_R$, $-_R$ and $\cdot_R$ in R are pointwise lifted to define

$$\begin{aligned} (f + g)(x) &= f(x) +_R g(x) \\ (-f)(x) &= -_R(f(x)) \\ (f \times g)(x) &= f(x) \cdot_R g(x) \end{aligned}$$

producing an algebra on $F(\mathbb{R}, R)$.

Example 4.2.11 (Sequences, Continued)
Recalling our example where $F(\mathbb{N}, N)$ represented the set of all spatial objects assigning data in the Peano algebra $N = \langle \mathbb{N}; 0; n + 1, n + m, n \times m \rangle$ to points in the space $X = \mathbb{N}$, we can lift the Peano-operations of N to operations on sequences in $F(\mathbb{N}, N)$ as follows: For $f, g \in F(\mathbb{N}, N)$ we define

$$\begin{aligned} (f + 1)(x) &= f(x) + 1, \\ (f + g)(x) &= f(x) + g(x), \\ (f \times g)(x) &= f(x) \times g(x) \end{aligned}$$

In traditional sequence notation, these operators are written as

$$((a_i)_{i=0}^{\infty} + 1) = (a_i + 1)_{i=0}^{\infty}$$

and

$$(a_i)_{i=0}^{\infty} + (b_i)_{i=0}^{\infty} = (a_i + b_i)_{i=0}^{\infty}$$

4.2.3 Pointwise Space Operations on Spatial Objects

We have seen many examples in the previous section showing how one may take operations on the data in the carriers of the Σ-algebra A and lift them to operations on the spatial objects in $F(X, A)$. We now study how one may pointwise lift operations on space.

Let X be a space and A some arbitrary Σ-algebra. Let

$$\beta : X \to X$$

be some operation on the space X. We define an operation

$$\overline{\beta} : F(X, A) \to F(X, A)$$

such that for any spatial object $\phi \in F(X, A)$ and point x in the space X

$$\overline{\beta}(\phi)(x) = \phi(\beta(x)).$$

Example 4.2.12 (2D Grids, Continued) Recall Example 4.2.2 where we defined the space $X = \mathbb{Z}^2$, the 2D integer grid which is mapped to data in a Σ-algebra A.

Now suppose we have define an operation working on space $+k : \mathbb{Z}^2 \to \mathbb{Z}^2$ such that

$$(x, y) + k = (x + k, y + k)$$

for a fixed number k in $\mathbb{Z}$.

We define an operation $+_F k : F(\mathbb{Z}^2, A) \to F(\mathbb{Z}^2, A)$ for a spatial object $\phi \in F(\mathbb{Z}^2, A)$ and a point $(x, y) \in \mathbb{Z}^2$ as

$$\begin{aligned}(+_F k)(\phi)(x, y) &= \phi((x, y) + k) \\ &= \phi(x + k, y + k)\end{aligned}$$

Similarly, we define another operation $\cdot k : \mathbb{Z}^2 \to \mathbb{Z}^2$ as $(x, y) \cdot k = (x \cdot k, y \cdot k)$ and lift this to the operation $\cdot_F k : F(\mathbb{Z}^2, A) \to F(\mathbb{Z}^2, A)$ as

$$\begin{aligned}(\cdot_F k)(\phi)(x, y) &= \phi((x, y) \cdot k) \\ &= \phi(x \cdot k, y \cdot k).\end{aligned}$$

Example 4.2.13 (Matrix Transformations)
Rotation is an important operation in any computer graphics application. Standard textbooks on graphics such as [6] show how matrix transformations define rotation of a point about the origin by an angle θ. We extend this to show how we can rotate spatial objects in two-dimensions.

Let X be the two dimensional Euclidean space $\mathbb{E}^2$. To calculate a counter-clockwise rotation by an angle θ about the origin we use the matrix transformation $\beta_\theta : X \to X$ defined by

$$\beta_\theta(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}.$$

We can pointwise lift this to operations on $F(X, A)$ as follows: define $\overline{\beta}_\theta : F(X, A) \to F(X, A)$ as

$$\begin{aligned}\overline{\beta}_\theta(\phi)(x, y) &= \phi(\beta_\theta((x, y))) \\ &= \phi((\cos\theta \cdot x - \sin\theta \cdot y, \sin\theta \cdot x + \cos\theta \cdot y)).\end{aligned}$$

4.2.4 Covariance and Contravariance

In this section, we study general transformations on data and space: *covariant* and *contravarient* functions between algebras of spatial objects.

Covariant functions modify data in spatial data types. For example, if we have a three dimensional object represented in space, we may wish to change attributes such as colour or opacity. Any data attributes associated with the object can be modified by means of a covariant function.

Continuing with this example, one can imagine contravariant transformations modifying the shape of an object in three dimensional space. We could define a contravariant function to rotate an object about the x axis, or to scale it by a factor of 2.

We have the following definitions:

Definition 4.2.2 (Covariant Functions) *Let $F(X,A)$ and $F(X,B)$ be two functions spaces where A and B are Σ-algebras. For the mapping $\alpha : A \to B$ we define $\overline{\alpha} : F(X,A) \to F(X,B)$ as*

$$\begin{aligned}\overline{\alpha}(\phi)(x) &= (\alpha \circ \phi)(x) \\ &= \alpha(\phi(x))\end{aligned}$$

for a given point $x \in X$ and $\phi \in F(X,A)$.

Covariant functions translate *data*. Alternatively, we may translate the *space* by contravariant functions:

Definition 4.2.3 (Contravariant Functions) *Let $F(X,A)$ and $F(Y,A)$ be two functions spaces where X and Y are sets and A is a Σ-algebra. For the mapping $\beta : Y \to X$ we define the function $\overline{\beta} : F(Y,A) \to F(X,A)$ as*

$$\begin{aligned}\overline{\beta}(\phi)(y) &= (\phi \circ \beta)(y) \\ &= \phi(\beta(y))\end{aligned}$$

for a given point $y \in Y$, and $\phi \in F(X,A)$.

4.2.5 The Evaluation and Substitution Operators

So far we have seen how pointwise lifting can be used to construct operations on spatial objects. We now discuss some useful operations on spatial objects that allow *access* to the actual data stored in space as a means of retrieving and modifying it.

To this end, we expand the signature Σ by adding the operators *eval* and *sub* [43]. Generally speaking, the *eval* operator is used to evaluate a spatial object $\phi \in F(X,A)$ on a point $x \in X$. We have

$$eval : F(X,A) \times X \to A$$

defined as

$$eval(\phi, x) = \phi(x),$$

which returns a data value in the algebra A.

We may also wish to update a spatial object so that it returns a new data element at a specific point. The *sub* operator takes a point $x \in X$, a data value $a \in A$ and spatial object $\phi \in F(X, A)$ and sets $\phi(x)$ to the value a. We have

$$sub : F(X, A) \times X \times A \to F(X, A)$$

defined as

$$sub(\phi, x, a)(y) = \begin{cases} a & \text{if } x = y, \\ \phi(y) & \text{otherwise.} \end{cases}$$

for a point $y \in X$.

We construct a new signature Σ^+ using the sorts *so*, *sp* and *s*, the constants c_{so}, c_{sp} and c_s, and the original operations $f \in \Sigma_s^{s^n}$ along with the operations *eval* and *sub*.

For the interpreting algebra F^+, we add the carriers $F(X, A)$, X and A_s, the constants c_F, c_X and c_A and the functions $f_A : A_s^n \to A_s$ from the algebra A. The operators $eval_A$ and sub_A are also included.

We list this new many sorted signature Σ^+ with the general interpreting algebra F^+ as follows:

Signature	Σ^+
Sorts	sp, so, s
Constants	$\ldots, c_s :\to s, \ldots$ $\ldots, c_{so} :\to so, \ldots$
Operations	$\ldots, f : s \times \cdots \times s \to s, \ldots$ $\ldots, F : so^n \to so, \ldots$ $sub : so \times sp \times s \to so$ $eval : so \times sp \to s$

Algebra	F^+
Carriers	$F(X,A)$, X, A_s
Constants	$\ldots, c_A :\rightarrow A_s, \ldots$ $\ldots, c_{F^+} : X \rightarrow A_s, \ldots$ $\ldots, c_X :\rightarrow X, \ldots$
Operations	$\ldots, f_A : A_s \times \cdots \times A_s \rightarrow A_s, \ldots$ $\ldots, f_{F^+} : F(X,A)^n \rightarrow F(X,A), \ldots$ $sub_A : F(X,A) \times X \times A_s \rightarrow F(X,A)$ $eval_A : F(X,A) \times X \rightarrow A_s$
Definitions	$c_{F^+}(x) = c_A$ $f_{F^+}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$ $sub(\phi, x, a)(y) = \begin{cases} a & \text{if } x = y, \\ \phi(y) & \text{otherwise.} \end{cases}$ $eval(\phi, x) = \phi(x)$

As a technical point of the status of equations in this thesis, we make the note that pointwise lifting is formally defined as an equation

$$eval(F(\phi_1, \ldots, \phi_n, x)) = f(eval(\phi_1, x), \ldots, eval(\phi_n, x))$$

over the signature Σ^+.

4.2.6 Subalgebras of $F(X,A)$

We prove two lemmas that give useful results on the relationship between the data in A and the spatial objects in $F(X,A)$. We first show that pointwise lifting preserves the subalgebra property.

Lemma 4.2.1 (Pointwise Lifting of a Σ-subalgebra)
Let A be a Σ-algebra. If $B \subseteq A$ is a Σ-subalgebra then $F(X,B) \subseteq F(X,A)$ is a Σ-subalgebra of $F(X,A)$.

Proof Let $\phi_1, \ldots, \phi_n$ be functions in $F(X,B)$. The operation

$$f_{F(X,A)}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$$

is a typical pointwise lifted operation of $F(X,A)$ (constants are lifted in the same manner). For all $x \in X$ we have

$$\begin{aligned} f_{F(X,A)}(\phi_1,\ldots,\phi_n)(x) &= f_A(\phi_1(x),\ldots,\phi_n(x)) \\ &= f_B(\phi_1(x),\ldots,\phi_n(x)) \text{ since } B \text{ is a } \Sigma\text{-subalgebra} \\ &= f_{F(X,B)}(\phi_1,\ldots,\phi_n)(x) \end{aligned}$$

satisfying the conditions for $F(X,B)$ to be a Σ-subalgebra of $F(X,A)$. ■

Lemma 4.2.2 *Suppose $S(X,A)$ is a Σ-subalgebra of $F(X,A)$. Define $B_x = \{\phi(x) \mid \phi \in S(X,A)\}$ for x in X. Then B_x is a Σ-subalgebra of A.*

Proof Assume $S(X,A) \leq F(X,A)$. Pick an $x \in X$ and define $B_x = \{\phi(x) \mid \phi \in S(X,A)\}$. Now, pick values $b_1,\ldots,b_n \in B_x$. Then there exists $\phi_1,\ldots,\phi_n \in S(X,A)$ such that $\phi_i(x) = b_i$. We have

$$\begin{aligned} f_A(b_1,\ldots,b_n) &= f_A(\phi_1(x),\ldots,\phi_n(x)) \\ &= f_{F(X,A)}(\phi_1,\ldots,\phi_n)(x) \end{aligned}$$

By our assumption we have the equation

$$\begin{aligned} f_{F(X,A)}(\phi_1,\ldots,\phi_n)(x) &= f_{S(X,A)}(\phi_1,\ldots,\phi_n)(x) \\ &= \phi' \in S(X,A) \end{aligned}$$

But $\phi' \in S(X,A)$ and $\phi'(x) = f_A(b_1,\ldots,b_n) \in B_x$ completing the proof. ■

4.3 Terms

This section describes terms formed from the signature Σ and variables in the set Y. For simplicity, these terms are *single sorted* with the sort s.

We shall define evaluation of these terms over the Σ-algebras $F(X,A)$ and A and prove

- the Validity Theorem 4.3.1 which shows equational validity is equivalent on both A and $F(X,A)$ for equations of the form $t = t'$,

- the Validity Theorem 4.3.2 which shows the validity for conditional equations of the form $t_1 = t_1' \Rightarrow t = t'$ are equivalent on both A and $F(X,A)$, and

- that universal formulae which are true on A do not necessarily hold on $F(X,A)$.

4.3.1 Term Evaluation

Roughly speaking, term evaluation over a Σ-algebra A is a mapping from the set of terms $T(\Sigma, Y)$ to A that works out the value of a term in A. The aim of this section is to define term evaluation over Σ-algebras A and $F(X, A)$ and show their equivalence.

We define the collection of *assignment functions* that map variables in Y to data in A and $F(X, A)$ as

$$\begin{aligned} V(Y, A) &= \{v : Y \to A\}, \\ V(Y, F(X, A)) &= \{\overline{v} : Y \to F(X, A)\} \end{aligned}$$

respectively. We shall use these assignment functions to define two functions mapping terms in $T(\Sigma, Y)$ to data elements in algebras A and $F(X, A)$.

Now, we have already seen the definition of term evaluation in Section 3.3 for the many sorted case, however it is worthwhile to reproduce it here. This definition for the single sorted Σ-algebra A is a modification of the one found in [55] and is as follows:

Definition 4.3.1 (Term Evaluation) *Given an assignment $v : Y \to A$ of an element $v(y_i) \in A$ to a variable $y_i \in Y$, we define the term evaluation map*

$$[\![-]\!]_A^v : T(\Sigma, Y) \times V(Y, A) \to A$$

by induction on the structure of terms: for all constants $c \in \Sigma_s^\lambda$, variables $y_i \in Y$, operations in $f \in \Sigma_s^{s^n}$ and terms $t_1, \ldots, t_n$ in $T(\Sigma, Y)$ we have

$$\begin{aligned} [\![c]\!]_A^v &= c_A, \\ [\![y_i]\!]_A^v &= v(y_i), \\ [\![f(t_1, \ldots, t_n)]\!]_A^v &= f_A([\![t_1]\!]_A^v, \ldots, [\![t_n]\!]_A^v). \end{aligned}$$

The notation we use in this definition is described in [55] as

$[\![t]\!]_A^v =$ the evaluation of t over A with values of the variables given by v

In lieu of an assignment function, we may also supply the values for the finite set of variables that a term t consists of by means of an n-tuple. For example, we write

$$t(a_1, \ldots, a_n)$$

to mean that the variables in the term $t \in T(\Sigma, Y)$ are given the values

$$y_1 := a_1, \ldots, y_n := a_n,$$

for a_i in A.

We define the term evaluation map

$$[\![-]\!]_F^{\overline{v}} : T(\Sigma, Y) \times V(Y, F(X, A)) \to F(X, A)$$

for the Σ-algebra $F(X,A)$ inductively with the assignment function $\overline{v} : Y \to F(X,A)$ by the following cases:
Base Cases:

$$\begin{aligned}
[\![c]\!]_F^{\overline{v}} &= c_F \\
[\![y_i]\!]_F^{\overline{v}} &= \overline{v}(y_i)
\end{aligned}$$

Structural Induction

$$[\![f(t_1,\ldots,t_n)]\!]_F^{\overline{v}} = f_F([\![t_1]\!]_F^{\overline{v}},\ldots,[\![t_n]\!]_F^{\overline{v}}).$$

We use the notation $[\![t]\!]_F^{\overline{v}}$ to mean the evaluation of t over $F(X,A)$ with values of the variables given by $\overline{v}$, and write

$$[\![t]\!]_F(\phi_1,\ldots,\phi_n)$$

to mean the evaluation over $F(X,A)$ of a term t using the values $\phi_i \in F(X,A)$ with $1 \leq i \leq n$, rather than denoting a specific assignment function.

4.3.2 Pointwise Lifting of Term Equivalence

We want to show the equivalence of term evaluation on algebras $F(X,A)$ and A.

Lemma 4.3.1 (The Pointwise Lifting of Term Equivalence)
For any term t in $T(\Sigma,Y)$ and spatial objects $\phi_1,\ldots,\phi_n$ in $F(X,A)$, the equation

$$[\![t]\!]_F(\phi_1,\ldots,\phi_n)(x) = [\![t]\!]_A(\phi_1(x),\ldots,\phi_n(x))$$

holds for all $x \in X$.

Proof We prove the result using structural induction on terms.
Base Cases: For each constant c in Σ_s^λ, we have

$$\begin{aligned}
[\![c]\!]_F(\phi_1,\ldots,\phi_n)(x) &= c_F(x) \\
&= c_A \\
&= [\![c]\!]_A(\phi_1(x),\ldots,\phi_n(x)).
\end{aligned}$$

For each $y_i \in Y$ and $1 \leq i \leq n$, we have

$$\begin{aligned}
[\![y_i]\!]_F(\phi_1,\ldots,\phi_n)(x) &= \phi_i(x) \\
&= [\![y_i]\!]_A(\phi_1(x),\ldots,\phi_n(x)).
\end{aligned}$$

Inductive Step: Let $t = f(t_1,\ldots,t_n)$ where $f \in \Sigma_{s^n,s}$ and $t_1,\ldots,t_n$ are terms. We have

$$[\![f(t_1,\ldots,t_n)]\!]_F(\phi_1,\ldots,\phi_n)(x) = f_F([\![t_1]\!]_F(\phi_1,\ldots,\phi_n),\ldots,[\![t_n]\!]_F(\phi_1,\ldots,\phi_n))(x)$$

by the definition of term evaluation on $F(X,A)$. From the pointwise definition of f_F this is equal to

$$= f_A([\![t_1]\!]_F(\phi_1,\ldots,\phi_n)(x),\ldots,[\![t_n]\!]_F(\phi_1,\ldots,\phi_n)(x))$$

and by the induction hypothesis on the t_i we have

$$\begin{aligned} &= f_A([\![t_1]\!]_A(\phi_1(x),\ldots,\phi_n(x)),\ldots,[\![t_n]\!]_A(\phi_1(x),\ldots,\phi_n(x)) \\ &= [\![f(t_1,\ldots,t_n)]\!]_A(\phi_1(x),\ldots,\phi_n(x)) \end{aligned}$$

completing the proof. ■

4.3.3 Equational Validation

Using the pointwise lifting results of term equivalence in the previous section, we prove a useful result proving equational validity also lifts. To begin, we define the notion of satisfiability of equations as found in [43]:

Definition 4.3.2 (Equational Validity) *Let A be a Σ-algebra. The equation*

$$t = t'$$

of terms in $T(\Sigma, Y)$ is satisfied by, or is valid in A if

$$\forall(a_1,\ldots,a_n)\big[[\![t]\!]_A(a_1,\ldots,a_n) = [\![t']\!]_A(a_1,\ldots,a_n)\big],$$

for $a_i \in A$ with $1 \le i \le n$.

Equational validity of an equation $t = t'$ over the algebra A will be denoted by

$$A \models t = t',$$

and we use this definition in the obvious way for the algebra $F(X,A)$: the equation $t = t'$ is valid in $F(X,A)$ if

$$\forall(\phi_1,\ldots,\phi_n)\big[[\![t]\!]_F(\phi_1,\ldots,\phi_n) = [\![t']\!]_F(\phi_1,\ldots,\phi_n)\big]$$

for $\phi_i \in F(X,A)$ with $1 \le i \le n$.

Theorem 4.3.1 (Validity Theorem) *For any Σ-equation $t = t'$ the following are equivalent:*

(1) $t = t'$ is valid in A; i.e. $A \models t = t'$.

(2) $t = t'$ is valid in $F(X,A)$; i.e. $F(X,A) \models t = t'$.

In symbols,

$$A \models t = t' \iff F(X,A) \models t = t'.$$

PROOF (1) $\Rightarrow$ (2). Assume that $A \models t = t'$. Choose any $\phi_1, \dots, \phi_n$ in $F(X, A)$. For a point $x \in X$, let $a_1 := \phi_1(x), \dots, a_n := \phi_n(x)$. We have

$$\begin{aligned}
[\![t]\!]_F(\phi_1, \dots, \phi_n)(x) &= [\![t]\!]_A(\phi_1(x), \dots, \phi_n(x)) \text{ by Lemma 4.3.1} \\
&= [\![t]\!]_A(a_1, \dots, a_n) \\
&= [\![t']\!]_A(a_1, \dots, a_n) \text{ by Hypothesis (1)} \\
&= [\![t']\!]_A(\phi_1(x), \dots, \phi_n(x)) \\
&= [\![t']\!]_F(\phi_1, \dots, \phi_n)(x) \text{ by Lemma 4.3.1.}
\end{aligned}$$

(1) $\Leftarrow$ (2). Conversely, we now assume that $F(X, A) \models t = t'$ is true. Choose any $a_1, \dots, a_n$ in A. We construct the *constant* functions

$$\phi_1(x) = a_1, \dots, \phi_n(x) = a_n,$$

for all $x \in X$. Clearly these functions must be in the set $F(X, A)$ because each are total mappings from X to A. Now,

$$\begin{aligned}
[\![t]\!]_A(a_1, \dots, a_n) &= [\![t]\!]_A(\phi_1(x), \dots, \phi_n(x)) \text{ by construction} \\
&= [\![t]\!]_F(\phi_1, \dots, \phi_n)(x) \text{ by Lemma 4.3.1} \\
&= [\![t']\!]_F(\phi_1, \dots, \phi_n)(x) \text{ by Hypothesis (2)} \\
&= [\![t']\!]_A(\phi_1(x), \dots, \phi_n(x)) \text{ by Lemma 4.3.1} \\
&= [\![t']\!]_A(a_1, \dots, a_n) \text{ by construction.}
\end{aligned}$$

■

The Validity Theorem is actually a result of the fact that $A^X = F(X, A)$, shown in [43](p.278). We have spelt out the proof here in detail as we use it extensively in our work with data types. The next corollary uses the theorem to show that if an implementation A of the signature Σ satisfies a set of equational laws, then the pointwise lifted implementation $F(X, A)$ of Σ also satisfies these equational laws.

Corollary 4.3.1 *Let A satisfy a set E of Σ-equations. Then $F(X, A)$ also satisfies E.*

PROOF Let $t = t'$ be a typical Σ-equation in E satisfied by A. Then by the Validity Theorem 4.3.1

$$A \models t = t' \iff F(X, A) \models t = t'$$

proving the result. ■

We will see this corollary used in Sections 4.3.4 and 4.4 and the limitations of the Validity Theorem will also be shown with a counterexample.
We now turn to *conditional equations*: Let $t_1 = t_1', \dots t_k = t_k'$ be equations of terms in $T(\Sigma, Y)$. Equations of the form

$$t_1 = t_1' \wedge \dots \wedge t_k = t_k' \implies t = t'$$

are called conditional equations and are valid in A if

$$\begin{aligned}\forall(a_1,\ldots,a_n)\big[[\![t_1]\!]_A(a_1,\ldots,a_n) &= [\![t_1]\!]_A(a_1,\ldots,a_n)\\ &\vdots\\ [\![t_k]\!]_A(a_1,\ldots,a_n) &= [\![t_k]\!]_A(a_1,\ldots,a_n)\end{aligned}$$

implies

$$[\![t]\!]_A(a_1,\ldots,a_n) = [\![t]\!]_A(a_1,\ldots,a_n)\big].$$

Theorem 4.3.2 (Validity Theorem For Conditional Equations)
For any Σ conditional equation $t_1 = t_1' \Rightarrow t = t'$ the following are equivalent:

(1) $t_1 = t_1' \wedge \cdots \wedge t_k = t_k' \implies t = t'$ is valid in A,

(2) $t_1 = t_1' \wedge \cdots \wedge t_k = t_k' \implies t = t'$ is valid in $F(X, A)$.

That is,

$$\begin{aligned}& A \models t_1 = t_1' \wedge \cdots \wedge t_k = t_k' \implies t = t'\\ \iff\ & F(X,A) \models t_1 = t_1' \wedge \cdots \wedge t_k = t_k' \implies t = t'.\end{aligned}$$

Proof Without loss of generality, assume that $k = 1$.
(1) $\Rightarrow$ (2). Suppose that (1) holds and $F(X, A) \models t_1 = t_1'$. This means that

$$[\![t_1]\!]_F(\phi_1,\ldots,\phi_n) = [\![t_1']\!]_F(\phi_1,\ldots,\phi_n)$$

or equivalently, for all $x \in X$:

$$[\![t_1]\!]_F(\phi_1,\ldots,\phi_n)(x) = [\![t_1']\!]_F(\phi_1,\ldots,\phi_n)(x).$$

By Lemma 4.3.1,

$$\begin{aligned}[\![t_1]\!]_F(\phi_1,\ldots,\phi_n)(x) &= [\![t_1]\!]_A(\phi_1(x),\ldots,\phi_n(x))\\ [\![t_1']\!]_F(\phi_1,\ldots,\phi_n)(x) &= [\![t_1']\!]_A(\phi_1(x),\ldots,\phi_n(x))\end{aligned}$$

By the conditional equation law in A (assumption of (1)), we know

$$[\![t]\!]_A(\phi_1(x),\ldots,\phi_n(x)) = [\![t']\!]_A(\phi_1(x),\ldots,\phi_n(x))$$

and again by Lemma 4.3.1

$$[\![t]\!]_F(\phi_1,\ldots,\phi_n)(x) = [\![t']\!]_F(\phi_1,\ldots,\phi_n)(x).$$

Hence $t_1 = t_1' \Rightarrow F(X, A) \models t = t'$.

(1) $\Leftarrow$ (2). Now suppose that (2) holds and $t_1 = t_1'$ is valid on A. This means that for all $a_1, \ldots, a_n \in A$

$$[\![t_1]\!]_A(a_1,\ldots,a_n) = [\![t_1']\!](a_1,\ldots,a_n).$$

For each value a_i in A, $1 \leq i \leq n$, we construct the constant function ϕ_i such that for all $x \in X$, $\phi_i(x) = a_i$. By Lemma 4.3.1 the following equalities hold:

$$\begin{aligned} \llbracket t_1 \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)) &= \llbracket t_1 \rrbracket_F(\phi_1, \ldots, \phi_n)(x) \\ \llbracket t_1' \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)) &= \llbracket t_1' \rrbracket_F(\phi_1, \ldots, \phi_n)(x). \end{aligned}$$

But by the assumption that (2) is valid,

$$\llbracket t \rrbracket_F(\phi_1, \ldots, \phi_n)(x) = \llbracket t' \rrbracket_F(\phi_1, \ldots, \phi_n)(x)$$

and by Lemma 4.3.1

$$\begin{aligned} \llbracket t \rrbracket_F(\phi_1, \ldots, \phi_n)(x) &= \llbracket t \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)) \\ \llbracket t' \rrbracket_F(\phi_1, \ldots, \phi_n)(x) &= \llbracket t' \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)) \end{aligned}$$

Hence

$$\llbracket t \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)) = \llbracket t' \rrbracket_A(\phi_1(x), \ldots, \phi_n(x)).$$

That is, $A \models t = t'$. ■

4.3.4 Pointwise Lifting of Commutative Rings

We can apply the theoretical results of this section to the case where the Σ-algebra A is a *commutative ring.* Recall the set of equational axioms of a commutative ring from the Preliminaries, Section 3.4. We prove

Lemma 4.3.2 *If the Σ-algebra A is a commutative ring then the pointwise lifted Σ-algebra $F(X, A)$ is also a commutative ring for any space X.*

PROOF We define

Algebra	$F(X, A)$
Carriers	$F(X, A)$
Constants	$0, 1 :\rightarrow F(X, A)$
Operations	$+ : F(X, A) \times F(X, A) \rightarrow F(X, A)$ $\cdot : F(X, A) \times F(X, A) \rightarrow F(X, A)$ $- : F(X, A) \rightarrow F(X, A)$

where the constants and operations are pointwise lifted from A as usual. We show that $F(X, A) \in Alg(\Sigma_{CRing}, CRing)$. Since each of the axioms listed in $CRing$ are equations, Corollary 4.3.1 applies and setting $E = CRing$, we immediately conclude that $F(X, A)$ is a ring. ■

We have just shown that each of the axioms in the algebraic specification of the commutative ring A pointwise lift to $F(X, A)$, and in general this is true for all equational axioms by the Validity Theorem 4.3.1. Particularly, for any set X the sets

$$F(X, \mathbb{Z}),\ F(X, \mathbb{Z}_n),\ F(X, \mathbb{Q}),\ F(X, \mathbb{R}) \text{ and } F(X, \mathbb{C})$$

are all commutative rings. Furthermore, we have also shown that conditional equations lift to $F(X, A)$ by the conditional equation Validity Theorem 4.3.2.

In this section we show that universal formulae do not have this lifting property. Specifically, we add to the specification of A a *universal formula* and show through a concrete counterexample that $F(X, A)$ does not satisfy this new specification. To this end, we make the following definitions from [39]:

Definition 4.3.3 (Divisors of Zero) *Let A be a ring, and a an arbitrary element in A. If there exists a nonzero element $b \in A$ such that $ab = 0$ or there exists a nonzero element $c \in A$ such that $ca = 0$ then a is said to be a divisor of zero.*

Clearly $a = 0$ is a divisor of zero in any ring aside from the trivial case when a is the only element of the ring. We say that a is a *proper divisor of zero* if it is a *non-zero* divisor of zero. Thus we define

Definition 4.3.4 *A commutative ring with unit without proper zero divisors is called an integral domain.*

By this definition, for any elements a, b in the integral domain A the formula

$$a \cdot b = 0 \Rightarrow a = 0 \vee b = 0 \tag{4.1}$$

is true. This is an example of a universal formula and any ring that satisfies this condition is an integral domain. We add Formula (4.1) to the list of axioms in $CRing$ to make the commutative ring A an integral domain. We aim to show that the pointwise lifting of the ring operations of A does not necessarily produce an integral domain on $F(X, A)$.

Observe that for $F(X, A)$ to be an integral domain, we would need the property that for any elements f, g in $F(X, A)$ the formula

$$f \cdot g = \mathbf{0} \Rightarrow f = \mathbf{0} \vee g = \mathbf{0},$$

holds, where $\mathbf{0} \in F(X, A)$ is the additive identity element of $F(X, A)$ (the zero function). We show that this is not the case by a counterexample: let $X = \mathbb{Z}$ and $A = \mathbb{Z}$.

Assume: $f \cdot g = \mathbf{0}$. By the definition of pointwise lifting, this means that for all x in $\mathbb{Z}$

$$(f \cdot g)(x) = f(x) \cdot g(x) = 0.$$

We show that this does not imply $f = \mathbf{0} \vee g = \mathbf{0}$. Define two functions

$$f(x) = \begin{cases} 0 & \text{if } x \geq 0, \\ 1 & \text{if } x < 0. \end{cases}$$

and

$$g(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

We have the situation $f(x) \cdot g(x) = 0$ for all x in $\mathbb{Z}$ but neither f or g is the zero function $\mathbf{0}$. That is,

$$f \cdot g = \mathbf{0} \not\Rightarrow f = \mathbf{0} \vee g = \mathbf{0}$$

and thus $F(X, A)$ is not an integral domain. Therefore, by this counterexample, universal formulae that are true on A are not necessarily true on $F(X, A)$.

4.3.5 Term Definable Functions

Suppose we are given a signature Σ and an interpreting Σ-algebra A. We can build a new signature Σ' whose interpreting algebra A' implements function symbols by evaluating terms in $T(\Sigma, Y)$ over the Σ-algebra A.

We begin by defining precisely what it means for a function to be term definable.

Definition 4.3.5 (Term Definable Operations) *Let Σ be a signature and A a Σ-algebra. We say that a function $\phi : A^n \longrightarrow A$ is Σ-term definable if there exists a term $u(y_1, , \ldots, y_n) \in T(\Sigma, Y)$ such that for all $a_1, \ldots, a_n \in A$*

$$\phi(a_1, \ldots, a_n) = [\![u(y_1, \ldots, y_n)]\!]_A(a_1, \ldots, a_n).$$

Note that by this definition any Σ-operation implemented by A is trivially Σ-term definable.

For some subset $H \subset A_s$ of the carrier set of A, we define the subalgebra $\langle H \rangle_\Sigma$ generated by the repeat applications of the Σ-operations on the subset H. Starting with the elements of H, we construct the set by recursion on the terms in $T(\Sigma, Y)$:

Base Cases:

$$[\![y]\!]_A(v_i) \in \langle H \rangle_\Sigma$$

for the variable $y \in T(\Sigma, Y)$ and a value $v_i \in H$,

$$[\![c]\!]_A \in \langle H \rangle_\Sigma$$

for the Σ-constant symbol $c :\longrightarrow s$,

Inductive Case:

$$[\![f(t_1,\ldots,t_n)]\!]_A(h_1,\ldots,h_{n_i}) \in \langle H\rangle_\Sigma$$

for Σ-operation $f : s^n \to s$, terms $t_1,\ldots,t_n \in T(\Sigma,Y)$ and $h_1,\ldots,h_n \in H$.

Now consider another signature Σ' for a new algebra A' which we build from the Σ-operations of A:

Signature	Σ'
Sorts	s
Constants	$k :\to s$
Operations	$g^1 : s^{n_1} \to s,\ldots,g^m : s^{n_m} \to s$

The interpreting Σ'-algebra A' is defined such that each function symbol $g^i : s^{n_i} \to s$ for $1 \le i \le m$ is defined using terms in $T(\Sigma,Y)$. That is, for any input values $a_1,\ldots,a_{n_i}$ in A, we calculate the value $g^i_{A'}(a_1,\ldots,a_{n_i})$ by evaluating $u_i(y_1,\ldots,y_{n_i})$ over the Σ-algebra A. In symbols,

$$g^i_{A'}(a_1,\ldots,a_{n_i}) = [\![u_i(y_1,\ldots,y_{n_i})]\!]_A(a_1,\ldots,a_{n_i})$$

for all $a_,\ldots,a_{n_i} \in A$ and $1 \le i \le m$. In particular, any constant operation $k :\to s$ will be interpreted by the evaluation of a *closed* term u in $T(\Sigma,Y)$. The Σ-term defined Σ'-algebra A' is displayed as follows:

Algebra	A'
Carriers	A_s
Constants	$k_{A'} :\to A_s$
Operations	$g^1_{A'} : A_s^{n_1} \to A_s,\ldots,g^m_{A'} : A_s^{n_m} \to A_s$
Definitions	$k_{A'} = [\![u]\!]_A$
	$g^1_{A'}(a_1,\ldots,a_{n_1}) = [\![u_1(y_1,\ldots,y_{n_1})]\!]_A(a_1,\ldots,a_{n_1})$
	$\vdots$
	$g^m_{A'}(a_1,\ldots,a_{n_m}) = [\![u_m(y_1,\ldots,y_{n_m})]\!]_A(a_1,\ldots,a_{n_m})$

Since each operation in Σ' is defined by the terms in $T(\Sigma, Y)$, we can define a mapping to translate terms from $T(\Sigma', Y)$ to $T(\Sigma, Y)$.

Definition 4.3.6 *We define a function $\alpha : T(\Sigma', Y) \rightarrow T(\Sigma, Y)$ recursively over terms in $T(\Sigma', Y)$ as follows:*

$$\alpha(y) = y,$$

for each variable $y \in Y$, and

$$\alpha(k) = u,$$

for each Σ'-constant $k :\rightarrow s$ where u is a closed term in $T(\Sigma)$, and

$$\alpha(g^i(t_1, \ldots, t_{n_i})) = u_i(\alpha(t_1), \ldots, \alpha(t_{n_i}))$$

where $t_1, \ldots, t_n \in T(\Sigma', Y)$ and $g^i : s^{n_i} \rightarrow s$ is a Σ'-term definable operation defined by the term $u \in T(\Sigma, Y)$.

Let $H \subseteq A$ be a subset of the carrier set A and consider the set $\langle H \rangle_{\Sigma'}$ of all elements generated by the repeated application of operations in Σ' on H. We define such a set recursively on the Σ'-terms:

$$[\![y]\!]_{A'}(v_i) \in \langle H \rangle_{\Sigma'}$$

for the variable $y \in T(\Sigma', Y)$ and a value $v_i \in H$,

$$[\![k]\!]_{A'} \in \langle H \rangle_{\Sigma'}$$

for the Σ'-constant symbol $k :\rightarrow s$,

$$[\![g^1(t_1, \ldots, t_{n_1})]\!]_{A'}(h_1, \ldots, h_{n_1}) \in \langle H \rangle_{\Sigma'}$$
$$\vdots$$
$$[\![g^m(t_1, \ldots, t_{n_m})]\!]_{A'}(h_1, \ldots, h_{n_m}) \in \langle H \rangle_{\Sigma'}$$

for Σ-operation $g^i : s^{n_i} \rightarrow s$, terms $t_1, \ldots, t_{n_i} \in T(\Sigma', Y)$ and $h_1, \ldots, h_{n_i} \in H$, for $1 \leq i \leq m$.

The main result of this section shows that $\langle H \rangle_{\Sigma'}$ is indeed a subset of $\langle H \rangle_{\Sigma}$. To begin, we first prove a technical lemma:

Sub Lemma 4.3.1 *For any term $t \in T(\Sigma', Y)$*

$$[\![t]\!]_{A'}(a_1, \ldots, a_n) = [\![\alpha(t)]\!]_A(a_1, \ldots, a_n)$$

for all $a_1, \ldots, a_n \in A$.

PROOF We prove the result by structural induction over $T(\Sigma', Y)$.
Base cases: For variables $y \in Y$

$$[\![y]\!]_{A'}(a_1) = a_1 = [\![y]\!]_A(a_1) = [\![\alpha(y)]\!]_A(a_1).$$

For constant $k :\to s$

$$[\![k]\!]_{A'} = k_{A'} = [\![u]\!]_A = [\![\alpha(k)]\!]_A.$$

Structural Induction: Let $t_1, \ldots, t_n \in T(\Sigma', Y)$. For any Σ'-operation $g^i : s^{n_i} \to s$ we have

$$[\![g^i(t_1, \ldots, t_n)]\!]_{A'}(a_1, \ldots, a_n)$$

since g^i is a Σ'-term defined operation we have

$$\begin{aligned} &= [\![u_i(t_1, \ldots, t_{n_i})]\!]_{A'}(a_1, \ldots, a_n) \\ &= u_i([\![t_1]\!]_{A'}(a_1, \ldots, a_n), \ldots, [\![t_{n_i}]\!]_{A'}(a_1, \ldots, a_n)) \end{aligned}$$

by the definition of term evaluation. The base cases yield:

$$=u_i([\![\alpha(t_1)]\!]_A(a_1, \ldots, a_n), \ldots, [\![\alpha(t_{n_i})]\!]_A(a_1, \ldots, a_n))$$

and by definition of term evaluation

$$=[\![u_i(t_1, \ldots, t_{n_i})]\!]_A(a_1, \ldots, a_n)$$

then by the definition of the mapping α we have

$$=[\![\alpha(g^i(t_1, \ldots, t_{n_i}))]\!]_A(a_1, \ldots, a_n)$$

which completes the proof. ■

Lemma 4.3.3 *If $H \subseteq A$ then $\langle H \rangle_{\Sigma'} \subseteq \langle H \rangle_\Sigma$.*

PROOF For any Σ'-term t and values $h_1, \ldots, h_n$ in H we have

$$[\![t]\!]_{A'}(h_1, \ldots, h_n) \in \langle H \rangle_{\Sigma'}$$

by the recursive definition of $\langle H \rangle_{\Sigma'}$. But by SubLemma 4.3.1

$$\begin{aligned} [\![t]\!]_{A'}(h_1, \ldots, h_n) &= [\![\alpha(t)]\!]_A(h_1, \ldots, h_n) \\ &\in \langle H \rangle_\Sigma \end{aligned}$$

since $\alpha(t)$ is a Σ-term and by the recursive definition of $\langle H \rangle_\Sigma$. ■

4.4 Pointwise Lifting of Lattices to $F(X, A)$

Recalling Definition 3.6.5, we consider lattices as an algebraic structure. Let L be a non-empty set. We define operations *meet* and *join*, denoted $\wedge$ and $\vee$ respectively, and specify axioms that these operations satisfy. We display the lattice signature Σ_{lat} and axioms T_{lat} as follows.

Signature	Σ_{lat}
Sorts	*lat*
Operations	$\wedge : lat \times lat \longrightarrow lat$ $\vee : lat \times lat \longrightarrow lat$

We list the eight lattice axioms:

Axioms	T_{lat}
Idempotent	$a \wedge a = a$ $a \vee a = a$
Commutative	$a \wedge b = b \wedge a$ $a \vee b = b \vee a$
Associativity	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$ $a \vee (b \vee c) = (a \vee b) \vee c$
Absorption	$a \wedge (a \vee b) = a$ $a \vee (a \wedge b) = a$

The signature Σ_{lat} and the axioms in T_{lat} define an axiomatic specification for lattices. The set of all Σ_{lat}-algebras which interpret the operations $\wedge$ and $\vee$ and that satisfy the axioms in T_{lat} is denoted as

$$Alg(\Sigma_{lat}, T_{lat}).$$

Definition 4.4.1 (Lattice) *A Σ_{lat}-algebra L is defined to be a a lattice if, and only if, it satisfies the above axioms. i.e., $L \in Alg(\Sigma_{lat}, T_{lat})$.*

A useful example of a lattice is a lattice structure on the real numbers:

Example 4.4.1 Let $R = (\mathbb{R}, \wedge, \vee)$ be a lattice where the meet of two elements $x, y \in \mathbb{R}$ is defined as $x \wedge y = \min(x, y)$, and the join as $x \vee y = \max(x, y)$.

We display this Σ_{lat}-algebra as:

Algebra	R
Carriers	$\mathbb{R}$
Operations	$\wedge : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ $\vee : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
Definitions	$x \wedge y = \min(x, y)$ $x \vee y = \max(x, y)$

It can be easily shown that the operations defined in R satisfy the lattice axioms, and hence $R \in Alg(\Sigma_{lat}, T_{lat})$.

4.4.1 The Pointwise Lifting of Lattices

We define the operations of meet and join, $\wedge_F$ and $\vee_F$ respectively, on $F(X, L)$ pointwise by

$$(f \wedge_F g)(x) = f(x) \wedge_L g(x),$$

and

$$(f \vee_F g)(x) = f(x) \vee_L g(x).$$

We prove

Corollary 4.4.1 *If L is a lattice then $F(X, L)$ satisfies the axioms of a lattice. That is, $F(X, L) \in Alg(\Sigma_{lat}, T_{lat})$.*

Proof Since the lattice axioms are equational formulae, this result follows directly from the Validity Corollary 4.3.1. That is, if the result holds on L, then it holds on the pointwise lifting $F(X, L)$. ∎

Examples

To illustrate, we can extend Example 4.4.1 pointwise as follows. Let $R = (\mathbb{R}, \wedge, \vee)$ be the lattice over the real numbers and denote the set of all functions mapping points of X to R as

$$F(X, R) = \{f \mid f : X \to R\}.$$

We define the meet and join operations for $f, g \in F(X, R)$ as $(f \wedge g)(x) = \min(f(x), g(x))$ and $(f \vee g)(x) = \max(f(x), g(x))$. We display the Σ_{lat}-algebra as follows:

Algebra	$F(X, R)$
Import	R
Carriers	$F(X, \mathbb{R})$
Operations	$\wedge : F(X, \mathbb{R})^2 \to F(X, \mathbb{R})$ $\vee : F(X, \mathbb{R})^2 \to F(X, \mathbb{R})$
Definitions	$(f \wedge g)(x) = f(x) \wedge g(x)$ $(f \vee g)(x) = f(x) \vee g(x)$

.

4.4.2 Sublattices

Definition 4.4.2 (Sublattice) *Let L be a lattice. A lattice L' is a sublattice of L if, and only if, L' is a Σ_{lat}-subalgebra of L.*

This means that L' is closed under the meet and join operations. That is, for $x, y \in L'$ both $x \wedge y$ and $x \vee y$ are in L'.

The unit interval $[0, 1] \subset \mathbb{R}$ and the meet and join operations constitute a sublattice of $R = (\mathbb{R}, \wedge, \vee)$ since for $x, y \in [0, 1]$ $x \wedge y = \min(x, y)$ and $x \vee y = \max(x, y)$ are both in $[0, 1]$. We display this Σ_{lat}-subalgebra as follows.

Algebra	I
Carriers	$[0, 1]$
Operations	$\wedge : [0, 1] \times [0, 1] \to [0, 1]$ $\vee : [0, 1] \times [0, 1] \to [0, 1]$
Definitions	$x \wedge y = \min(x, y)$ $x \vee y = \max(x, y)$

In fact, any subset of $\mathbb{R}$ is a sublattice of R

Lemma 4.4.1 *Let $L = (L, \wedge_L, \vee_L)$ be a lattice and $F(X, L) = (F(X, L), \wedge, \vee)$ the pointwise lifted lattice. If L' is a sublattice of L then $F(X, L')$ is a sublattice of $F(X, L)$.*

PROOF This is immediately true from Lemma 4.2.1. ∎

In particular, we know that $I = ([0, 1], \wedge, \vee)$ is a sublattice of $R = (\mathbb{R}, \wedge, \vee)$. Then by Lemma 4.4.1 $F(X, I)$ is a sublattice of $F(X, R)$. We display the pointwise lifted Σ_{lat}-algebra $F(X, I)$ as follows:

Algebra	$F(X, I)$
Import	I
Carriers	$F(X, [0, 1])$
Operations	$\wedge : F(X, [0, 1])^2 \rightarrow F(X, [0, 1])$ $\vee : F(X, [0, 1])^2 \rightarrow F(X, [0, 1])$
Definitions	$(f \wedge g)(x) = f(x) \wedge g(x)$ $(f \vee g)(x) = f(x) \vee g(x)$

4.5 Applications to CVG

We show how the results of this chapter can be applied to Constructive Volume Geometry.

Recall the equational laws of scalars in S which we listed in Chapter 2, Section 2.2.1.

Applying the Equational Validation Theorem 4.3.1 to the laws for operations on S we immediately have identical equational laws for the set $F(X, S)$, for any set X. Consider the law

$$s_1 + \max(s_2, s_3) = \max(s_1 + s_2, s_1 + s_3)$$

on scalars $s_1, s_2, s_3 \in S$. Then for scalar fields $\phi_1, \phi_2, \phi_3 \in F(X, S)$ we have for all $x \in X$

$$\begin{aligned}(\phi_1 + Max(\phi_2, \phi_3))(x) &= \phi_1(x) + \max(\phi_2(x), \phi_3(x)) \\ &= \max(\phi_1(x) + \phi_2(x), \phi_1(x) + \phi_3(x)) \\ &= (Max(\phi_1 + \phi_2, \phi_1 + \phi_3))(x).\end{aligned}$$

The other laws listed in Table 2.1 follow immediately as a result of the Validity Theorem 4.3.1.

4.5.1 CSG Embedded in CVG

With our algebraic notation and results thus far, we are able to show an example of how CVG can represent other models of computer graphics currently in use.

Constructive Solid Geometry is one such example, and we have already introduced this model in Chapter 2. We show that there exists an injective homomorphism (Recall Definition 3.9.1 in the Preliminaries) $\epsilon : A_{CSG} \rightarrow A_{CVG}$ which shows that CSG can be embedded in CVG. We prove

Theorem 4.5.1 *The Constructive Solid Geometry (CSG) based on union $\cup$, intersection $\cap$ and difference $-$ is isomorphic to the corresponding Boolean Opacity-Only Model of CVG based on $\boxed{\cup}$, $\boxed{\cap}$and $\boxed{-}$. That is, we can define an bijective homomorphism $\epsilon : A_{CSG} \rightarrow A_{CVG}$.*

PROOF Let $\mathbb{B} = \{0,1\}$. A mapping $\epsilon : \mathscr{P}(E^3) \rightarrow F(E^3, \mathbb{B})$ is defined by

$$\epsilon(A) = \phi_A$$

where $\phi_A \in F(E^3, \mathbb{B})$ is a spatial object such that

$$\phi_A(z) = \begin{cases} 1 & \text{if } z \in A, \\ 0 & \text{if } z \notin A \end{cases}$$

for every $z \in E^3$.

That is, ϵ takes a CSG object defined by a subset $A \subset E^3$ and maps it to a CVG object which at every point z inside A the data is set to 1 and every other point outside A is 0, thereby specifying the geometry of the object in the CVG Boolean Opacity-Only Model.

We show that ϵ is injective. Let $A, B \subset E^3$.

$$\begin{aligned} \epsilon(A) \neq \epsilon(B) &\Longrightarrow (\exists z \in E^3)(\epsilon(A)(z) \neq \epsilon(B)(z)) \\ &\Longleftrightarrow (\exists z \in E^3)(z \in A \not\leftrightarrow z \in B) \\ &\Longrightarrow A \neq B \end{aligned}$$

We show ϵ is surjective. Let ϕ in $F(E^3, \mathbb{B})$ be given. Then define $A = \{z \in E^3 \mid \phi(z) = 1\}$, and thus $\epsilon(A) = \phi$.

We now show that the mapping ϵ has the homomorphism properties

$$\begin{aligned} \epsilon(A \cup B) &= \epsilon(A)\boxed{\cup}\epsilon(B) \\ \epsilon(A \cap B) &= \epsilon(A)\boxed{\cap}\epsilon(B) \\ \epsilon(A - B) &= \epsilon(A)\boxed{-}\epsilon(B). \end{aligned}$$

To show this, we inspect the following truth tables:

$\phi_A(z)$	$\phi_B(z)$	$\phi_{A\cup B}(z)$	$(\phi_A \boxed{\cup} \phi_B)(z)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

$$\epsilon(A \cup B) = \epsilon(A) \boxed{\cup} \epsilon(B)$$

$\phi_A(z)$	$\phi_B(z)$	$\phi_{A\cap B}(z)$	$(\phi_A \boxed{\cap} \phi_B)(z)$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

$$\epsilon(A \cap B) = \epsilon(A) \boxed{\cap} \epsilon(B)$$

$\phi_A(z)$	$\phi_B(z)$	$\phi_{A-B}(z)$	$(\phi_A \boxed{-} \phi_B)(z)$
0	0	0	0
0	1	0	0
1	0	1	1
1	1	0	0

$$\phi_{A-B}(z) = (\phi_A \boxed{-} \phi_B)(z)$$

Thus, by inspection of the tables, ϵ is an isomorphism. ■

4.6 Notes and Sources

This study of the basic algebra of spatial objects $F(X, A)$ seems to be new and is joint work with my supervisor. It was also briefly discussed in [43]. Although not difficult it is essential in laying a foundation for what follows.

The basic idea is to model algebraically spatial objects as an abstract data type. If this is done well - as I believe it is - then we can use the theory of abstract data types to study specifications and compare representations. Furthermore we can use the theory of programming over abstract data types to model high level programs over spatial object data types. This, of course, is a critical issue for the development of Constructive Volume Geometry.

Chapter 5

Spatial Objects Over Discrete Space

5.1 Introduction

We have defined spatial objects

$$\phi : X \to A$$

and some operations on spatial objects for any set X and any algebra A. In this chapter we consider the case where X is a discrete space such as

$$\mathbb{N},\ \mathbb{N}^k,\ \{0,1\}^k,\ \mathbb{Z},\ \mathbb{Z}^k$$

which, for example, represent memories and grids. (Examples 4.2.4 and 4.2.2).

The discrete case is of great importance for computing and so we develop some theory especially for spatial objects over discrete spaces.

When X is infinite the space $F(X, A)$ of all spatial objects is uncountable and complicated. The theme of this chapter is to look at *finite approximations* based on restricting the spatial object to some finite subspace $X_{fin} \subset X$ and consider how

$$\phi : X_{fin} \to A$$

approximates

$$\phi : X \to A.$$

This idea of a finite approximation is captured by the basic open sets of a topology on $F(X, A)$.

Actually, this approach is generalised in the next chapter to the case of continuous spaces X. So this chapter also serves to motivate and guide later theory as well as yield insights to the main computational case.

In this chapter, we consider some topological properties of $F(X, A)$, when X is a discrete space. We have already seen some examples of discrete spaces in Chapter 4, namely the discrete grid of integers $\mathbb{Z}^2$, and memory address space where $Addr = \mathbb{N}$ and $Addr = \{0,1\}^k$. Recall the general signature Σ:

Signature	Σ
Sorts	s
Constants	$\dots, c :\to s, \dots$
Operations	$\dots, f : s \times \cdots \times s \to s, \dots$

We define the Σ-algebra A as

Algebra	A
Carriers	A_s
Constants	$\dots, c_A :\to A_s, \dots$
Operations	$\dots, f_A : A_s \times \cdots \times A_s \to A_s, \dots$

and following Chapter 4, we pointwise lift the operations in A to make a Σ-algebra $F(X, A)$.

Algebra	$F(X, A)$
Carriers	$F(X, A)$
Constants	$\dots, c_F :\to F(X, A), \dots$
Operations	$\dots, f_F : F(X, A) \times \cdots \times F(X, A) \to F(X, A), \dots$
Definitions	$\dots, c_F(x) = c_A, \dots$ $\dots, f_F(\phi_1, \dots, \phi_n)(x) = f_A(\phi_1(x), \dots, \phi_n(x)), \dots$

Some fundamental topological properties of $F(X, A)$ are discussed in Section 5.2: separation, compactness and density.

We show $F(X, A)$ inherits the Hausdorff property from the topology on A, and give necessary and sufficient conditions for a subset $S \subseteq X$ to be compact

(Theorems 5.2.1 and 6.4.4) Lastly, density properties of $F(X, A)$ are discussed and Theorem 5.2.2 shows how the set $C_{ae}(X, A)$ of almost everywhere constant functions are dense in $F(X, A)$.

Section 5.3 discusses the continuity of operators on $F(X, A)$ when A is a topological Σ-algebra and X is a discrete set. In particular, Theorem 5.3.3 shows that the pointwise lifted functions on $F(X, A)$ are continuous with respect to the topology on $F(X, A)$. The evaluation and substitution functions are defined, and are shown to be continuous (Theorems 5.3.1 and 5.3.2).

Section 5.4 is dedicated to an important example of discrete spaces. In this section, we show how an unlimited register machine can be generalised to compute arbitrary data on a discrete space X. The mathematical framework developed in this chapter can be used to model a machine and be a useful tool in the analysis of computations.

Recall Example 4.2.7 in Chapter 4 shows how spatial objects can be viewed as a Cartesian product A^X of the algebra A indexed by the space X. We further extend this example to show topological relationships between A^X and $F(X, A)$. We denote

$$\mathbf{A}_x = \prod\{A_x : x \in X\}$$

to be the set of all functions $\phi : X \to A_x$, such that $\phi(x) \in A_x$ for each $x \in X$.

The product topology was defined in Preliminaries Definition 3.8.15 where the basic open sets of the product topology are of the form

$$U_{x_1} \times \cdots \times U_{x_n} \times \prod\{A_x \mid x \neq x_1, \ldots, x_n\} = \bigcap_{i=1}^{n} \pi_{x_i}^{-1}[U_{x_i}]$$

where n is finite, U_{x_i} is open in A_{x_i} and $x, x_1, \ldots, x_n \in X$.

For the special case where $A_x = A$ for all $x \in X$, we observe that the Cartesian product $A^X = \prod\{A : x \in X\}$ is the set of all functions $\phi : X \to A$, such that $\phi(x) \in A$ for each $x \in X$, and this is precisely the definition of $F(X, A)$.

Now, let $Z = \{x_1, \ldots, x_n\}$ be a finite subset of the index set X. A typical element B of the basis for the product topology is of the form

$$\begin{aligned} B &= \bigcap\{\pi_{x_i}^{-1}[U_{x_i}] : x_i \in Z\} \\ &= \{\phi : \phi(x_i) \in U_{x_i} \text{ where } x_i \in Z,\ U_{x_i} \text{ open in } A \text{ and } 1 \leq i \leq n\}. \end{aligned}$$

The set B represents all functions $\phi : X \to A$ mapping points in $Z \subset X$ to open sets U_x of A. For our work in this chapter we use the notation

$$\begin{aligned} B &= \{\phi : \phi(x_i) \in U_{x_i} \text{ where } 1 \leq i \leq n\} \\ &= B(x_1, \ldots, x_n, U_1, \ldots, U_n) \end{aligned}$$

for the basic open sets of $F(X, A)$, with X a discrete set.

Intuitively, one can interpret this set of functions as approximations of the values of spatial objects - via $U_1, \ldots, U_n$ - on finitely many points $x_1, \ldots, x_n$ in the space X.

5.2 Properties of the Topology on $F(X, A)$

In this section, we discuss some topological properties possessed by $F(X, A)$ when X has the discrete topology and A is a topological Σ-algebra.

5.2.1 Separation

We start by proving a useful theorem showing how the separability of the topological space T_A implies the separability property for the topology T of $F(X, A)$.

Recall that if a topological space T_A of A is Hausdorff, (i.e. T_2-separable) then for any two unique points $x, y \in A$ we can find two open sets $U, V \in T_A$ such that $x \in U$, $y \in V$ and these sets are disjoint; $U \cap V = \emptyset$.

Theorem 5.2.1 (Pointwise Lifting of the Hausdorff Property) *If T_A is a Hausdorff space, then so is T.*

Proof Let f and g be two distinct functions in the space $F(X, A)$. Then there is a point $x \in X$ such that $f(x) \neq g(x)$. Since A is Hausdorff, we can find disjoint sets $U, V \in A$ such that

$$f(x) \in U, g(x) \in V,$$

or equivalently,

$$f \in B(x, U), g \in B(x, V).$$

To see that $B(x, U) \cap B(x, V) = \emptyset$ suppose, for a contradiction, that there is a function $h \in F(X, A)$ such that

$$h \in B(x, U) \cap B(x, V) = B(x, U \cap V).$$

Then the point $h(x)$ is mapped to the open set $U \cap V$. But we chose U and V such that $U \cap V = \emptyset$. Therefore $B(x, U) \cap B(x, V) = \emptyset$ implying T is Hausdorff. ∎

5.2.2 Density

In the context of the topology on $F(X, A)$, we prove a theorem that shows that a set D is dense in $F(X, A)$. Thus for any function $\phi \in F(X, A)$, we can use the functions in the set D to come within an arbitrary degree of precision of ϕ. This property is essential if we wish to approximate functions using other "simpler", more convenient functions.

We start this section with an observation of what it means for a set to be dense in $F(X, A)$.

Observation 5.2.1 *A set D is dense in $F(X, A)$ if, and only if, for any basic open set*

$$B = B(x_1, \dots, x_n, U_1, \dots, U_n)$$

the intersection $D \cap B \neq \emptyset$.

To show that a set D is dense in $F(X, A)$ we must exhibit, for any basic open set B, that a function ϕ can always be found such that $\phi \in B \cap D$. To this end, we define:

Definition 5.2.1 *Let $\{x_1, \dots, x_n\}$ be a subset of X and $\{y_1, \dots, y_n\}$ a subset of A_s. Let a be an element of A_s. A function $\phi_{i,a} : X \longrightarrow A_s$ is said to be almost everywhere constant if for a point $x \in X$*

$$\phi_{i,a}(x) = \begin{cases} y_i & \text{if } x \in \{x_1, \dots, x_n\}, \\ a & \text{otherwise.} \end{cases}$$

Let $C_{ae}(X, A)$ be the set of all "almost everywhere constant" functions mapping X to A. Each function in the set varies at only a finite number of points $\{x_1, \dots, x_n\}$ in X, and is a constant value at every other point. We show that the set C_{ae} is a dense set in $F(X, A)$.

Theorem 5.2.2 *The set $C_{ae}(X, A)$ is dense in $F(X, A)$.*

PROOF We need to show that for any basic open set

$$B = B(x_1, \dots, x_n, U_1, \dots, U_n)$$

there is a function common in both sets; i.e. $B \cap C_{ae}(X, A) \neq \emptyset$.

For each point x_i and each open set U_i in A choose a $y_i \in U_i$. Then we have a function $\phi : X \longrightarrow A$ that maps each point in $\{x_1, \dots, x_n\}$

$$\begin{aligned} \phi(x_1) &= y_1, \\ \phi(x_2) &= y_2, \\ &\vdots \\ \phi(x_n) &= y_n. \end{aligned}$$

The set B contains all functions mapping points to the open sets $U_1, \dots, U_n$, so the function ϕ is contained in B as it maps points to elements in the specified open sets.

The set $C_{ae}(X, A)$ contains *all* functions mapping a finite number of points to elements of A. We choose the function in $\phi \in C_{ae}(X, A)$ such that

$$\phi : (x_1, \dots, x_n) \mapsto (y_1, \dots, y_n).$$

Thus given any basic open set $B(x_1, \dots, x_n, U_1, \dots, U_n)$ we may choose suitable elements $\{y_1, \dots, y_n\}$ in each open set $\{U_1, \dots, U_n\}$ respectively, and find a function in $C_{ae}(X, A)$ that maps $\phi(x_i) = y_i$ for $1 \leq i \leq n$. Therefore $C_{ae}(X, A) \cap B \neq \emptyset$. ■

Corollary 5.2.1
Let $X = \mathbb{N}$ and $A = \mathbb{N}$ both with the discrete topology. Then the set of all almost everywhere constant functions $C_{ae}(X, A)$ is dense in $F(\mathbb{N}, \mathbb{N})$.

PROOF The almost everywhere constant functions can be defined, for example, as

$$\phi_c(x) = \begin{cases} y_i & \text{if } x \in \{x_1, \dots, x_n\}, \\ c & \text{otherwise,} \end{cases}$$

where $c \in \mathbb{N}$. The result follows immediately from Theorem 5.2.2. ■

5.3 Continuous Operators of $F(X, A)$

This section provides some results on the continuity of the operators *eval* and *sub* and shows that pointwise lifting preserves continuity. As with the rest of this chapter, the set X is given the discrete topology where every subset of X is an open set. The consequence of this is that each function in $F(X, A)$ is continuous, regardless of the topology given to A.

5.3.1 Continuity of Evaluation and Substitution

Recall the evaluation function $eval : F(X, A) \times X \to A$ was defined in Chapter 4 in Section 4.2.5 as

$$eval(\phi, x) = \phi(x),$$

which for a spatial object ϕ and a point x in X, returns a data value in A. We prove

Theorem 5.3.1 (Continuity of Evaluation) *The function $eval : F(X, A) \times X \to A$ is continuous.*

PROOF We need to show that for an open set U of A the preimage $eval^{-1}[U]$ is an open set in the product topology $F(X, A) \times X$.

Observe that $eval^{-1}[U] = \{(f, x) \mid f(x) \in U\}$ where $f \in F(X, A)$ and x is a point in X. We show the set

$$\bigcup_{x \in X} \{B(x, U) \times \{x\}\} \tag{5.1}$$

is equal to the set

$$\{(f, x) \mid f(x) \in U\}. \tag{5.2}$$

5.1 $\subseteq$ 5.2: For an element (f, x) of 5.1 where $f \in B(x, U)$ we have

$$\begin{aligned} eval(f, x) &= f(x) \\ &\in U, \end{aligned}$$

and thus $(f, x) \in eval^{-1}[U]$.
$5.2 \subseteq 5.1$: For a typical element $(g, x) \in eval^{-1}[U]$ we have that $g(x) \in U \iff g \in B(x, U)$. Therefore $(g, x) \in B(x, U) \times \{x\}$. ■

Theorem 5.3.2 (Continuity of Substitution)
The function $sub : F(X, A) \times X \times A \to F(X, A)$ *defined as*

$$sub(\phi, x, a)(y) = \begin{cases} a & \textit{if } x = y, \\ \phi(y) & \textit{otherwise,} \end{cases}$$

is continuous.

Proof We show that the preimage $sub^{-1}[B]$ is open for any basic open set

$$B = B(x_1, \ldots, x_n, U_1, \ldots, U_n)$$

in $F(X, A)$, where U_i is an open set in A for $1 \leq i \leq n$.

Consider the parameter X. There are two cases:

Case (i) We choose an x_i such that $x_i \in \{x_1, \ldots, x_n\}$. Then for $sub(f, x_i, a)$ we must have $a \in U_i$ for x_i giving the open set

$$\bigcup_{i=1,\ldots,n} B(x_1, x_{i-1} \ldots, x_{i+1}, x_n, U_1, U_{i-1}, \ldots, U_{i+1}, U_n) \times \{x_i\} \times U_i.$$

Case (ii) We choose an x such that $x \notin x_1, \ldots, x_n$. In this case we need not worry about which open set a is in. We have the open set

$$\bigcup_{x \in X} B(x_1, \ldots, x_n, U_1, \ldots, U_n) \times \{x\} \times A,$$

which is an open set on the product topology $F(X, A) \times X \times A$. ■

5.3.2 Continuity of Pointwise Lifting

We specify the pointwise lifting of functions on $F(X, A)$ and show that they are continuous whenever X has the discrete topology and A is an arbitrary topological Σ-algebra. Recall the pointwise extension $f_{F(X,A)}$ of a function f_A is defined at a point x in X by

$$f_{F(X,A)}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$$

where $\phi_1, \ldots, \phi_n$ are functions in $F(X, A)$.

We wish to show that the pointwise lifting of functions in the Σ-algebra A preserves continuity. We start by proving a lemma which shows the composition of continuous functions is continuous.

Lemma 5.3.1 *Let $f : A^n \longrightarrow A$ and $g : X \longrightarrow A^n$ be continuous functions. Then the composition $(f \circ g) : X \longrightarrow A$ is continuous.*

PROOF Denote the composition $h = (f \circ g)$. Given an open set $U \subseteq A$ we must exhibit the preimage $h^{-1}[U]$ is open in X. We have

$$\begin{aligned} h^{-1}[U] &= (f \circ g)^{-1}[U] \\ &= g^{-1}(f^{-1}[U]). \end{aligned}$$

By the continuity of f, we know that $V = f^{-1}[U]$ is an open set in A^n. Furthermore, by the continuity of g, the preimage $g^{-1}[V]$ is open in X, and thus the composition $(f \circ g)$ is continuous. ■

We wish to prove that the pointwise lifting of the Σ-algebra A to $F(X, A)$ preserves continuity. To this end we prove

Lemma 5.3.2 *$F(X, A)^n$ is homeomorphic to $F(X, A^n)$.*

PROOF Define the function $\Phi : F(X, A)^n \longrightarrow F(X, A^n)$ by

$$\Phi(\phi_1, \ldots, \phi_n)(x) = (\phi_1(x), \ldots, \phi_n(x))$$

for functions $\phi_1, \ldots, \phi_n$ in $F(X, A)$ and $x \in X$. We show Φ is a bicontinuous bijection.

Bijectivity: Let $(\phi_1, \ldots, \phi_n)$ and $(\phi'_1, \ldots, \phi'_n)$ be elements of $F(X, A)^n$ and suppose we have $(\phi_1, \ldots, \phi_n) \neq (\phi'_1, \ldots, \phi'_n)$. Then there is at least one $x \in X$ such that

$$(\phi_1(x), \ldots, \phi_n(x)) \neq (\phi'_1(x), \ldots, \phi'_n(x))$$

since there is one or more coordinate functions that are not equal at x. By definition of the mapping Φ we have

$$\begin{aligned} &\Longrightarrow \Phi(\phi_1, \ldots, \phi_n)(x) \neq \Phi(\phi'_1, \ldots, \phi'_n)(x) \\ &\Longrightarrow \Phi(\phi_1, \ldots, \phi_n) \neq \Phi(\phi'_1, \ldots, \phi'_n). \end{aligned}$$

To show surjectivity, pick a $f \in F(X, A^n)$. It will have the form

$$f(x) = (a_1, \ldots, a_n)$$

for elements $a_1, \ldots, a_n \in A$ and a point $x \in X$. But for each a_i we can find a function $\phi_i \in F(X, A)$ such that $a_i = \phi(x)$ for $x \in X$. Then setting $\Psi = (\phi_1, \ldots, \phi_n)$ we have

$$\Phi(\Psi)(x) = f(x).$$

Bicontinuity: To show that Φ is a continuous mapping, let an open set B of $F(X,A^n)$ be given. It will have the form

$$B = B(x_1,\ldots,x_n,V_1,\ldots,V_n)$$

where each $V_i = U_1^i \times \cdots \times U_n^i$ $1 \le i \le n$ is the product of open sets U_j^i in A for $1 \le j \le n$. Thus each V_i is open in the product topology on A^n. Now by the definition of the function Φ, we simply have the preimage $\Phi^{-1}[B]$ mapping B to the set $B^1 \times \cdots \times B^n$ where

$$B^i = B(x_1,\ldots,x_n,U_1^i,\ldots,U_n^i),\ 1 \le i \le n.$$

Clearly each set B^i is open in $F(X,A)$, and thus $B^1 \times \cdots \times B^n$ is open in the product topology of $F(X,A)^n$, completing the proof.

Conversely, let B be an open set of $F(X,A)^n$ where $B = B^1 \times \cdots \times B^n$ such that

$$B^i = B(x_1,\ldots,x_n,U_1^i,\cdots,U_n^i),\ 1 \le i \le n$$

for open sets U_j^i in A, $1 \le j \le n$.

Now by the definition of Φ, we have

$$\Phi[B^1 \times \cdots \times B^n] = B(x_1,\ldots,x_n,V_1,\ldots,V_n)$$

where each $V_i = U_1^i \times \cdots \times U_n^i$ $1 \le i \le n$ is open in A^n. That is, V_i is the product of open sets U_j^i in A for $1 \le j \le n$.

Thus $B(x_1,\ldots,x_n,V_1,\ldots,V_n)$ is an open set of $F(X,A^n)$, completing the proof. ∎

The next Theorem is an easy consequence of Lemmas 5.3.1 and 5.3.2.

Theorem 5.3.3 (Continuity of Pointwise Lifting)
The pointwise lifting

$$f_{F(X,A)}(\phi_1,\ldots,\phi_n)(x) = f_A(\phi_1(x),\ldots,\phi_n(x))$$

of a continuous function f_A is continuous for $\phi_1,\ldots,\phi_n$ in $F(X,A)$, $x \in X$.

PROOF We can express the pointwise lifting as

$$\begin{aligned} f_{F(X,A)}(\phi_1,\ldots,\phi_n)(x) &= (f_A \circ \Phi(\phi_1,\ldots,\phi_n)(x)) \\ &= f_A(\phi_1(x),\ldots,\phi_n(x)). \end{aligned}$$

That is, pointwise lifting is equal to the composition of functions f_A and Φ. Now by assumption f_A is continuous and Lemma 5.3.2 proves the continuity of Φ. Therefore by Lemma 5.3.1 we have the result. ∎

Alternatively, we can lift operations on space to spatial objects. We show that such liftings are also continuous.

Theorem 5.3.4 (Continuity of Spatial Pointwise Lifting) *Let $\alpha : X \to X$ be a continuous spatial transformation and $\overline{\alpha} : F(X, A) \to F(X, A)$ the pointwise lifting defined at a point x in X as*

$$\overline{\alpha}(\phi)(x) = \phi(\alpha(x))$$

for continuous $\phi \in F(X, A)$. Then $\overline{\alpha}$ is continuous.

Proof We show $\overline{\alpha}$ maps open sets to open sets. Let U be an open set of A. Then by the continuity of ϕ

$$V = \phi^{-1}[U]$$

is an open set in X. By the continuity of α

$$W = \alpha^{-1}[U]$$

is an open set in X. Thus $\overline{\alpha}$ is continuous. ∎

We shall study both the pointwise lifting of spatial and data operations in greater generality in Chapter 6.

5.4 Modelling Abstract Memories and Machines

This section shows how abstract machines can be modelled and analysed using the general theory of spatial objects. The machines we model will have an infinite amount of *registers* which act as the memory for which the *data* we perform our computations on is stored. Memory then, can be thought of as a spatial object from a discrete set of registers to a set of data:

$$memory : Registers \to Data,$$

where each register contains a single element of data.

This abstract machine is a generalisation of an Unlimited Register Machine such as the model found in [23]. We refer to our machine as a *Spatially Extended Register Machine* or SERM.

We give a detailed account of program syntax and semantics, and show how the SERM can be used to compute functions on a Σ-algebra. The Local Computation Theorem 5.4.1 is proved in Section 5.4.5 which is a result showing the output data computed by the SERM machine is found in a subalgebra generated by the user input. Using the algebraic methods introduced in Chapter 4, we show how a SERM can be pointwise lifted to compute in parallel over a discrete space X.

5.4.1 The SERM Computational Model

The SERM shall execute *programs* based on a signature Σ, which gives the names of operations and tests that a SERM may perform in the course of a computation. The register space that the SERM uses to store data will be a discrete space R, consisting of registers indexed by the natural numbers. Each register contains exactly one data value from a Σ-algebra A.

The Σ-algebra A contains carriers, constants and functions. The carriers are an arbitrary data set A_s and the set $\{\mathsf{t}, \mathsf{f}\}$ of Booleans. The constant c_A is a typical constant of type A_s, and t, f are constants of type $\mathbb{B}$. The function f_A is a typical function whose input is a tuple of A_s and has an output type of A_s. The relation w is a relation on A_s whose output is either t or f. The two sorted signature Σ and an associated Σ-algebra A is listed below.

Signature	Σ
Sorts	s, $Bool$
Constants	$c \rightarrow s$
	$true$, $false \rightarrow Bool$
Operations	$\ldots, f : s \times \cdots \times s \rightarrow s, \ldots$
	$\ldots, w : s \times \cdots \times s \rightarrow Bool, \ldots$

Algebra	A
Carriers	$A_s, \mathbb{B}$
Constants	$\ldots, c_A \rightarrow A_s, \ldots$
	$\mathsf{t}, \mathsf{f} \rightarrow \mathbb{B}$
Operations	$\ldots, f_A : A_s \times \cdots \times A_s \rightarrow A_s, \ldots$
	$\ldots, r_A : A_s \times \cdots \times A_s \rightarrow \mathbb{B}, \ldots$

5.4.1.1 The Syntax of SERM Programs

The contents of the registers in R are transformed by the use of *programs*, and we start a computation on the SERM by *executing* a program.

A typical SERM program makes use of registers in R which are specified by means of a *register declaration* D (cf. [59](p.41)). This declaration contains three finite lists, consisting of registers in R, specifying

1. I, the registers containing user input data,
2. O, the registers where the output data will be located when execution has successfully terminated, and
3. S some auxiliary storage registers.

We denote the *declaration body* of a SERM program as

$$D = (I, O, S)$$

and specify that each listing in D are pairwise disjoint. A SERM program may only modify the registers specified in the declaration body, otherwise it is unmodified throughout the execution of the program. Thus we have

Definition 5.4.1 (SERM Programs)
A SERM *program P contains a declaration body $D = (I, O, S)$ and a finite list of instructions indexed by the natural numbers such that*

$$P = (D; 1 : I_1, \ldots, l : I_l).$$

In this definition, P is a SERM program with l instructions, which we call the *length* of P. The set of all SERM programs that use Σ is denoted as

$$\text{SERM}(\Sigma).$$

The SERM maintains a natural number called the *program counter.* It is the task of the program counter to store the index of the instruction that is to be executed next, and this value shall be updated after every instruction has finished its execution. When the last instruction in the program list is executed, the SERM computation is complete and the program counter is set to zero.

There are four types of instructions in the SERM instruction set. They are listed below with an informal description of their semantics.

$r := c$	The value at register r is set to c_A.
$r_1 := r_2$	Copies the value in register r_2 to register r_1 leaving r_2 unaltered.
$r := f(r_1, \ldots, r_n)$	The n-ary operation f is performed on n registers $r_1, \ldots, r_n$ and the result stored in the register r. No other registers are modified.
$Jump_w(r_1, \ldots, r_n, q)$	If the relation $w(r_1, \ldots, r_n)$ is true then jump to instruction q, otherwise continue program execution.

For each constant c in the signature Σ, we define an instruction $r := c$ such that the register r is specified, and when executed, the value c_A is stored in r.

The transfer function $r_1 := r_2$ simply copies the value in register r_2 to register r_1, with r_2 unmodified.

For each operation

$$f : s \times \cdots \times s \to s$$

where f is an n-ary operation in Σ, we define an instruction $r := f(r_1, \ldots, r_n)$ such that n registers are specified and when executed, the operation f_A is performed on the input values stored in $r_1, \ldots, r_n$ and the resulting value is stored in r. No other registers are modified.

The $Jump_w$ instruction is given n input registers $r_1, \ldots, r_n$ and an instruction label $q \geq 1$ such that, if an associated relation w holds, then the program counter is set to q to indicate the q-th instruction is the next instruction to be executed. If the relation does not hold, execution continues to the next instruction in the program listing.

For example, we could have the equality relation $w(r_1, r_2)$ that returns true if $r_1 = r_2$, and false otherwise. Thus, the instruction $Jump_w(r_1, r_2, q)$ would cause a jump to occur only if the specified register contents were equal.

The $Jump_w$ instruction may also cause a computation to finish. If a $Jump_w$ causes the SERM to execute the q-th instruction with $q > l$ then the SERM stops computation and the program counter is set to zero since this instruction does not exist.

5.4.2 States

In order to fully understand what happens when our machine is executing a program, it is necessary to describe precisely the state of the SERM. This means keeping track of the next instruction to be executed and the current contents of the registers. This allows us to have complete access to information about the SERM at any instance of a particular computation.

The term *state* will be used to describe

1. the program counter $c \in \mathbb{N}$,

2. the value of all registers in the machine $\gamma : R \longrightarrow A_s$,

Definition 5.4.2 *A state σ is a pair*

$$(c, \gamma)$$

where $c \in \mathbb{N}$ is the program counter; the label of the next instruction to execute, and $\gamma : R \longrightarrow A_s$ is the configuration of the machine registers; a mapping from the registers in R to their values in A_s such that

$$\gamma[r] = \textit{ the element } a \in A_s \textit{ in the register } r.$$

The use of square brackets are to remind the reader that we are actually accessing a memory location. One could, for example, think of the registers as an array. We can extend this notation to $\gamma[r_1, \ldots, r_m]$ which returns the m-tuple $(a_1, \ldots, a_m)$ where a_i is the element stored in the register r_i, for $1 \leq i \leq m$.

To access the program counter c of the state $\sigma = (c, \gamma)$ we write σ_{pc} and to access the memory configuration γ we write σ_m where m stands for *memory*. To summarise

$$(c, \gamma)_m = \gamma \text{ and } (c, \gamma)_{pc} = c.$$

The collection of all computational states σ is denoted as State(A) and the collection of all machine configurations is denoted Config(A). Since each configuration is a map $\gamma : R \longrightarrow A_s$, the set of configurations is equal to the collection of functions $F(R, A) = \{\gamma : R \longrightarrow A_s\}$. We use this set in the algebra of states:

Signature	*State*
Import	Σ
Sorts	*Config*, $\mathbb{N}$, *State*
Operations	$pc : State \longrightarrow \mathbb{N}$ $m : State \longrightarrow Config$

Algebra	*State*
Import	A
Carriers	Config(A), $\mathbb{N}$, State(A)
Operations	$pc : \text{State}(A) \to \mathbb{N}$ $m : \text{State}(A) \to \text{Config}(A)$
Definitions	$(c, \gamma)_{pc} = c$ $(c, \gamma)_m = \gamma$

5.4.2.1 Special States

In this section we consider the state of the SERM before computation of a program begins, and the state once computation has been finished.

Before we start a computation, we must ensure that the values of the SERM registers are correctly initialised. This includes setting registers to the given input values, and all others set to some default value. We give some definitions starting with *ground states.*

Definition 5.4.3 (Ground State) *A ground state is a state $\sigma = (c, \gamma)$ in which each register in the memory configuration $\gamma : R \longrightarrow A_s$ contains the constant value c_A in A_s.*

Once the SERM is in a ground state, we put the input data into the registers specified by the program declaration. Such states are called *initialised states.*

Definition 5.4.4 (Initialised State) *An initialised state is a ground state initialised with respect to a program declaration D where elements $a_1, \ldots, a_n$ in A_s are stored in the registers specified by the input list I.*

Often times we write $\sigma(a_1, \ldots, a_n)$ to mean that the ground state σ has been initialised with the data $a_1, \ldots, a_n \in A_s$ with respect to a program declaration D, specifying the n input registers the data will be stored in.

When the SERM has finished its computation, it enters a *halt state.* We define such a state as follows:

Definition 5.4.5 (Halt State) *A halt state σ is any state $(0, \gamma)$ whose program counter is* 0.

There are exactly two ways a SERM can enter a halt state:

1. when the last instruction in a SERM program is executed, or
2. when a jump is made to a non-existent instruction number.

5.4.3 A Programming Example

We give an example of a SERM program using the algebra of natural numbers. Below is a listing of the signature Σ_{nat} and a Σ_{nat}-algebra in compact form.

$$\begin{aligned}\Sigma_{nat} &= (nat, Bool; zero, true, false; succ, pred, equal)\\ A &= (\mathbb{N}, \mathbb{B}; 0, tt, ff; x+1, x-1, =)\end{aligned}$$

The instructions unique to the algebra of natural numbers and Booleans are

$$\begin{aligned}&r := zero,\\ &r := succ(r), r := pred(r),\\ &b := true, b := false,\\ &b := (r_1 = r_2).\end{aligned}$$

In this example we compute a program over the natural numbers such that when given two numbers a_1, a_2, calculates the sum a_1+a_2 and outputs the result in register z_1. Intermediate calculations are stored in the auxiliary registers r_1 and r_2.

Program: P_{add}
User Data: (a_1, a_2)
$D.\ I = x_1, x_2;\ O = z_1;\ S = r_1, r_2$
1. $x_1 := a_1$
2. $x_2 := a_2$
3. $r_1 := x_1$
4. $r_2 := x_2$
5. $Jump_=(r_2, zero, 9)$
6. $r_1 := succ(r_1)$
7. $r_2 := pred(r_2)$
8. $Jump_=(r_1, r_1, 5)$
9. $z_1 := r_1$

5.4.4 Operational Semantics

States record the status of the program counter and the register configuration of the SERM at every step of the computation of a program. We need to define in a precise way how each instruction of a program alters the computational state of the SERM. Specifically,

- how does the computation modify the registers in R,
- how does the program counter change,

- what happens when the computation is terminated.

To answer these questions, we must define how each instruction modifies the computational state. For this we consider the *substitution function*

$$sub : R \times A_s \times \text{Config}(A) \to \text{Config}(A)$$

which copies a value $a \in A_s$ to a register r in R. The *sub* function is defined as

$$sub(r, a, \gamma)(r_1) = \begin{cases} a & \text{if } r = r_1, \\ \gamma[r_1] & \text{otherwise.} \end{cases}$$

For brevity we shall write $\gamma[a\backslash r]$ to mean $sub(r, a, \gamma)$. The *sub* function will be used to define the semantics of SERM instructions on states over the Σ-algebra A.

Since the input and output are lists of registers in R, we may also use the *sub* function to modify these registers as well.

We use the term *step* to mean the computation of executing a single program instruction. Initially we discuss the execution of a single instruction, then a function is defined to execute many steps sequentially, with the purpose of executing an entire SERM program. Define the function

$$\text{step} : \text{SERM}(\Sigma) \times \text{State}(A) \to \text{State}(A)$$

such that a given program $P \in \text{SERM}(\Sigma)$ and a state $\sigma = (c, \gamma)$, a state $\text{step}(P, \sigma)$ is returned which is the resulting state after the execution of the c-th program instruction as specified by the program counter. If step is given a halt state $\sigma = (0, \gamma)$ as the beginning state, then the function simply returns σ unmodified.
There are four instructions to consider when defining the function step, and we consider each case:

$$\text{step}(P, \sigma) = \begin{cases} (c+1, \gamma[c_A\backslash r]) & I_c \equiv\ r := c, \\ (c+1, \gamma[\gamma[y]\backslash r]) & I_c \equiv r := y, \\ (c+1, & \\ \quad \gamma[f_A(\gamma[r_1], \ldots, \gamma[r_n])\backslash y] & I_c \equiv\ y := f(r_1, \ldots, r_n), \\ (q, \gamma) & I_c \equiv\ jump_w(r_1, \ldots, r_n, q), \\ & \text{and } w(r_1, \ldots, r_n) = true, \\ (c+1, \gamma) & I_c \equiv\ jump_w(r_1, \ldots, r_n, q), \\ & \text{and } w(r_1, \ldots, r_n) = false, \end{cases}$$

Having defined the step function that computes a single step in a SERM computation and returns the resulting state, we would like to compute many steps in succession such that an entire program may be completely executed and a halt state returned. To this end we define

Definition 5.4.6 (The comp Function) *Define the function*

$$\text{comp} : \text{SERM}(\Sigma) \times \mathbb{N} \times \text{State}(A) \to \text{State}(A)$$

recursively as

$$\begin{aligned} \text{comp}(P, 0, \sigma) &= \sigma \\ \text{comp}(P, k+1, \sigma) &= \text{step}(P, \text{comp}(P, k, \sigma)) \end{aligned}$$

which returns the resulting state after the completion of n computational steps as listed in the program P, beginning at the state σ.

5.4.5 The Local Computation Theorem

We prove a result which is useful in the analysis of the output of SERM programs. The *Local Computation Theorem* 5.4.1 shows that the output values of a SERM(Σ) program P must lie within the subalgebra generated by the application of the Σ-operations on the input data values.

Recalling the definition of generated subalgebra from Preliminaries Section 3.2.2, we give a result which considers the state of the SERM after a single program instruction is executed:

Lemma 5.4.1 (One Step Lemma) *Let P be a* SERM(Σ) *program, $a_1, \ldots, a_n$ data in A_s and $\sigma = (c, \gamma)$ a computation state. If for any register $r \in R$, $\sigma_m[r] \in \langle a_1, \ldots, a_n \rangle_A$ then*

$$\text{step}(P, \sigma)_m[r] \in \langle a_1, \ldots, a_n \rangle_A.$$

PROOF For brevity let $\gamma' = \text{step}(P, \sigma)_m$ and pick a $r \in R$. Now consider the following cases:

Case 1: $I_c \equiv r_1 := c$

$$\text{step}(P, \sigma)_m[r] = \gamma[c_A \backslash r_1][r] = \begin{cases} c_A & \text{if } r = r_1, \\ \gamma[r] & \text{otherwise.} \end{cases}$$

Case 2: $I_c \equiv r_1 := r_2$

$$\text{step}(P, \sigma)_m[r] = \gamma[\gamma[r_2] \backslash r_1][r] = \begin{cases} \gamma[r_2] & \text{if } r = r_1, \\ \gamma[r] & \text{otherwise.} \end{cases}$$

Case 3: $I_c \equiv r_0 := f(r_1, \ldots, r_n)$

$$\begin{aligned} \text{step}(P, \sigma)_m[r] &= \gamma[f_A(\gamma[r_1], \ldots, \gamma[r_n]) \backslash r_0][r] \\ &= \begin{cases} f_A(\gamma[r_1], \ldots, \gamma[r_n]) & \text{if } r = r_0, \\ \gamma[r] & \text{otherwise.} \end{cases} \end{aligned}$$

Case 4: $I_c \equiv Jump_w(r_1, \ldots, r_n, q)$

$$\begin{aligned} \text{step}(P, \sigma) &= (q, \gamma) \quad \text{if } w(r_1, \ldots, r_n) = true, \text{ and} \\ \text{step}(P, \sigma) &= (c+1, \gamma) \quad \text{if } w(r_1, \ldots, r_n) = false. \end{aligned}$$

By inspection of each case, we conclude that any one step computed on the state σ with the assumption that for any register $r \in R$ $\sigma_m[r] \in \langle a_1, \ldots, a_n \rangle_A$ results in a state such that

$$\text{step}(P, \sigma(a_1, \ldots, a_n))_m[r] \in \langle a_1, \ldots, a_n \rangle_A. \qquad \blacksquare$$

We now prove that the comp function returns a state configuration with data that is also in the subalgebra generated by $\langle a_1, \ldots, a_n \rangle_A$.

Theorem 5.4.1 (The Local Computation Theorem)
Let P be a SERM(Σ) *program with declaration D and $a_1, \ldots, a_n \in A_s$ are input values in A_s. If $\sigma(a_1, \ldots, a_n) = (c, \gamma)$ is an initialised state then for n steps of computation*

$$\text{comp}(P, n, \sigma(a_1, \ldots, a_n))_m[r] \in \langle a_1, \ldots, a_n \rangle_A$$

for any register $r \in R$.

Proof Pick a register $r \in R$. We consider two cases based on whether or not our selected register is listed in the program declaration D.
Case 1: The register r is not a declared register. That is, $r \in R \backslash D$. Therefore, r will never be modified during computation of P. Now since the state (c, γ) is initialised to the default value of $c_A \in A_s$, we have $\gamma[r] = c_A$ and thus $\text{comp}(P, n, \sigma(a_1, \ldots, a_n))_m[r] \in \langle a_1, \ldots, a_n \rangle_A$.
Case 2: The register r is a declared register. That is, $r \in D$. We prove the result by induction on the number of steps of computation.

Base Case: Let $n = 0$. We have $\text{comp}(P, 0, \sigma(a_1, \ldots, a_n))_m$ equal to the register configuration γ we started with, and the result holds.

Inductive Assumption: Assume the result holds for k. That is,

$$\text{comp}(P, k, \sigma(a_1, \ldots, a_n))_m[r] \in \langle a_1, \ldots, a_n \rangle_A.$$

Inductive Step: For $k+1$ we have by Definition 5.4.6

$$\text{comp}(P, k+1, \sigma(a_1, \ldots, a_n)) = \text{step}(P, \text{comp}(P, k, \sigma(a_1, \ldots, a_n))),$$

then by the inductive assumption, $\text{comp}(P, k, \sigma(a_1, \ldots, a_n))_m[r]$ is in $\langle a_1, \ldots, a_n \rangle_A$ and by the One Step Lemma 5.4.1 $\text{step}(P, \text{comp}(P, k, \sigma(a_1, \ldots, a_n)))_m[r]$ is in $\langle a_1, \ldots, a_n \rangle_A$. Thus we conclude $\text{comp}(P, n, \sigma)_m[r] \in \langle a_1, \ldots, a_n \rangle_A$.

By inspection of both cases

$$\text{comp}(P, n, \sigma)_m[r] \in \langle a_1, \ldots, a_n \rangle_A$$

for any $r \in R$. $\blacksquare$

5.4.6 Input/Output Semantics

In this section we analyse the properties of the state of the SERM after a computation has been completed and a halt state returned. During such a computation, a finite series of states

$$\sigma_0, \ldots, \sigma_h$$

is produced where σ_0 is the initial state, and $\sigma_h = (0, \gamma')$ is a halt state. To find a minimal h such that $\sigma_h = (0, \gamma')$, we introduce the *minimalisation* (or search) operator [23](p.43) defined by

Definition 5.4.7 *For any function* $f : \mathbb{N} \rightarrow \mathbb{N}$

$$\mu h(f(h) = 0) = \begin{cases} \textit{the least } h \textit{ such that} & \textit{(i) } f(k) \textit{ is defined, all } k \leq h \textit{ and} \\ & \textit{(ii) } f(h) = 0 \textit{ if such a } h \textit{ exists,} \\ \textit{undefined} & \textit{if there is no such } h \end{cases}$$

We use μ as a minimalisation operator to find a suitable h such that the given condition is satisfied. in this case, the condition is when the function f returns 0. This number is the least h such that the values $f(0), \ldots, f(h-1)$ are all defined with $f(0) \neq 0, \ldots, f(h-1) \neq 0$ and $f(h) = 0$.

Using the minimalisation operator, we define the length function that returns the minimal number of steps needed to reach a halt state.

Definition 5.4.8 *Let P be a* SERM *program, σ state, and $h \in \mathbb{N}$. Then* length : SERM(Σ) $\times$ State(A) $\rightarrow \mathbb{N}$ *is defined as*

$$\text{length}(P, \sigma) \simeq (\mu h)[\text{comp}(P, h, \sigma) = (0, \gamma')]$$

if a minimal h exists, undefined otherwise.

We now define the run function which computes a SERM program in the minimal amount of steps necessary.

Definition 5.4.9 (The run Function) *Let P be a* SERM(Σ) *program and σ be a state. Let* run : SERM(Σ) $\times$ State(A) $\rightarrow$ State(A) *be a function such that*

$$\text{run}(P, \sigma) \simeq \text{comp}(P, \text{length}(P, \sigma), \sigma).$$

Since for some input pair (P, σ) the final state may not exist, we use the symbol $\simeq$ to denote equality if, and only if, both states are defined and equal.

Having defined the meaning of computing one step and several steps in sequence, and proving that the computation must yield data values that exists in the subalgebra generated by the input, we now prove a corollary to Theorem 5.4.1.

This result gives an important property of the data values produced by a SERM computation. That is, any output of the SERM program must be generated by the Σ-operations on the input. We have

Corollary 5.4.1 (The Output of run)
Let P be a SERM(Σ) *program. Given an initialised state $\sigma(a_1,\dots,a_n)$ we have* $\text{run}(P,\sigma(a_1,\dots,a_n))_m[z] \in \langle a_1,\dots,a_n\rangle_A$ *for any output register z.*

Proof Set $k \simeq \text{length}(P,\sigma(a_1,\dots,a_n))$. By Definition 5.4.9 we have

$$\text{run}(P,\sigma(a_1,\dots,a_n)) \simeq \text{comp}(P,k,\sigma(a_1,\dots,a_n)).$$

Thus by The Local Computation Theorem 5.4.1

$$\text{run}(P,\sigma(a_1,\dots,a_n))_m[z] \in \langle a_1,\dots,a_n\rangle. \qquad \blacksquare$$

This result may be interpreted by saying that any output data we produce as a result of a SERM computation must always be present in the subalgebra generated by the input.

5.4.6.1 SERM Computable Functions

We use the run function to define SERM computability of functions on the Σ-algebra A.

Definition 5.4.10 (SERM Computable Functions) *A function $f : A_s^n \to A_s^m$ is computable on A by a* SERM *program P if for all input $a_1,\dots,a_n \in A_s$*

$$f(a_1,\dots,a_n) = \text{run}(P,\sigma(a_1,\dots,a_n))_m[z_1,\dots,z_m]$$

for the output registers $z_1,\dots,z_m$ specified in the program declaration.
It is SERM *computable if it is computable on A by some* SERM *program.*

We have the immediate Corollary

Corollary 5.4.2
Let $f : A_s^n \to A_s^m$ be a SERM *computable function over A. Then*

$$f(a_1,\dots,a_n) \in \langle a_1,\dots,a_n\rangle_A$$

for input $a_1,\dots,a_n$ in A_s^n.

Proof Since f is SERM computable, there exists a $P \in \text{SERM}(\Sigma)$ such that $f(a_1,\dots,a_n) = \text{run}(P,\sigma(a_1,\dots,a_n))$, and by Corollary 5.4.1 we have the result. $\blacksquare$

5.5 **Parallel** SERM

So far we have described a model of computation called a SERM machine. We have introduced the notion of SERM programs and formally defined the changes of the state of the machine whist a program is being executed. It is useful to consider the SERM as an algebra by grouping the states, register space and the computing operations step and comp as follows:

Signature	SERM_Σ
Import	Σ
Sorts	R, SERM(Σ), State, *nat*
Operations	step : SERM(Σ) $\times$ State $\to$ State comp : SERM(Σ) $\times$ State $\times$ *nat* $\to$ State run : SERM(Σ) $\times$ State $\to$ State

Algebra	SERM_A
Import	A
Carriers	R, SERM(Σ), State(A), $\mathbb{N}$
Operations	step : SERM(Σ) $\times$ State(A) $\to$ State(A) comp : SERM(Σ) $\times$ State(A) $\times$ $\mathbb{N}$ $\to$ State(A) run : SERM(Σ) $\times$ State(A) $\to$ State(A)

Now given any discrete space X we can use the algebraic methods of pointwise lifting (Chapter 4) to construct a SERM computing data at each point in the space. Such SERM computations can be thought of as *parallel computations* across the space X.

Algebra	$F(X, \text{SERM}_A)$
Import	SERM_A
Carriers	$F(X, R)$, $F(X, \text{SERM}(\Sigma))$, $F(X, \text{State}(A))$, $F(X, \mathbb{N})$
Operations	$\text{Step} : F(X, \text{SERM}(\Sigma)) \times F(X, \text{State}(A)) \to F(X, \text{State}(A))$ $\text{Comp} : F(X, \text{SERM}(\Sigma)) \times F(X, \text{State}(A)) \times F(X, \mathbb{N}) \to F(X, \text{State}(A))$
Definitions	$\text{Step}(P, \sigma)(x) = \text{step}(P(x), \sigma(x))$ $\text{Comp}(P, \sigma, n)(x) = \text{comp}(P(x), \sigma(x), n(x))$ $\text{Run}(P, \sigma)(x) = \text{run}(P(x), \sigma(x))$

Each of the carriers of the algebra SERM_A is pointwise lifted to the space X. That is, for each point $x \in X$ there exists a machine with its own separate set of registers, state (program counter and register configuration), time (the current *step* of the computation), and program. The operations step, comp and run are then pointwise lifted so that given x in X we can compute a SERM program on the data located at that point in space.

5.5.1 A Programming Example

Extending the example given in Section 5.4.3, we show how one can use the parallel SERM to compute over data distributed in the space X. That is, for each point $x \in X$ we compute a program using the operations available to us in the Σ_{nat}-algebra $\mathbb{N}$ of the natural numbers. Such a distribution of programs has the form

$$P : X \to \text{SERM}(\Sigma_{nat})$$

where $P(x)$ is the SERM program that is to be run using the data at the point x. The programs which we run are executed *independently* at each point in space, and in general different points will have different programs running.

Now suppose we are given two spatial objects $\phi_1, \phi_2 \in F(X, \mathbb{N})$ for which we want to compute the sum $\phi_1 + \phi_2$. Such an addition operation on spatial objects is simply lifted pointwise from addition on $\mathbb{N}$ as

$$(\phi_1 + \phi_2)(x) = \phi_1(x) + \phi_2(x),$$

for a point $x \in X$. To compute this function, we will execute a parallel SERM to add the natural numbers $\phi_1(x)$ and $\phi_2(x)$ for each point x in space. In this case the distribution P is a constant function such that

$$P(x) = P_{add} \text{ for all } x \in X,$$

where P_{add} is the following SERM program code:

```
Program: $P_{add}$
D. $I = x_1, x_2;\ O = z_1;\ S = r_1, r_2$
1. $x_1 := a_1$
2. $x_2 := a_2$
3. $r_1 := x_1$
4. $r_2 := x_2$
5. $Jump_=(r_2, zero, 9)$
6. $r_1 := succ(r_1)$
7. $r_2 := pred(r_2)$
8. $Jump_=(r_1, r_1, 5)$
9. $z_1 := r_1$
```

Each point in the space X has its own SERM computational state, and during the execution of P_{add} on a point in space this computational state is changed. These states are distributed by a mapping

$$\overline{\sigma} : X \to \text{State}(\mathbb{N})$$

where $\overline{\sigma}(x) = \sigma_x \in \text{State}(\mathbb{N})$ is the computational state of the SERM computing over the data at the point $x \in X$. A parallel SERM computation is begun by running a program beginning at an initial state. We have

$$\overline{\sigma}(\phi_1, \phi_2) = \sigma_x(\phi_1(x), \phi_2(x))$$

meaning that for each x the input data will be $\phi_1(x), \phi_2(x)$ and they are copied to the registers x_1, x_2 as specified by the program declaration of P_{add}.

The final state at each point after the computation has terminated (if it terminates) is

$$\begin{aligned}\overline{\sigma}'(x) &\cong \text{Run}(P, \overline{\sigma}(\phi_1, \phi_2))(x)\\ &\cong \text{run}(P_{add}, \sigma_x(\phi_1(x), \phi_2(x))).\end{aligned}$$

That is, an individual SERM will execute, in parallel, the programs P_{add} on each point in X. The resulting distribution $\overline{\sigma}'$ of states contains at every point x the sum $\phi_1(x) + \phi_2(x)$.

5.6 Notes and Sources

The topology of finite approximation used here is known in Classical Computability Theory as Baire space topology on maps

$$f : \mathbb{N} \to \mathbb{N}$$

and as Cantor space topology on maps

$$f : \mathbb{N} \to [0,1]$$

The development of the topology for general spatial objects over discrete space $F(X, A)$ and ideas of topological data types of these spatial objects seems to be new and is my own work, based on suggestions of Professor Tucker.

The work on the case study of register machines is my own work based on generalising the machine in Cutland [23] and using the methods of Tucker and Zucker [59]. These methods are also found in [29].

Chapter 6

Spatial Objects Over Continuous Space

6.1 Introduction

In this chapter we consider function spaces of *continuous* functions where X is an arbitrary topology and A is a topological Σ-algebra, as defined in Definition 3.9.3 (Preliminaries Section 3.9.3).

We define the set

$$C(X,A) = \{\phi \ \mid \phi : X \to A \text{ is continuous and total}\}$$

and give some basic algebraic properties in Section 6.5.

This chapter begins the development of the topological framework that is at the center of the general theory of spatial data types. Namely, we introduce the *compact-open topology*. We have chosen this topology because of it's well understood properties and their useful application to our theory.

The primary goal of this chapter is to develop a "language" in which we can use to compare two spatial objects, with the development of approximation methods being our fundamental motivation. That is, we will use the compact-open topology to describe the "nearness" of spatial objects.

The outline of the chapter is as follows. Section 6.3 defines the compact-open topology on the set $C(X,A)$. Here we give several examples motivated by applications of spatial objects, particularly to Constructive Volume Geometry. Section 6.4 defines two operations. The *composition* and *evaluation* operators over the set of continuous spatial objects are defined and conditions are given on $C(X,A)$ to ensure their continuity.

Later, we consider the operation of substitution and show that continuity for this operator is impossible (except in the case of the *discrete topology*).

In the particular instance of the compact-open topology on $C(X,\mathbb{R})$ where $A = \mathbb{R}$, we show the continuity of the operations of addition, multiplication and scalar multiplication of spatial objects.

6.2 Continuous Spatial Objects

In this chapter we move towards a general theory in which we consider topological spaces on X. Immediately some useful examples come to mind:

Example 6.2.1 (Euclidean Space) [26](p.64) Spatial objects in Constructive Volume Geometry are defined in Euclidean three dimensional space and therefore have $X = E^3$. The notation E^3 is used to denote the set $\mathbb{R}^3 = \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ equipped with the *Euclidean metric.* This metric is defined by

$$|x| = \sqrt{x_1{}^2 + x_2{}^2 + x_3{}^2}$$

at the point $x = (x_1, x_2, x_3) \in X$. Denoting the ball centered on the point x and radius $r > 0$ as $B(x,r) = \{y \mid |x-y| < r\}$, the basis for the topology on X is

$$\mathscr{B} = \{B(x,r) \mid x \in X, r > 0\}.$$

Example 6.2.2 (The Standard Topology of the Real Numbers)
Let $X = \mathbb{R}$ have the standard topology of the real numbers where a set $U \subset \mathbb{R}$ is open if for each $x \in U$ we can find a $r > 0$ such that $B(x,r) = \{y \mid |x-y| < r\}$ is a ball around x and $B(x,r) \subset U$.

Example 6.2.3 (Discrete Space)
We have seen in the previous chapter where X has the discrete topology. That is, each subset of X is an open set. Typical examples of sets with the discrete topology are $\mathbb{N}$ and $\mathbb{Z}$.

Thus, with both X and A having topologies we are able to consider the set of *continuous* spatial objects

$$C(X,A) = \{\phi \mid \phi : X \to A \text{ is continuous and total}\}.$$

6.3 The Compact-Open Topology on $C(X,A)$

Let X and A be topological spaces and $C(X,A)$ the set of all total continuous functions mapping X to A. This section gives a detailed description of the specification of the compact-open topology on the set $C(X,A)$.

Let K be a subset of X and U a subset of A. We write

$$W(K,U)$$

to denote the set of all *continuous* functions ϕ in $C(X,A)$ that take the points in K and map them to the points in U. In symbols we write this as

$$\phi[K] \subseteq U.$$

Now using this notation, consider the sets $W(K, U)$ such that $K \subseteq X$ is compact and $U \subseteq A$ is open. Using these sets, we construct the compact-open topology which we denote $\mathscr{T}$, on the set of total continuous functions $C(X, A)$.

Proceeding as shown in [36](p.221) we have the subbasis of the compact-open topology consists of all subsets

$$W(K, U)$$

where $K \subseteq X$ is compact and $U \subseteq A$ is open, and the basis $\mathscr{B}$ is defined as the family

$$\mathscr{B} = \left\{ \bigcap_{i=1}^{n} W(K_i, U_i) \mid K_i \text{ compact, } U_i \text{ open} \right\}$$

and then the family

$$\mathscr{T} = \left\{ \bigcup_{j \in J} B_j \mid B_j \in \mathscr{B} \right\}$$

of sets defines the compact-open topology on $C(X, A)$.

Throughout the text we refer to the compact-open topology $(C(X, A), \mathscr{T})$ as just $C(X, A)$, dropping the explicit indication of the collection of open sets $\mathscr{T}$ defining the topology.

We now turn to some examples illustrating the compact-open topology on $C(X, A)$ with various topologies on X and A.

Example 6.3.1 Let $X = \mathbb{R}$ and $A = \mathbb{R}$. Then we have the compact-open topology on $\mathscr{C}(\mathbb{R}, \mathbb{R})$ where a typical example of a subbasic set would be

$$W([-n, n], (a, b))$$

where $[-n, n]$, $n > 0$ is a closed (and compact) interval on $\mathbb{R}$ and $V = (a, b)$ is an open interval of $\mathbb{R}$ for $a > b$.

Example 6.3.2 Let $X = E^3$ and A $= \mathbb{R}^3$, where E^3 is the topology induced by the Euclidean metric (Example 6.2.1) and A is the product topology, with open sets of the form $V = U_1 \times U_2 \times U_3$ for open sets $U_i \subseteq \mathbb{R}$.
A typical example of a subbasic set for the compact-open topology on $\mathscr{C}(E^3, \mathbb{R}^3)$ then would be

$$W([-n, n]^3, V)$$

where $[-n, n]^3$ is a closed "cube" of E^3, for $n > 0$ and V is some open set in the product topology $\mathbb{R}^3$.

Example 6.3.3 (CVG The 4-Colour Channel Model) Let X be the Euclidean space E^3 with the standard topology and A is the product topology over the set $[0,1] \times \mathbb{R}^3$ whose values can be interpreted to represent opacity, red, green and blue.

The open sets of A are of the form $I \times U_1 \times U_2 \times U_3$ for an open set $I \subseteq [0,1]$ and open sets $U_1, U_2, U_3 \subseteq \mathbb{R}$.

An example of a subbasic set for the compact-open topology on $\mathscr{C}(E^3, [0,1] \times \mathbb{R}^3)$ would be

$$W([-n,n]^3, V)$$

where $[-n,n]^3$ is a closed cube in E^3, $n > 0$ and V is a typical open set in the product topology of A.

Example 6.3.4 Let $X = \mathbb{N}$ be the discrete topology and A has any topology. By the discreteness of X, every function $\phi : \mathbb{N} \to A$ is continuous and so we have $\mathscr{C}(\mathbb{N}, A) = F(\mathbb{N}, A)$.

6.4 General Properties of the Compact-Open Topology

It is convenient now to discuss some general properties of function spaces with the compact-open topology. We look at two operators that will be used extensively in our work: *composition* and *evaluation.*

Our exposition in this section follows closely to [26](p.259) and we develop a general theory by introducing machinery which allow us to consider these operations on arbitrary topological spaces X, Y, Z. We also use the function spaces of continuous functions $C(X,Y)$, $C(Y,Z)$ and $C(X,Z)$ each with the compact-open topology.

6.4.1 Composition

We define an operator which, given two functions from different function spaces $C(X,Y)$ and $C(Y,Z)$ returns the composite function in $C(X,Z)$. We show the continuity of this operator when one function parameter is fixed, and give conditions in which it is continuous in both functions. We define the property of *joint continuity* as

Definition 6.4.1 (Joint Continuity) *[36](p.223) A topology for $C(X,Y)$ is said to be jointly continuous if, and only if, the map $P : C(X,Y) \times X \to Y$ defined for a function $f \in C(X,Y)$ and $x \in X$ by*

$$P(f,x) = f(x)$$

is continuous.

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "(g \circ f)"'] & Y \arrow[d, "g"] \\
 & Z
\end{tikzcd}$$

Figure 6.1: Composition of functions

In a language suited more for our uses, joint continuity is in essence a statement about the continuity of the evaluation of a function f in $C(X,Y)$ at point $x \in X$.

Definition 6.4.2 (Composition) *The composition operator comp* : $C(X,Y) \times C(Y,Z) \to C(X,Z)$ *is defined as*

$$comp(f,g) = g \circ f$$

for functions $f : X \to Y$ *and* $g : Y \to Z$.

We describe this definition pictorially in Figure 6.1.

An important question that we can ask about the compact-open topology is what conditions are needed for operation *comp* to be continuous in both variables. To this end, we recall from the Preliminaries Section 3.8.2 that each point in a locally compact Hausdorff space has a relatively compact nbhd. That is, the closure of this nbhd is compact. We state and prove an adaptation of a theorem found in [26](p.260), which gives the criteria for a function space to be jointly continuous.

Theorem 6.4.1 *[26](p.260 – 261) Let X and Z be Hausdorff and Y locally compact. Then the map comp* : $C(X,Y) \times C(Y,Z) \to C(X,Z)$ *is continuous.*

PROOF Given functions f in $C(X,Y)$ and g in $C(Y,Z)$, let $W(K,U)$ be a nbhd of $(f \circ g)$, K compact, U open.

By continuity of g, $g^{-1}[U]$ is open and by the continuity of f, $f[K] \subset g^{-1}[U]$ is compact.

By Theorem 1.4(1) of [26](p.224) [1] the image $f[K]$ of compact K is compact in Y. By the local compactness of Y there is a relatively compact open set $V \subset Y$ with $f[K] \subset V \subset \overline{V} \subset g^{-1}[U]$, and so $comp[W(K,V), W(\overline{V},U)] \subset W(K,U)$. ■

[1]Theorem 1.4(1) states: The continuous image of a compact set is compact.

Composition With a Fixed Parameter

In many applications, we may not need such a strong composition result. Thus we now prove a theorem which shows that *comp* is always continuous whenever one of the parameters is fixed. This result is taken from [26](p.259).

Theorem 6.4.2 (Composition of Functions) *The operation comp is continuous when one of the parameters is fixed. Namely,*

1. *the function $g \circ f_1$ is continuous for fixed f_1 and*
2. *the function $g_1 \circ f$ is continuous for fixed g_1.*

PROOF (1). We fix a function $f_1 : X \to Y$. Let $W(K, U)$ be a subbasic set in $C(X, Z)$. We observe

$$g \circ f_1 \in W(K,U) \iff g \in W(f_1[K], U).$$

Since $f_1 : X \to Y$ is continuous, $f_1[K]$ is a compact set in Y, and so $W(f_1[K], U)$ is an open set of $C(Y, Z)$. Therefore

$$f_1 \times W(f_1[K], U) \subset C(X, Y) \times C(Y, Z)$$

is open and $comp[f_1, W(f_1[K], U)] = W(K, U)$ which proves the continuity of $g \circ f_1$.

(2). Let $W(K, U)$ be a subbasic set in $C(X, Z)$ We fix a function $g_1 : Y \to Z$ and note that

$$g_1 \circ f \in W(K, U) \text{ and } f \in W(K, g_1^{-1}[U])$$

where $g^{-1}[U]$ is an open set in Y. Therefore

$$W(K, g_1^{-1}[U]) \times g_1 \subset C(X, Y) \times C(Y, Z)$$

is open, and so $comp[W(K, g_1^{-1}[U]), g_1] = W(K, U)$. Therefore $g_1 \circ f$ is continuous. ∎

6.4.2 Evaluation

For any particular function in a function space $C(X, Y)$ we will need to evaluate the function $f \in C(X, Y)$ at a point x in X. This is done by way of the operator

$$eval(f, x) = f(x)$$

which given a function f and a point $x \in X$ returns the result $f(x)$ in the set Y. We have the definition

Definition 6.4.3 *The operation eval : $F(X, Y) \times X \to Y$ is defined for a function $f \in F(X, Y)$ and a point $x \in X$ by*

$$eval(f, x) = f(x).$$

We will adapt the operator *comp* in the previous section and use it's continuity result for a continuity proof of *eval*. In fact, we show that evaluation is simply a special case of composition.

To this end, let $\{z\}$ be the singleton set containing one point z of Z. Consider the set

$$C(z,Y) = \{f \in C(Z,Y) \mid f^{-1}[Y] \subseteq \{z\}\}$$

of all functions that map the point z to data in Y.

The proof of the composition operator has given the conditions needed to ensure the continuity of *comp* on both parameters. Now, to show the special case of evaluation, consider the function $j_Y : Y \to C(z,Y)$ that maps the elements of Y to the functions in $C(z,Y)$ such that $j_Y(y)$ maps to a function f in $C(z,Y)$ where $f(z) = y$. Recalling Definition 3.9.1 of a homeomorphism in the Preliminaries Section 3.8, we show

Lemma 6.4.1 *The function $j_Y : Y \to C(z,Y)$ is a homeomorphism.*

Proof This mapping is a homeomorphism between Y and $C(z,Y)$. That is, a bijective bicontinuous function. Let $y_1, y_2 \in Y$. To prove injectiveness

$$\begin{aligned} j_Y(y_1)(z) = j_Y(y_2)(z) &\implies [f_1(z) = f_2(z)] \\ &\implies y_1 = y_2. \end{aligned}$$

For surjectivity simply note that for all $f \in C(z,Y)$ we can choose the value $y = f(z)$ and then we have $j_Y(y) = f$, and so $Ran(j_Y) = C(z,Y)$.

For the continuity of j_Y note for open set $U \subset Y$ that $j_Y^{-1}[W(z,U)] = U$ by the definition of j_Y, and for the continuity of j_Y^{-1} note that for open set $V \subset Y$: $j_Y[V] = W(z,V)$. ■

We define a similar map $j_X : X \to C(z,X)$ where $j_X(z)$ maps to a function f in $C(z,X)$ such that $f(z) = x$. This function is also a homeomorphism and the proof is identical to the one for Lemma 6.4.1.

$$\begin{tikzcd}
C(X,Y) \arrow[d, "id"] & \times & X \arrow[r, "eval"] \arrow[d, "j_X"] & Y \arrow[d, "j_Y"] \\
C(X,Y) & \times & C(z,X) \arrow[r, "comp"'] & C(z,Y)
\end{tikzcd}$$

Figure 6.2: Commutative Diagram For Evaluation

Finally, we are able to prove a result which gives the necessary conditions for the continuity of the evaluation function on the compact-open topology:

Theorem 6.4.3 *If Y is locally compact then the evaluation map*

$$eval : C(X,Y) \times X \to Y$$

is continuous.

PROOF Consider the special case of composition

$$comp : C(X,Y) \times C(z,X) \to C(z,Y). \tag{6.1}$$

By Theorem 6.4.1, this function is continuous since X is locally compact. By Lemma 6.4.1 we have the homeomorphisms $X \cong C(z,X)$ and $Y \cong C(z,Y)$. Thus we can rewrite (6.1) as $comp : C(X,Y) \times X \to Y$. But this is just the definition of the evaluation function, hence the result. ■

We illustrate this theorem with an example.

Example 6.4.1 Suppose $X = \mathbb{N}$ has the discrete topology and $A = \mathbb{R}$ has the standard topology on the real numbers. The evaluation function will be

$$[-] : C(\mathbb{N},\mathbb{R}) \times \mathbb{N} \to \mathbb{R}$$

where

$$f[i] = r$$

for $f \in C(X,A)$, $i \in \mathbb{N}$ and a number $r \in \mathbb{R}$.

This example can be thought of as a "read" operation on arrays of real numbers; where the arrays are modelled by the function $f : \mathbb{N} \to \mathbb{R}$, and indexed by the set of natural numbers.

6.4.3 Covariant and Contravariant Operations

We have already seen many examples of operations that transform data and operations that transform space. We prove two lemmas that show the continuity of such operations when lifted pointwise.

Lemma 6.4.2 (Continuity of Covariant Operations) *Let $\alpha : Y \to Z$ be a continuous mapping and define $\overline{\alpha} : C(X,Y) \to C(X,Z)$ pointwise at $x \in X$ by*

$$\overline{\alpha}(\phi)(x) = \alpha(\phi(x))$$

for the continuous function $\phi \in C(X,Y)$. Then $\overline{\alpha}$ is continuous.

PROOF We have the equality

$$\overline{\alpha}(\phi) \quad = \quad \alpha \circ \phi.$$

Applying Theorem 6.4.2(2) yields the result. ■

Lemma 6.4.3 (Continuity of Contravariant Operations)
Let $\beta : X \to Y$ be a continuous function and define $\overline{\beta} : C(X,Z) \to C(Y,Z)$ pointwise at $x \in X$ by

$$\overline{\beta}(\phi)(x) = \phi(\beta(x)),$$

for continuous $\phi \in C(Y,Z)$. Then $\overline{\beta}$ is continuous.

Proof By the definition, $\overline{\beta}$ is simply the composition

$$(\phi \circ \beta),$$

to which we simply apply Theorem 6.4.2(2). ■

6.4.4 The Discrete Topology on X

6.4.4.1 Compactness

In this section we prove a result that gives necessary and sufficient conditions for the compactness of a discrete topology on X.

Theorem 6.4.4 *Let the set X have the discrete topology, and A has some topology T_A. Then*

(i) all maps $X \to A$ are continuous, and

(ii) $S \subseteq X$ is compact if, and only if, S is finite.

Proof (i) Let f be a function in $F(X,A)$ and let U be any open set of T_A. Since X has the discrete topology every subset of X is an open set. Thus each preimage $f^{-1}[U]$ of the open set U is an open set in X, hence f is continuous.

(ii) $\Rightarrow$ Let $x_1, \ldots, x_i, \ldots$ be all the elements of the set S. Consider the collection

$$C = \{\{x_i\} \mid x_i \in S\}.$$

The collection C is an open cover since each $\{x_i\}$ is an open set and clearly each point $x_i \in S$ is contained in one open set in C. By the compact property of S there exists a finite subcover

$$C_0 = \{\{x_i\}, \ldots, \{x_k\}\}$$

of C which covers S. Hence each element of S is contained in C_0 so S is finite.

$\Leftarrow$ Assume that S is finite. We must show that each open cover of S has a finite subcover. Let

$$C = \{U_1, \ldots, U_i, \ldots\}$$

be a typical open subcover of S. Then for each element $x_i \in S$, choose a $U_i \in C$ such that $x_i \in U_i$. We construct the collection

$$C_o = \{U_1, \ldots, U_k\}$$

where each x_i is covered. Since there are finitely many x_i, we need only a finite number of U_i, yielding a finite subcover. ■

Theorem 6.4.5 *Let X be a discrete space. The space A is compact if, and only if, $F(X, A)$ is compact.*

We note that Theorem 6.4.5 is actually a special case of a more general theorem. We state Tychonoff's Theorem as

Theorem 6.4.6 *[26](p.224) Let $\{A_i \mid i \in I\}$ be any family of spaces. Then the product*

$$\prod_{i \in I} A_i$$

is compact if, and only if, each A_i is compact.

By our remarks in Example 4.2.7 of Chapter 4, when X is discrete, $F(X, A) \simeq A^X$. Hence by Tychonoff's Theorem, Theorem 6.4.5 is immediately true.

6.5 The Topological Algebra of $C(X, A)$

In this section we consider the case where X has a topology and A is a topological Σ-algebra. Recall Definition 3.9.3 in Section 3.9.3 of the Preliminaries. We reproduce the definition here

Definition 6.5.1 (Topological Σ-Algebras)
A topological Σ-algebra is a Σ-algebra with topologies on the carriers such that each of the basic Σ-operations is continuous.

A natural question to ask is whether or not the pointwise lifted operations over $C(X, A)$ is a topological Σ-algebra, with respect to the compact-open topology. To this end, we prove

Lemma 6.5.1 *$C(X, A)^n$ is homeomorphic to $C(X, A^n)$.*

PROOF To prove this result, we define the function $\Phi : C(X, A)^n \to C(X, A^n)$ as

$$\Phi(\phi_1, \ldots, \phi_n)(x) = (\phi_1(x), \ldots, \phi_n(x)),$$

for (continuous) functions $\phi_1, \ldots, \phi_n$ in $C(X, A)$ and a point x in X. We must show the function Φ is bicontinuous and bijective.
Bicontinuity Let $W(K, U_1 \times \cdots \times U_n)$ be a basic open set of $C(X, A^n)$. Then by the definition of Φ we have

$$\Phi^{-1}[W(K, U_1 \times \cdots \times U_n)] = W(K, U_1) \times \cdots \times W(K, U_n),$$

open in $C(X, A)^n$.
To prove continuity in the other direction, let $W(K, U_1) \times \cdots \times W(K, U_n)$ be a basic open set of $C(X, A)^n$. Then by definition

$$\Phi[W(K, U_1) \times \cdots \times W(K, U_n)] = W(K, U_1 \times \cdots \times U_n),$$

which is a basic open set in $C(X, A^n)$.
Bijectivity is obvious from Lemma 5.3.2. ∎

Theorem 6.5.1 (Pointwise Lifting of Topological Algebras)
Let X be a topological space, A a topological Σ-algebra. The pointwise lifted functions of A are continuous on the compact-open topology over $C(X, A)$. That is, $C(X, A)$ is a topological Σ-algebra.

PROOF We list the Σ-algebra $C(X, A)$ as follows

Algebra	$C(X, A)$
Carriers	$C(X, A)$
Constants	$\ldots, c_{C(X,A)} :\to C(X, A), \ldots$
Operations	$\ldots, f_{C(X,A)} : C(X, A) \times \cdots \times C(X, A) \to C(X, A), \ldots$
Definitions	$c_{C(X,A)}(x) = c_A$
	$f_{C(X,A)}(\phi_1, \ldots, \phi_n)(x) = f_A(\phi_1(x), \ldots, \phi_n(x))$

We show that each operation is continuous with respect to the compact-open topology on $C(X, A)$. To this end, we observe that given a typical operation $f_{C(X,A)}$ we can express the pointwise lifting at $x \in X$ in terms of a composition

$$(f_{C(X,A)})(\phi_1, \ldots, \phi_n) \quad = \quad (f_A \circ \Phi(\phi_1, \ldots, \phi_n))$$

for input values $\phi_1, \ldots, \phi_n$ in $C(X, A)$. But by the continuity of the function Φ in Lemma 6.5.1 and Theorem 6.4.2 gives us the result. ∎

We immediately have a useful corollary for the case of the real numbers. Let $\mathbb{R}$ be the topological Σ-algebra

Algebra	$\mathbb{R}$
Carriers	$\mathbb{R}$
Operations	$+, \cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
	$- : \mathbb{R} \to \mathbb{R}$

where the carrier set $\mathbb{R}$ has the standard topology and let X have a topology. We show the pointwise lifting of the operations on $\mathbb{R}$ to the set

$$C(X,\mathbb{R}) = \{\phi \mid \phi : X \to \mathbb{R}\}$$

forms a topological Σ-algebra, where all operations are continuous with respect to the compact-open topology on $C(X,\mathbb{R})$. We list the Σ-algebra as follows.

Algebra	$C(X,\mathbb{R})$
Carriers	$C(X,\mathbb{R})$
Operations	$+, \times : C(X,\mathbb{R}) \times C(X,\mathbb{R}) \to C(X,\mathbb{R})$
	$- : C(X,\mathbb{R}) \to C(X,\mathbb{R})$
Definitions	$(f+g)(x) = f(x) + g(x)$
	$(f \times g)(x) = f(x) \cdot g(x)$
	$-(f)(x) = -f(x)$

Corollary 6.5.1 *Let f and g be functions in $C(X,\mathbb{R})$, and $\lambda \in \mathbb{R}$. The algebraic operations of addition $(f+g)$, multiplication $(f \times g)$, and scalar multiplication $(\lambda \cdot f)$ pointwise lifted from $\mathbb{R}$ form a topological Σ-algebra on $C(X,\mathbb{R})$*

PROOF Immediate from Theorem 6.5.1. ■

Theorem 6.5.2 *Let X be a topology and A a topological Σ-algebra. The Σ-algebra $C(X,A)$ is a Σ-subalgebra of $F(X,A)$.*

PROOF Clearly $C(X,A)$ is a subset of $F(X,A)$. Now for each constant function c_F in $F(X,A)$ we have $c_F(x) = c_A$ for all $x \in X$ by definition. But every constant function is continuous and $c_C(x) = c_F(x) = c_A$, thus c_C is continuous. Now choose any Σ-operation $f_F : F(X,A)^n \to F(X,A)$ and any continuous functions $\phi_1, \ldots, \phi_n$ from the set $C(X,A)$. We want to show that f_C is closed on $C(X,A)$; i.e. for any continuous input, the output is continuous. By definition

$$\begin{aligned} f_F(\phi_1, \ldots, \phi_n) &= f_C(\phi_1, \ldots, \phi_n) \\ &= f_A(\phi_1(x), \ldots, \phi_n(x)) \end{aligned}$$

for all $x \in X$. But since A is a topological Σ-algebra, $f_A : A^n \to A$ is continuous. So for any input of continuous functions the operation f_C returns a continuous function, and therefore is closed on $C(X,A)$. ■

We will also discuss a substitution operator over the set which, when given a point $x \in X$, a datum $a \in A$ and a function $f : X \to A$. The operator returns a new function $f'(x) = a$ which remains unchanged at every other point. This operator is slightly more problematic than the others, for clearly $C(X, A)$ is not closed under substitution. That is, we can use this operator to build non-continuous spatial objects.

6.5.1 Substitution

The substitution operator is an operation used to replace a single data value of the spatial object $\phi : X \to Y$ at a point in x in X. We define an operation $sub : C(X, Y) \times X \times Y \to C(X, Y)$ by

$$sub(\phi, x, a)(y) = \begin{cases} a & \text{if } x = y, \\ \phi(y) & \text{otherwise.} \end{cases}$$

Example 6.5.1 Let $X = \mathbb{N}$ and $A = \mathbb{R}$. We can write

$$sub(\phi, i, r) \;\text{ as }\; \phi[i] := r$$

for $\phi \in \mathscr{C}(\mathbb{N}, \mathbb{R})$, $i \in \mathbb{N}$ and $r \in \mathbb{R}$.

This example can be viewed as an "update" operation for arrays of real indexed by the natural numbers; and would be a generally useful operation to use with our spatial objects. Some problems arise with its use however, as illustrated in this next example.

Example 6.5.2 Let $X = [0, 1]$ and $A = \mathbb{R}$. Define the constant function $\mathbf{0} : [0, 1] \to \mathbb{R}$ such that $\mathbf{0}(x) = 0$ for all x in $[0, 1]$. Clearly $\mathbf{0}$ is continuous and in $\mathscr{C}([0, 1], \mathbb{R})$. Pick any $x_0 \in [0, 1]$. Then $\psi = sub(\mathbf{0}, x_0, 1)$ is a discontinuous function.

This example shows that in general, the set $C(X, A)$ is not closed under the substitution operator. That is, discontinuous functions may be returned as a result of substituting values in spatial objects.

Remark 6.5.1 *Is it possible to define a substitution operator in which $C(X, A)$ is closed under? That is to say, can we somehow ensure that the function returned is always continuous.*

The root of the problem is the modification of a single value. If we can produce an operation that somehow changes a value, and then all local values are "modified" to accommodate this value, then perhaps a suitable operation can be defined. However we shall not explore this subject here, but consider it as a question for further research.

6.6 The Topology of Uniform Convergence

In this chapter we are setting up some mathematical machinery that will allow us to describe approximation. At the heart of approximation lies the notion of distance, and so in this section we define mathematically a common method of comparing distances between spatial objects.

Distances between spatial objects correspond with specifying a norm and we now look at the special case of the compact-open topology on the set of total continuous functions in $C(K, \mathbb{R})$ where $X = K$, a compact space and $A = \mathbb{R}$. The norm we use is the sup norm, or otherwise referred to in the literature as the norm of *uniform convergence*. Most textbooks outlining the compact-open topology construction, such as [11], [26], [27] and [52], give special treatment to this norm. It was also studied in the early papers of Arens [3] as an important special case of the compact-open topology.

In this section we define the sup norm on $C(K, \mathbb{R})$ and show that it satisfies the properties of a metric on $C(K, \mathbb{R})$. By defining a base using this norm, we are able to prove a theorem relating it with the compact-open topology. We use the sup norm $\| - \|_\infty : C(K, \mathbb{R}) \to \mathbb{R}$ to compare spatial objects, defined pointwise from the absolute function on the real numbers. For $f \in C(K, \mathbb{R})$ we have

$$\|f\|_\infty = \sup_{x \in K}\{|f(x)|\},$$

and to compare f to another spatial object g we simply evaluate

$$\|f - g\|_\infty = \sup_{x \in K}\{|(f(x) - g(x)|\}.$$

Furthermore, it is easy to show that the sup norm satisfies the properties of a metric space on the set $C(K, \mathbb{R})$, and thus this norm defines what we could intuitively describe as a measurement of "distance".

6.6.1 The Compact-Open Topology on $C(K, \mathbb{R})$

Continuing with the notion of distance, we define ϵ-spheres around spatial objects: For $\epsilon > 0$ we define the open ϵ-sphere

$$B(f, \epsilon) = \{g \in C(K, \mathbb{R}) \mid \|f - g\|_\infty < \epsilon\}$$

which specifies the family of functions that are within ϵ of f.

Now, the sup norm on $C(K, \mathbb{R})$ defines a metric, and therefore by Definition 3.8.17 defines a topology on $C(K, \mathbb{R})$. The basis for this induced topology according to Lemma 3.8.3 is the family

$$\mathscr{T}_\infty = \{B(f, \epsilon) \mid f \in C(K, \mathbb{R}), \epsilon > 0\}$$

of all ϵ-spheres.

We show that the induced topology from this metric is in fact the same as the compact-open topology on $C(K, \mathbb{R})$. That is, a set B is a basic open set in the compact-open topology $\mathscr{T}$ if, and only if, B is open in the topology induced by the sup norm $\mathscr{T}_\infty$. Thus we state and prove

Theorem 6.6.1 *The topology induced by the* sup *norm is the compact-open topology. That is,*

$$\mathscr{T}_\infty = \mathscr{T}$$

PROOF The proof is found in [26](p.271) and is as follows:

$\subset$ We show that given a nbhd $W(K, U)$ of a function $f \in C(K, \mathbb{R})$, an ϵ-nbhd $B(f, \epsilon)$ of f can be placed completely within $W(K, U)$.

Continuous functions preserve compactness, and so the image set $f[K]$ is compact. We know $f[K] \cap [\mathbb{R} \setminus U] = \emptyset$ and we put $\epsilon = d(f[K], [\mathbb{R} \setminus U])$, which is always positive because the sets are disjoint. This yields $f \in B(f, \epsilon) \subset W(K, U)$.

$\supset$ Conversely, we show that each ϵ-nbhd $B(f, \epsilon)$ contains an open set $W(K, U)$. Since K is compact, it can be covered by finitely many open sets $U_1, \ldots, U_n$ such that the greatest distance between two points in each image set $f[U_i]$, denoted $\delta(U_i)$ is less than $\frac{\epsilon}{4}$ for each $i = 1, \ldots, n$. Let V_i be an $\frac{\epsilon}{4}$-nbhd of the compact image $f(\overline{U}_i)$. We have $\delta(V_i) < \frac{3}{4}\epsilon$, and so $f \in \cap_1^n (\overline{U}_i, V_i) \subset B(f, \epsilon)$, completing the proof. ∎

We make some remarks on the density condition on $C(X, A)$ in terms of the compact-open topology and the sup norm.

In the language of the compact-open topology, D is dense in $\mathscr{C}(X, A)$ if for any subbasic set $W(K, U)$ of $C(X, A)$ we have

$$D \cap W(K, U) \neq \emptyset.$$

Now consider the sup topology over the set $C(K, \mathbb{R})$, where $X = K$ and $A = \mathbb{R}$ with the standard topology. In the context of the topology induced by the sup norm, D is dense in $C(K, \mathbb{R})$ if any spatial object can be approximated to an arbitrary degree of accuracy by objects in D. If we wish to approximate an object $\phi \in C(K, \mathbb{R})$ to within an arbitrary precision say, $\epsilon > 0$ using objects in D, then we find a $g \in D$ such that for all points $x \in X$

$$d(\phi(x), g(x)) = |\phi(x) - g(x)| < \epsilon.$$

In terms of the sup norm we write this as

$$\|\phi - g\|_\infty < \epsilon.$$

6.7 Equational Validity on $C(X, A)$

In this section we give a useful result which shows that it is sufficient for an equation to be true on a dense subset D of a topology A to be true on the entire space A. Then by the Validity Theorem 4.3.1, we prove that equational validity on a dense subset of A is suitable for the equations to be valid on the pointwise lifted algebra $C(X, A)$. We begin with

Lemma 6.7.1 *Let X have an arbitrary topology, A a Hausdorff space, and let $f, g : X \to A$ be continuous functions. If $D \subset X$ is dense and $f = g$ for all points in D then $f = g$ for all points in X.*

PROOF We follow closely Dugundji's proof [26](p.140), where the following lemma is given:

Sub Lemma 6.7.1 *Let X be arbitrary and Y be Hausdorff and $f, g : X \to Y$ be continuous. Then $\{x \mid f(x) = g(x)\}$ is closed in X.*

This lemma states that the set of points $U = \{x \mid f(x) = g(x)\}$ is closed in A. Now this closed set U contains every point x of D, but by the definition of density 3.8.14, we have $X = \overline{D} \subset \overline{U} = U$, and thus for each $x \in X$ $f(x) = g(x)$ completing the proof. ■

Theorem 6.7.1 *Let D be a dense subset of the topological Σ-algebra A. Suppose t and t' are terms in $T(\Sigma, Y)$. Then*

1. *$t = t'$ is valid in D if, and only if,*
2. *$t = t'$ is valid in A.*

PROOF Recall Definition 4.3.2 in Chapter 4, Section 4.3.3 that an equation is valid if, and only if,

$$\forall(a_1, \ldots, a_n)[[\![t]\!]_A(a_1, \ldots, a_n) = [\![t']\!]_A(a_1, \ldots, a_n)].$$

(1) $\Rightarrow$ (2). We define the functions $f : A^n \to A$ to be

$$f(a_1, \ldots, a_n) = [\![t]\!]_A(a_1, \ldots, a_n)$$

and $g : A^n \to A$ to be

$$g(a_1, \ldots, a_n) = [\![t']\!]_A(a_1, \ldots, a_n)$$

for all $a_1, \ldots, a_n \in A$. Then by Lemma 6.7.1 we have $f = g$ on $D \implies f = g$ on A.

(1) $\Leftarrow$ (2). This is obvious since if the equation $t = t'$ is valid on A, it is certainly valid on some subset D of A. ■

Theorem 6.7.2 *Let D be a dense subset of the topological Σ-algebra A and $C(X, A)$ the pointwise lifted algebra over an arbitrary topological space X. Suppose t and t' are terms in $T(\Sigma, Y)$. Then*

1. *$t = t'$ is valid in D if, and only if,*
2. *$t = t'$ is valid in $C(X, A)$.*

PROOF (1) $\Longrightarrow$ (2). Suppose $t = t'$ is valid in D. Then by Theorem 6.7.1 we have $t = t'$ is valid in A, and by the Validity Theorem 4.3.1 $t = t'$ is valid in the pointwise lifted algebra $C(X, A)$.

(2) $\Longrightarrow$ (1). Conversely, suppose $t = t'$ is valid on $C(X, A)$. Then again by the Validity Theorem 4.3.1 $t = t'$ is valid in A. But certainly if the equation is valid on all of A, then it is valid for a subset D of A. ∎

6.8 Inverse Limits of Topological Σ-Algebras

Section 6.6 of this chapter we have shown the equivalence of the compact-open topology and the topology induced by the sup norm, under the condition that our space $X = K$ is compact. It was stated that the motivation for such a characterisation of the compact-open was for the approximation of spatial objects. However, many of the examples we have already seen are for non-compact spaces such as the real numbers. In fact, our application of spatial objects to Constructive Volume Geometry uses the Euclidean space E^3, i.e. $\mathbb{R}^3$ equipped with the Euclidean metric. Thus in our general theory of abstract spatial data types, this compactness requirement is usually not feasible.

To remedy this situation, we define the algebraic construction of an *inverse system* over the family of compact subsets of the space X. We define a topology over this construction and Theorem 6.8.1 proves that it is indeed equivalent to the compact-open topology on $C(X, A)$.

6.8.1 A Topology on the Inverse Limit

Define the family

$$A = \{A_i \mid i \in I\}$$

of Σ-algebras where I is an index set. Supposing that each of the Σ-algebras in A is a Hausdorff space, Theorem 1.3 in [26](p.138) ensures that the product topology on $\prod A$ (Section 3.8.3 of preliminaries) inherits the Hausdorff property as well. Furthermore, the topology on $\varprojlim A$ is the *induced* or *subspace* topology of $\prod A$. Now recall from the Preliminaries Section 3.7.2 we defined the projection map restricted to the inverse limit as

$$\phi_i : \pi_i \mid \varprojlim A : \varprojlim A \to A_i.$$

We use this in the next lemma, reproduced directly from [26](p.428):

Lemma 6.8.1 *If I is a directed set, then the sets*

$$\{\phi_i^{-1}[U] \mid \textit{for all } i \in I \textit{ and for all open } U \subset A_i\}$$

form a basis for $\varprojlim A$.

Proof Let $x \in V$ where V is an open set in $\varprojlim A$. Since $\varprojlim A$ is a subset of $\prod_{i \in I} A_i$, there are finitely many indices $\alpha_1, \ldots, \alpha_n$ and open sets $U_{\alpha_1} \subset A_{\alpha_1}, \ldots, U_{\alpha_n} \subset A_{\alpha_n}$ such that

$$x \in \langle U_{\alpha_1}, \ldots, U_{\alpha_n} \rangle \cap \varprojlim A \subset V.$$

We want to show that for some suitable $i \in I$ and open set $U \subset A_i$

$$x \in \phi_i^{-1}[U] \subset \langle U_{\alpha_1}, \ldots, U_{\alpha_n} \rangle \cap \varprojlim A.$$

Because I is directed, we can choose an $i \in I$ such that $\alpha_1, \ldots, \alpha_n < i$ and then define

$$U = \bigcap_{j=1}^{n} {\phi_{\alpha_j}^{i}}^{-1}[U_{\alpha_j}]$$

which is open in the coordinate space A_i. We have

$$\begin{aligned}\phi_i^{-1}[U] &= \bigcap_{j=1}^{n} \left(\phi_i^{-1} \circ {\phi_{\alpha_j}^{i}}^{-1}\right)[U_{\alpha_j}] \\ &= \bigcap_{j=1}^{n} \phi_{\alpha_j}^{-1}[U_{\alpha_j}]\end{aligned}$$

so that an element $a \in \varprojlim A$ belongs to $\phi_i^{-1}[U]$ if, and only if, it's α_j-th coordinate is in U_{α_j} for each $j = 1, \ldots, n$. Hence we have $a \in \phi_i^{-1}[U] \subset \langle U_{\alpha_1}, \ldots, U_{\alpha_n} \rangle \cap \varprojlim A$. ∎

6.8.2 An Inverse System on the Compact Subsets of X

Let $\mathscr{K} = \{K \mid K \text{ is a compact subset of } X\}$ be the collection of compact sets of the Hausdorff space X. We prove a lemma that shows $\mathscr{K}$ forms a directed set under set inclusion denoted $\leq$ where we define the relation $I \leq J$ to mean $I \subseteq J$ for any pair of sets in $\mathscr{K}$.

Lemma 6.8.2 *The countable family $\mathscr{K}$ of all compact spaces in a Hausdorff space X where $X = \cup_{i \in \mathbb{N}} K_i$ forms a directed set under the relation $\leq$.*

Proof This proof is similar to [27](p.160). The family $\mathscr{K}$ is partially ordered since for any compact sets I,J and K we observe the properties of

reflexivity: $I \subseteq I$,

transitivity: $I \subseteq J$ and $J \subseteq K$ implies $I \subseteq K$ and

anti-symmetry: $I \subseteq J$ and $J \subseteq I$ implies $I = J$

from basic results in set theory. Now, given any two compact sets I and J in $\mathscr{K}$ we must exhibit a set K in $\mathscr{K}$ such that $I, J \subseteq K$. Simply define K to be the union

$$I \bigcup J.$$

By the compactness properties of I and J the union is a compact space and is in $\mathscr{K}$. It also has the desired property $I, J \subseteq K$, hence $\mathscr{K}$ is a directed set. ∎

Definition 6.8.1 *[26](p.12) Given a function $f : X \to A$ and a subset $I \subset X$, the map f considered only on I is called the restriction of f to I, is written f_I.*

We define $C(\mathscr{K}, A)$ to be the product $\prod_{K \in \mathscr{K}} C(K, A)$ of function spaces indexed by $\mathscr{K}$, where each coordinate function space has the compact-open topology. Thus, an element of $C(\mathscr{K}, A)$ is a tuple

$$(f) = f_{I_1} \times f_{I_2} \times f_{I_3} \times \cdots$$

of continuous functions, where each f_I is a function (or restriction) from the compact set I to the topological Σ-algebra on A, belonging to the coordinate space $C(I, A)$ (See Figure 6.3).

We focus our attention to a particularly useful subset of this product. From [27](p.160 – 161), we define an inverse system $S(X)$ over $C(\mathscr{K}, A)$ consisting of

(i) the directed set $\mathscr{K}$,

(ii) a $\mathscr{K}$-indexed set of Σ-algebras $\{C(K, A) \mid K \in \mathscr{K}\}$,

(iii) an indexed family of Σ-homomorphisms $\phi^I_J : C(I, A) \to C(J, A)$ for each $J \subseteq I$ such that $\phi^J_K \circ \phi^I_J = \phi^I_K$ for all $K \subseteq J \subseteq I$ and ϕ^I_I is the identity map.

The inverse limit $\varprojlim C(\mathscr{K}, A)$ of $S(X)$ is the set

$$\varprojlim C(\mathscr{K}, A) = \left\{(f) \in \prod_{K \in \mathscr{K}} C(K, A) \mid \text{for all } I \geq J \text{ such that } \phi^I_J(f_I) = f_J\right\},$$

where the homomorphisms ϕ^I_J are simply the projection functions such that for $I \geq J$ we have $\phi^I_J(f_I) = f_J$; the restriction f_I is projected onto a smaller restriction f_J. This condition also guarantees that the restriction functions are compatible. That is, for any point $x \in I \cap J$ we have $f_I(x) = f_J(x)$.

$$\begin{tikzcd}
(f) = \cdots \times & f_K \arrow[d, "\text{is in}"'] & \times & f_J \arrow[ll, bend right, "\phi^J_K"'] \arrow[d, "\text{is in}"'] & \times & f_I \arrow[ll, bend right, "\phi^I_J"'] \arrow[d, "\text{is in}"'] & \times \cdots \\
& C(K,A) & & C(J,A) & & C(I,A) &
\end{tikzcd}$$

Figure 6.3: A thread (f) in the inverse limit.

The purpose of defining such a construction is to prove an important theorem. We want to show that the inverse limit $\varprojlim C(\mathscr{K}, A)$ is "the same as" the set $C(X, A)$ with the compact-open topology. We do this by defining a homeomorphism $F : \varprojlim C(\mathscr{K}, A) \to C(X, A)$ so that

$$F(f) = \bigcup_{K \in \mathscr{K}} f_K : X \to A,$$

where the restriction f_K is in the subset $C(K, A)$. Defining the union of all restrictions makes sense because each element is a thread of the inverse limit and thus the restrictions are compatible.

Theorem 6.8.1 (The Inverse Limit Theorem)
The inverse limit $\varprojlim C(\mathscr{K}, A)$ *is homeomorphic to* $C(X, A)$.

PROOF Define $F : \varprojlim C(\mathscr{K}, A) \to C(X, A)$ as

$$F(f) = \bigcup_{K \in \mathscr{K}} f_K : X \to A.$$

For the result, we show that F is a homeomorphism by proving

(i) F is continuous,

(ii) F^{-1} is continuous, and

(iii) F is a bijection.

(i) To show the continuity of F, select any set $W(K, U)$ open in $C(X, A)$ and show the preimage $F^{-1}[W(K, U)]$ is an open set in $\varprojlim C(\mathscr{K}, A)$. Now, by the definition of F, the preimage

$$F^{-1}[W(K, U)] = \langle W(K, U)_K \rangle \bigcap \varprojlim C(\mathscr{K}, A).$$

That is, each preimage is mapped by F^{-1} to an open set in $\varprojlim C(\mathscr{K}, A)$ where each element (thread) satisfies the conditions of an inverse limit, and has as an element in the K-th coordinate system, all functions mapping compact K to open U. From the definition of the topology on the inverse limit, this is an open set. Hence F is continuous.

(ii) To show continuity of the inverse function F^{-1}, we show that the preimage $F[S]$ is open where

$$S = \langle W(K_1, U_1), \dots, W(K_n, U_n) \rangle \cap \varprojlim C(\mathscr{K}, A)$$

is an open set of the inverse limit. That is, S is the set containing all threads such that

$$\begin{tikzcd}
(f) = & f_{K_1} \arrow[d, "\text{is in}"'] & \times \cdots \times & f_{K_n} \arrow[d, "\text{is in}"'] & \times \cdots \\
& W(K_1, U_1) & & W(K_n, U_n) &
\end{tikzcd}$$

where $W(K_1, U_1), \dots, W(K_n, U_n)$ are open sets in the compact-open topology.

So we are interested in functions in $C(X, A)$ that map the compact sets K_i to U_i for $1 \leq i \leq n$ and all the other compact spaces can be mapped to any other open set U in A. Thus, F maps the set S to

$$\bigcap_{i=1}^{n} W(K_i, U_i) \ \bigcup \ \left(\bigcup_{K} W(K, U) \right)$$

for $K \notin \{K_1, \dots, K_n\}$. This is an open set in the compact-open topology on $C(X, A)$, and hence the inverse mapping F^{-1} is continuous.

(iii) We prove the mapping F is a bijection.

For injectivity we need to show

$$(f) \neq (g) \implies F((f)) \neq F((g)).$$

Assume $(f) \neq (g)$. This implies that there is a $Z \in \mathscr{K}$ such that $f_Z \neq g_Z$. That is, $\exists x \in Z : f_Z(x) \neq g_Z(x)$. But this implies

$$F((f)) \neq F((g)).$$

For surjectivity we exhibit for each $f \in C(X, A)$ there exists a $(f) \in \varprojlim C(\mathscr{K}, A)$ such that $F((f)) = f$. Pick any function $f \in C(X, A)$. We simply observe that the thread

$$(f) = \cdots \times f_I \times f_J \times f_K \times \cdots .$$

of restrictions of f to the compact subsets in $\mathscr{K}$ satisfying the properties of the inverse limit is the image of the function f in $C(X, A)$. That is,

$$\bigcup_{K \in \mathscr{K}} f_K = f,$$

therefore F is onto.

Hence, both F and F^{-1} are continuous and F is a bijection, and therefore a homeomorphism. ∎

6.9 Notes and Sources

The compact-open topology is well understood and its properties have been studied extensively and documented in much of the literature: Bourbaki [11], Dugundji [26], Engelking [27] and Kelley [36], where many of the topological results contained in this chapter can be found. Also useful is Aren's early paper *A topology for spaces of transformations* [3] giving the details of defining a metric space on the compact-open topology.

In Section 6.4 I used Dugundji [26] for most of the material regarding the continuity of composition of functions, evaluation and for the Equational Validity Lemma 6.7.1.

The inverse limit construction of a family of topological algebras can be found in [26] and in [43]. Its use in connection with $C(X, A)$ is ideally suited to approximate the compact bounded boxes seen in Constructive Volume Geometry.

Chapter 7

Expressiveness and Completeness

7.1 Introduction

As we have seen, the applications of spatial object data types are innumerable (e.g. from machine memory to medical imaging) and so are the possible choices of operations for each application. For example, Constructive Volume Geometry is an algebraic framework to support the selection of interesting high level operations in Volume Graphics and its applications. When dealing with CVG, computation of such spatial objects and their operations is vital. Thus, in the interest of developing a computability theory, some finite method of representing the spatial objects we are interested in is necessary.

Speaking generally, we would like some operations and some *basic* spatial objects that we could use to generate all the spatial objects we are interested in. In this way, we would have a composition of spatial objects and the operations performed on them to create more and more complex objects.

The primary obstacle to this approach is that this set of generated sequences of operations on spatial objects, which we will call "terms", is countable. Therefore we definitely cannot represent every spatial object in the uncountable spaces that we are interested in as a term.

We can however produce a set as described to generate all spatial objects possibly by *approximation.* We already have the necessary topological foundations which an approximation theory demands. This chapter presents the Stone-Weierstrass Theorem as our method of approximation.

7.2 The Expressiveness and Completeness Problems

This chapter aims to answer the completeness problem for the general theory of spatial data types. To answer such a question in a general setting is a complicated task, and we proceed in stages.

Approximation

To generalise, in the first stage we have to understand what it means for a set of operations to be adequate. We address this concern in Section 7.6. In choosing finitely many operations $F_1, \ldots, F_k$ on $C(X, A)$, we determine classes of spatial objects by applying the operations to some given set B of spatial objects. Specifically, each term over the signature of the chosen algebraic structure determines the spatial object which lies in the subalgebra generated by B, simply $\langle B \rangle_\Sigma$.

Clearly the set of terms is countable so we know that most spatial objects in $C(X, A)$ cannot be constructed by the repeated application of the operations. We therefore adapt the completeness question as follows:

Completeness *Can we find a collection of operations that is adequate to approximate all the spatial objects we are interested in?*

The idea of approximation adds a new and complicated parameter to the problem. For to understand adequacy we must choose not only a collection of operations, but a method of approximation. In any particular application area there may be more than one standard topology that captures useful notions of approximations.

For example, we may use the sup norm on $C(K, \mathbb{R})$, where K is compact. This approximation works by finding the maximum value the spatial object ϕ attains in the space K, even though this maximal value may only reflect a small portion of the entire function ϕ. Of course the sup norm and compact-open topology are a starting point for any continuous spatial data type, and for the purposes of this chapter we use the compact-open topology (induced by the sup norm). We have already developed the mathematics required for approximation with the sup norm in Chapter 6 and provided some useful topological and algebraic results.

We are now in a position to give an account of the completeness problem.

Informal Definition of the Completeness Problem:
Can we find

(i) a collection of operators $F = \{F_1, \ldots, F_k\}$ on $C(X, A)$,

(ii) a subset $B \subset C(X, A)$ of basic spatial objects

such that when we apply the operations of F to the elements of B the set $\langle B \rangle_F$ we generate is adequate to approximate all spatial objects. Specifically, factors we also need to consider

(iii) a topology on $C(X, A)$ to formulate the method of approximation.

In this case, the requirement of adequacy based on approximation is simply

$$\langle B \rangle_F \text{ is dense in } C(X, A).$$

The method of approximation that we use, and is of central importance to our work, is the *Stone-Weierstrass Theorem* given in Section 7.3. This theorem allows us to approximate any spatial object in $C(K, \mathbb{R})$ using only objects that are generated by performing some simple operations on a set $B \subset C(K, \mathbb{R})$ of basic spatial objects. This set B must satisfy some simple properties:

1. The basic spatial objects separate the points of the space K, and
2. contains at least one non-zero constant spatial object.

In Section 7.4 we use this result to extend the theorem to other types of spatial objects

- $C(K, [0, 1])$, mapping points of K to the unit interval (Lemma 7.4.1),
- $C(K, [0, 1]^m \times \mathbb{R}^n)$, mapping points of K to the Cartesian product $[0, 1]^m \times \mathbb{R}^n$ (Lemma 7.4.2); of particular importance for CVG, and
- as a special case of the previous result, setting $m = 0$ gives us an approximation for spatial objects in $C(K, \mathbb{R}^n)$.

Section 7.6 examines conditions for Σ-operations to be adequate i.e. generate a dense subset by means of the Stone-Weierstrass Theorem.

Lastly, Section 7.7 extends Stone-Weierstrass to cases where K is not compact. Using the standard inverse limit construction of compact sets we show a spatial object which approximates locally on a compact set is suitable to approximate globally.

7.3 The Stone-Weierstrass Theorem

In Section 6.6 we described the conditions needed for a set to be *dense* in $C(K, \mathbb{R})$. In this section we describe how the *Stone-Weierstrass* method of approximation uses some simple operations on basic spatial objects to produce a dense subset of all spatial objects. We do not limit ourselves to just the spatial objects of the form $\phi : K \rightarrow \mathbb{R}$, but also look at the spatial objects in

$$C(K, [0, 1]) \text{ and } C(K, [0, 1]^m \times \mathbb{R}^n).$$

These results will be particularly useful when applying the general theory to CVG.

7.3.1 The Sublattice $C(X, \mathbb{R})$

We consider a lattice structure on $C(X, \mathbb{R})$, for an arbitrary topological space X (not necessarily compact) and prove a result which shows $C(X, \mathbb{R})$ satisfies the properties of a lattice. The lattice operations have already been defined in Chapter 4 for the function space $F(X, \mathbb{R})$ and they are displayed as:

Algebra	$F(X, \mathbb{R})_{Lat}$
Carriers	$F(X, \mathbb{R})$
Operations	$\wedge : F(X, \mathbb{R}) \times F(X, \mathbb{R}) \to F(X, \mathbb{R})$ $\vee : F(X, \mathbb{R}) \times F(X, \mathbb{R}) \to F(X, \mathbb{R})$
Definitions	$(f \wedge g)(x) = \min(f(x), g(x))$ $(f \vee g)(x) = \max(f(x), g(x))$

We prove that

$$C(X, \mathbb{R})_{Lat} \in Alg(\Sigma_{lat}, T_{Lat})$$

by showing it is a sublattice (Σ_{lat}-subalgebra) of $F(X, \mathbb{R})$:

Lemma 7.3.1 *For any topological space X, $C(X, \mathbb{R})_{Lat}$ is a Σ_{lat}-subalgebra of $F(X, \mathbb{R})_{Lat}$.*

Proof We must show that for any continuous functions $f, g \in C(X, \mathbb{R})$ the functions $(f \wedge g)(x) = \min(f(x), g(x))$ and $(f \vee g)(x) = \max(f(x), g(x))$ are also in $C(X, \mathbb{R})$. We proceed as in [52](p.107).

Observe that intervals of the form $A = (-\infty, a)$ and $B = (b, +\infty)$ for $a, b \in \mathbb{R}$ form an open subbase of $\mathbb{R}$. That is, all open sets can be described in terms of finite intersections of A and B. For example: $(0, 1) = (-\infty, 1) \cap (0, +\infty)$, $(-1, 1) = (-\infty, 1) \cap (-1, +\infty)$. We show $(f \wedge g)$ and $(f \vee g)$ are continuous by showing the preimage of the sets A and B are open.

We show that the function $(f \wedge g)$ is in $C(X, \mathbb{R})$:

Case: A

$$\begin{aligned} (f \wedge g)^{-1}[A] &= \{x : \min(f(x), g(x)) < a\} \\ &= \{x : f(x) < a\} \cup \{x : g(x) < a\} \end{aligned}$$

Case: B

$$\begin{aligned} (f \wedge g)^{-1}[B] &= \{x : \min(f(x), g(x)) > b\} \\ &= \{x : f(x) > b\} \cap \{x : g(x) > b\} \end{aligned}$$

Similarly, we show that the function $(f \vee g)$ is in $C(X, \mathbb{R})$:

Case: A

$$\begin{aligned}(f \vee g)^{-1}[A] &= \{x : \max(f(x), g(x)) < a\} \\ &= \{x : f(x) < a\} \cap \{x : g(x) < a\}\end{aligned}$$

Case: B

$$\begin{aligned}(f \vee g)^{-1}[B] &= \{x : \max(f(x), g(x)) > b\} \\ &= \{x : f(x) > b\} \cup \{x : g(x) > b.\}\end{aligned}$$

Each case yields a finite intersection of subbasic open sets of the form $(-\infty, a)$ and $(b, +\infty)$, and hence are open in $\mathbb{R}$. Since $(f \wedge g)$ and $(f \vee g)$ are continuous, they are in $C(X, \mathbb{R})$ which satisfies the sublattice conditions completing the proof. ∎

7.3.2 Dense Lattices of $C(K, \mathbb{R})$

Consider a subset L of the set $C(K, \mathbb{R})$ of all continuous spatial objects mapping a compact space K to the real numbers. Using the operations of a lattice on this subset L of continuous spatial objects, we show the generated subalgebra $\langle L \rangle_{\Sigma_{Lat}}$ is dense in $C(K, \mathbb{R})$. To do this, we need the following property.

Definition 7.3.1 (Strong Separation) *We say a subset L of $C(K, \mathbb{R})$ is strongly separating if there exists a function f_{xy} in L such that for any distinct points $x, y \in K$ and numbers $a, b \in \mathbb{R}$ we have $f_{xy}(x) = a$ and $f_{xy}(y) = b$. We say that the function f_{xy} is a strongly separating function.*

Theorem 7.3.1 *Let L be a Σ_{lat}-subalgebra of $C(K, \mathbb{R})$ that strongly separates the points of K. Then the subalgebra generated by the operations in Σ_{lat} on the set L is dense in $C(K, \mathbb{R})$.*

i.e. Given $\epsilon > 0$ and $\phi \in C(K, \mathbb{R})$, we can construct a g in $\langle L \rangle_{\Sigma_{Lat}}$ such that

$$\|\phi - g\|_\infty < \epsilon.$$

Proof Adapting the proof of [52](p.158) and [47](p.165), we construct a function g in $\langle L \rangle_{\Sigma_{Lat}}$ which can approximate the given function $\phi \in C(K, \mathbb{R})$ to within an ϵ-error margin.

Fix a point x in K and follow this procedure for every point $y \neq x$ in K:

1. By the strong separation property of L, we have a function $f_{xy} : K \to \mathbb{R}$ in L such that $f_{xy}(x) = \phi(x)$ and $f_{xy}(y) = \phi(y)$.

2. Obtain an open set $G_y = \{z \in K \mid f_{xy}(z) < \phi(z) + \epsilon\}$. This is an open cover for K since each point y is contained in at least one G_y; i.e. $f_{xy}(x) = \phi(x) < \phi(x) + \epsilon$ and $f_{xy}(y) = \phi(y) < \phi(y) + \epsilon$.

3. By the compactness of K select a finite subcover $G_{y_1}, \ldots, G_{y_m}$. We pick functions $f_{xy_1}, \ldots, f_{xy_n}$ in L which we associate with each of these open sets whereby $f_{xy_i}(x) = \phi(x)$ and $f_{xy_i}(y_i) = \phi(y_i)$.

4. We construct the function $f_x = f_{xy_1} \wedge \cdots \wedge f_{xy_m}$. Observe that f_x is in $\langle L\rangle_{\Sigma_{lat}}$ and has the property $f_x(z) < \phi(z) + \epsilon$ for each $z \in K$.

5. Define the open set $H_x = \{z \in K \mid f_x(z) > \phi(z) - \epsilon\}$. This set is non-empty since $f_x(x) = \phi(x) > \phi(x) - \epsilon$.

We now have an open set H_x and an associated function f_x which has the property $\|\phi - f_x\|^{H_x} < \epsilon$ for each point x in K. By the compactness properties of K, we select open sets $H_{x_1}, \ldots, H_{x_k}$ which provide a finite subcover of K.

With each function f_{x_i} associated with the open set H_{x_i} we define $g = f_{x_1} \vee \cdots \vee f_{x_k}$ also in $\langle L\rangle_{\Sigma_{lat}}$. The functions f_{x_i} are bounded below by $\phi(z) - \epsilon$, and thus so is g. Now, observe that for all $z \in K$, $g(z) > \phi(z) - \epsilon$ and each f_{x_i} has the property such that for all $z \in K$ $f_{x_i}(z) < \phi(z) + \epsilon$. Hence, we conclude that $\phi(z) - \epsilon < g(z) < \phi(z) + \epsilon$ for all z in K. That is,

$$\|\phi - g\|_{\infty} < \epsilon,$$

which means that $g \in \langle L\rangle_{\Sigma_{lat}}$ is a suitable approximating function for ϕ. ∎

This theorem describes a method of producing a function g in L which approximates the input ϕ to within the error-margin ϵ. In fact, the proof gives us a method of constructing such a function using finite meets and joins. The general form of this term is

$$t(y_{11}, \ldots, y_{1m}, \ldots, y_{k1}, \ldots, y_{km}) = (y_{11} \wedge \cdots \wedge y_{1m}) \vee \cdots \vee (y_{k1} \wedge \cdots \wedge y_{km})$$

and evaluation with the functions described in the proof yields

$$\begin{aligned}&[\![t(y_{11}, \ldots, y_{1m}, \ldots, y_{k1}, \ldots, y_{km})]\!](f_{x_1y_1}, \ldots, f_{x_1y_m}, \ldots, f_{x_ky_1}, \ldots, f_{x_ky_m})\\ &= (f_{x_1y_1} \wedge \cdots \wedge f_{x_1y_m}) \vee \cdots \vee (f_{x_ky_1} \wedge \cdots \wedge f_{x_ky_m})\\ &= g\end{aligned}$$

which is an approximating function of ϕ.

7.3.3 The Stone-Weierstrass Algebra

Theorem 7.3.1 proves, essentially, a density result for a sublattice L of $C(K, \mathbb{R})$. To do this, we assume a strong separation property, i.e. a function f_{xy} in L such that for any distinct points $x, y \in K$ and numbers $a, b \in \mathbb{R}$ we have $f_{xy}(x) = a$ and $f_{xy}(y) = b$. We have no information on how such a function could be constructed, merely the guarantee that it exists. This makes analysing the effectivity of the theorem impossible.

Therefore, we introduce a signature Σ_{SW} which has operations of addition, multiplication and scalar multiplication for spatial objects, and is displayed as

Signature	Σ_{SW}
Import	*Data*
Sorts	*Space*, *SO*
Operations	$+ : SO \times SO \rightarrow SO$ $\times : SO \times SO \rightarrow SO$ $\cdot : Data \times SO \rightarrow SO$

Using this signature we explicitly construct strongly separating functions by performing the Σ_{SW}-operations on a subset of $C(K, \mathbb{R})$, which we will call *basic spatial objects*. This set of basic spatial objects requires two weaker assumptions. The set must contain:

1. a non-zero constant spatial object, and
2. for any points $x \neq y$ in K there exists a separating function h such that $h(x) \neq h(y)$.

Separating functions are generally chosen depending on the application. For example, users of CVG may have functions which are relevant to the graphics application which happens to separate points. This would be the obvious choice of separating functions in this instance. From these two assumptions we construct a strong separating function in Theorem 7.3.2. Theorem 7.3.1 then shows that the subalgebra generated by the repeated application of the Σ_{SW}-operations on the basic spatial objects is dense in $C(K, \mathbb{R})$.

Recall in Chapter 6 we defined the compact-open topology on the set $C(K, \mathbb{R})$ of all continuous functions from K to $\mathbb{R}$. We now define a topological Σ_{SW}-algebra with the carrier $C(K, \mathbb{R})$ which interprets the symbols $+$, $\times$ and $\cdot$ in Σ_{SW} by pointwise extension from the common functions $+_{\mathbb{R}}$, $\times_{\mathbb{R}}$ and $\cdot_{\mathbb{R}}$ on $\mathbb{R}$. That is, $C(K, \mathbb{R})$ has the compact-open topology, and the

Σ_{SW}-operations are continuous by Lemma 6.5.1, Chapter 6. We display the interpreting Σ_{SW}-algebra $C(K, \mathbb{R})$ as follows:

Algebra	$C(K, \mathbb{R})$
Import	$\mathbb{R}$
Carriers	K, $C(K, \mathbb{R})$
Operations	$+ : C(K, \mathbb{R}) \times C(K, \mathbb{R}) \to C(K, \mathbb{R})$ $\times : C(K, \mathbb{R}) \times C(K, \mathbb{R}) \to C(K, \mathbb{R})$ $\cdot : \mathbb{R} \times C(K, \mathbb{R}) \to C(K, \mathbb{R})$
Definitions	$(\phi + \phi')(x) = \phi(x) +_{\mathbb{R}} \phi'(x)$ $(\phi \times \phi')(x) = \phi(x) \times_{\mathbb{R}} \phi'(x)$ $(\lambda \cdot \phi)(x) = \lambda \cdot_{\mathbb{R}} \phi(x)$

We use the Σ_{SW} signature not only for the set $C(K, \mathbb{R})$, but also for our extensions of the Stone-Weierstrass Theorem. We show some of these sets in Figure 7.1.

$$\begin{tikzcd}
& \Sigma_{SW} \arrow[dl] \arrow[d] \arrow[dr] & \\
C(K, \mathbb{R}) & C(K, [0,1]) & C(K, \mathbb{R}^n)
\end{tikzcd}$$

Figure 7.1: Examples of Σ_{SW}-algebras

Suppose that B is a topological Σ_{SW}-subalgebra of $C(K, \mathbb{R})$. We show that the closure $\overline{B}$, Definition 3.8.6 in the preliminaries, is also a Σ_{SW}-subalgebra.

Lemma 7.3.2 *If $B \subset C(K, \mathbb{R})$ is a Σ_{SW}-subalgebra then the closure $\overline{B}$ is also a Σ_{SW}-subalgebra in $C(K, \mathbb{R})$.*

PROOF The main idea behind the proof is to show if ϕ and ϕ' are spatial objects in $\overline{B}$ then so are

$$\phi + \phi', \ \phi \times \phi' \text{ and } \lambda \cdot \phi.$$

The reader is refered to [26](p.280) for complete details. ∎

7.3.4 The Absolute Value Function

The absolute value function on the real numbers is commonly defined by two cases

$$|r| = \begin{cases} r & \text{if } r \geq 0, \\ -r & \text{if } r < 0. \end{cases}$$

We lift the absolute function pointwise to define an operator $|-| : C(K, \mathbb{R}) \to C(K, \mathbb{R})$ as $|\phi|(x) = |\phi(x)|$ for a spatial object ϕ in $C(K, \mathbb{R})$ and a point x in X.

Lemma 7.3.3 *The absolute value function $G = |-| : C(K, \mathbb{R}) \to C(K, \mathbb{R})$ is continuous.*

PROOF Before proving the result, it is useful to first prove the continuity of $g = |-| : \mathbb{R} \to \mathbb{R}$. We consider three cases as follows:

1. $0 < a < b$. We have

   $$g^{-1}[(a, b)] = (a, b) \cup (-a, -b),$$

   which is open in $\mathbb{R}$.

2. $a < b < 0$. Since g returns no negative values

   $$g^{-1}[(-a, -b)] = \emptyset.$$

3. $a < 0 < b$. The interval containing 0 is the most complicated case. We can rewrite the interval $(-a, b)$ as $(-a, 0) \cup \{0\} \cup (0, b)$. Then

   $$\begin{aligned} g^{-1}[(-a, b)] &= g^{-1}[(-a, 0) \cup \{0\} \cup (0, b)] \\ &= \emptyset \cup \{0\} \cup (0, b) \cup (-b, 0) \\ &= (-b, b). \end{aligned}$$

   which is an open set in $\mathbb{R}$.

Thus g is a continuous function. By Theorem 6.5.1 in Chapter 6, the pointwise lifted function $G : C(K, \mathbb{R}) \to C(K, \mathbb{R})$ defined by

$$G(\phi)(x) = |\phi(x)|$$

is continuous. ■

Lemma 7.3.4 *Let B be a Σ_{SW}-subalgebra of $C(K, \mathbb{R})$, and $\overline{B}$ it's closure (also a Σ_{SW}-subalgebra). If f is a function in B then the function $|f|$ is in $\overline{B}$.*

PROOF In [26](p.280), it is proven than there exists a sequence of polynomials $\{p_n\}$ that converges uniformly on $[0,1]$ to the function $\phi(t) = \sqrt{t}$. Now to prove that the absolute value $|f|$ is in $\overline{B}$ we show that each neighbourhood $\bigcap_{i=1}^{n} W(K_i, U_i)$ of $|f|$ contains some function $g \in B$. It turns out that given $\epsilon > 0$ and letting $g(x) = \sqrt{x}$ yields $||f|(x) - g(x)| < \epsilon$ for each point $x \in K = \bigcup_{i=1}^{n} K_i$.

By the compactness of K, each function is bounded in $C(K, \mathbb{R})$, and so we have a constant bound $C < \infty$ for $|f|$ where $|f(x)| \leq C$ for all $x \in K$. Using the sequence of polynomials $\{p_n\}$ we write

$$p_n(\frac{f^2}{C}) = \sqrt{\frac{f^2}{C^2}} = \frac{|f|}{C}$$

uniformly on K.

Since each polynomial in the sequence $p_n(\frac{f^2}{C^2})$ is in B and the uniform limit $|f|$ is in $\overline{B}$ we have the result. ■

We use this result on the set $\langle B\rangle_{\Sigma_{SW}}$, such that if $f \in \langle B\rangle_{\Sigma_{SW}}$ then Lemma 7.3.4 shows that $|f| \in \overline{\langle B\rangle}_{\Sigma_{SW}}$. That is, the absolute value of the spatial object can be approximated by objects in $\langle B\rangle_{\Sigma_{SW}}$.

7.3.5 The *Max* and *Min* Functions

The set $C(K, \mathbb{R})$ with operations $\wedge$ and $\vee$ is a sublattice of $F(K, \mathbb{R})$ by Lemma 7.3.1, and can be implemented using the standard max and min functions on the reals:

$$\begin{aligned}(\phi \vee \phi')(x) &= Max(\phi, \phi')(x)\\ &= \max(\phi(x), \phi'(x)),\end{aligned}$$

$$\begin{aligned}(\phi \wedge \phi')(x) &= Min(\phi, \phi')(x)\\ &= \min(\phi(x), \phi'(x))\end{aligned}$$

for all $x \in X$ and $\phi, \phi' \in C(K, \mathbb{R})$. These standard functions can be expressed in terms of arithmetical operations and the absolute value function on $\mathbb{R}$:

Lemma 7.3.5 *The equations*

$$\max(a,b) = \frac{1}{2}(a+b) + \frac{1}{2}|a-b|, \tag{7.1}$$

$$\min(a,b) = \frac{1}{2}(a+b) - \frac{1}{2}|a-b|. \tag{7.2}$$

hold for all $a, b \in \mathbb{R}$.

Proof (7.1) There are two cases to consider:

Case 1. $a \geq b$.

$$\begin{aligned} \max(a,b) &= a \\ &= \frac{1}{2}(a+b) + \frac{1}{2}(a-b) \\ &= \frac{1}{2}(a+b) + \frac{1}{2}|a-b|. \end{aligned}$$

Case 2. $a < b$.

$$\begin{aligned} \max(a,b) &= b \\ &= \frac{1}{2}(a+b) + \frac{1}{2}(b-a) \\ &= \frac{1}{2}(a+b) + \frac{1}{2}(-(a-b)) \\ &= \frac{1}{2}(a+b) + \frac{1}{2}|a-b| \text{ by definition of absolute value.} \end{aligned}$$

(7.2) The proof for the min function is similar. ∎

The next lemma says that the meet and join operations of a lattice on $C(K,\mathbb{R})$ can be described in terms of the elements in the generated Σ_{SW}-subalgebra $\langle B\rangle_{\Sigma_{SW}}$. This result is taken from [52](p.159).

Lemma 7.3.6 *The Σ_{SW}-subalgebra $\overline{B}$ of $C(K,\mathbb{R})$ is a Σ_{lat}-subalgebra of $C(K,\mathbb{R})$. That is, $\overline{B} \in Alg(\Sigma_{lat}, T_{lat})$.*

Proof [52](p.159) By Lemma 7.3.4 for $\phi \in B$ the absolute value $|\phi|$ is in $\overline{B}$. Thus each of the necessary operations to define the Max and Min functions are in $\overline{B}$, and therefore $\phi \wedge \phi'$ and $\phi \vee \phi'$ are in $\overline{B}$, yielding the result. ∎

7.3.6 The Stone-Weierstrass Theorem for $\mathbb{R}$

The theoretical framework which we have developed thus far greatly simplifies the proof of the Stone-Weierstrass Theorem. Now, suppose we have some subset $B = \{b_1, b_2, \ldots\}$ of $C(K,\mathbb{R})$ of basic spatial objects. With some additional properties, we can replace the strong separation property on our set B with a weaker property:

Definition 7.3.2 (Weak Separation) *We say a set $B \subset C(K,\mathbb{R})$ has the weak separation property if for each pair of points x and y in K there exists a function $h \in B$ such that*

$$x \neq y \implies h(x) \neq h(y).$$

We prove a result which shows that the strong separation property can be replaced on the subset B with a weaker separation property, assuming that we have at least one non-zero constant spatial object in B:

Lemma 7.3.7 *Let B be a subset of $C(K, \mathbb{R})$ which is*

- *weakly separating, and*
- *contains a non-zero constant spatial object.*

Then $\langle B \rangle_{\Sigma_{SW}}$ is strongly separating.

Proof Given x and y in K and a, b in $\mathbb{R}$ we have a $h : K \longrightarrow \mathbb{R}$ in B such that $h(x) \neq h(y)$. Now construct a spatial object $f_{xy} : K \longrightarrow \mathbb{R}$ in $\langle B \rangle_{\Sigma_{SW}}$ such that

$$f_{xy}(x) = a \text{ and } f_{xy}(y) = b$$

for any two distinct points $x \neq y$ in K and $a, b \in \mathbb{R}$. Consider a function $\lambda h + \mu$ where λ, μ are constants in $\mathbb{R}$ such that the equations

$$a = \lambda h(x) + \mu \tag{7.3}$$
$$b = \lambda h(y) + \mu \tag{7.4}$$

are satisfied. To find the appropriate values for λ and μ, we rewrite Eq. 7.3 as

$$\mu = a - \lambda h(x) \tag{7.5}$$

and substitute into Eq. 7.4 yielding

$$\begin{aligned} b &= \lambda h(y) + (a - \lambda h(x)) \\ &= \lambda(h(y) - h(x)) + a \end{aligned} \tag{7.6}$$

and rewriting Eq. 7.6 gives the value for λ:

$$\lambda = \frac{b-a}{h(y) - h(x)}.$$

By substituting this value into Eq. 7.5 yields the value for μ:

$$\mu = a - h(x) \cdot \left[\frac{b-a}{h(y) - h(x)}\right].$$

We set $f_{xy} = \lambda h + \mu$ as the required strongly separating function, completing the proof. ∎

Theorem 7.3.2 (Stone-Weierstrass) *Let K be a compact topological space and $C(K, \mathbb{R})$ be the set of all continuous spatial objects $\phi : K \longrightarrow \mathbb{R}$ with the compact-open topology. If B is a subset of $C(K, \mathbb{R})$ which*

- *is weakly separable, and*
- *contains a non-zero constant spatial object,*

then the Σ_{SW}-subalgebra generated by B is dense in $C(K,\mathbb{R})$.
i.e. Given $\epsilon > 0$ and $\phi \in C(K,\mathbb{R})$, we can construct a $g \in \langle B\rangle_{\Sigma_{SW}}$ such that

$$\left\|\phi - g\right\|_{\infty} < \epsilon.$$

PROOF By Lemma 7.3.7 we can construct a strong separating function in $\langle B\rangle_{\Sigma_{SW}}$ and by applying Lemma 7.3.6 to $\overline{\langle B\rangle}_{\Sigma_{SW}}$, we conclude $\overline{\langle B\rangle}_{\Sigma_{SW}}$ is a lattice.

Applying the density result for lattices in Theorem 7.3.1, the set $\overline{\langle B\rangle}_{\Sigma_{SW}}$ is dense in $C(K,\mathbb{R})$, and since it is the closure, $\overline{\langle B\rangle}_{\Sigma_{SW}} = C(K,\mathbb{R})$. But by Definition 3.8.14 this just means that $\langle B\rangle_{\Sigma_{SW}}$ is dense in $C(K,\mathbb{R})$, yielding the result.

7.4 Extensions of the Stone-Weierstrass Theorem

We have given a theory of spatial data types that maps points in a space to data attributes. These attributes can be many things, such as tuples of real numbers or more simply, spatial objects in $C(K,[0,1])$. How are we to adapt the Stone-Weierstrass Theorem to these types of spatial objects? Our motivation is chiefly from our need to approximate CVG spatial objects.

In this section we prove some extensions to the Stone-Weierstrass Theorem: for spatial objects of the form $K \longrightarrow [0,1]$ and $K \longrightarrow [0,1]^m \times \mathbb{R}^n$, where the unit interval is usually interpreted by CVG as an "opacity" channel representing the visual geometry of an object.

7.4.1 The Stone-Weierstrass Theorem Over $[0,1]$

Using the signature Σ_{SW}, we introduce an interpreting Σ_{SW}-algebra which uses specialized operations $+_I$, $\times_I$ and $\cdot_I$ on the unit interval. For these operations, we define a function which in effect "chops" the real number value to a value in the unit interval.

Definition 7.4.1 (The Chop Function) *The function $\alpha : \mathbb{R} \longrightarrow [0,1]$ is defined by the following three cases for $r \in \mathbb{R}$:*

$$\alpha(r) = \begin{cases} 1 & \textit{if } r > 1, \\ r & \textit{if } r \in [0,1], \\ 0 & \textit{if } r < 0. \end{cases}$$

We lift this function pointwise to define an operator $A : C(K,\mathbb{R}) \to C(K,[0,1])$ as

$$A(\phi)(x) = \alpha(\phi(x)) \tag{7.7}$$

which takes a ϕ in $C(K,\mathbb{R})$ and "chops" it to return a function in the set $C(K,[0,1])$. The chop function is used to define an interpreting algebra with the carrier $C(K,[0,1])$ for the signature Σ_{SW}.

Algebra	$C(K,[0,1])$
Import	$[0,1]$
Carriers	K, $C(K,[0,1])$
Operations	
	$+_I : C(K,[0,1]) \times C(K,[0,1]) \to C(K,[0,1])$ $\times_I : C(K,[0,1]) \times C(K,[0,1]) \to C(K,[0,1])$ $\cdot_I : [0,1] \times C(K,[0,1]) \to C(K,[0,1])$
Definitions	$(\phi +_I \phi')(x) = \alpha(\phi(x) +_{\mathbb{R}} \phi'(x))$ $(\phi \times_I \phi')(x) = \phi(x) \times_{\mathbb{R}} \phi'(x)$ $(\lambda \cdot_I \phi)(x) = \lambda \cdot_{\mathbb{R}} \phi(x)$, $0 \leq \lambda \leq 1$

Corollary 7.4.1 *Let K be a compact topological space and $C(K,[0,1])$ be the space of all continuous spatial objects $\phi : K \to [0,1]$ with the compact open topology. If B is a subset of $C(K,[0,1])$ which*

- *separates the points of space K,*
- *contains a non-zero constant spatial object,*

then the Σ_{SW}-subalgebra generated by the repeated application of the Σ_{SW}-operations on B is dense in $C(K,[0,1])$.

That is, given $\epsilon > 0$ and any $\phi \in C(K,[0,1])$, we can produce a $g \in \overline{\langle B \rangle}_{\Sigma_{SW}}$ such that

$$\|\phi - g\|_\infty < \epsilon.$$

PROOF We use the Stone-Weierstrass Theorem 7.3.2 to produce a $g' : K \to \mathbb{R}$ in $C(K,\mathbb{R})$ with the property

$$\|\phi - g'\|_\infty < \frac{\epsilon}{2}.$$

From g' we construct a suitable approximation $g : K \longrightarrow [0,1]$ of ϕ. To this end, we show that the function $A(g') : K \longrightarrow [0,1]$ defined in Equation 7.7 is in $C(K,[0,1])$ and is a suitable candidate to approximate the function ϕ within an ϵ-error margin.

Sub Lemma 7.4.1 *Let $g' : K \longrightarrow \mathbb{R}$ be in $C(K,\mathbb{R})$. Then the composition $(\alpha \circ g') : K \longrightarrow [0,1]$ is continuous.*

PROOF To show the continuity of $\alpha : \mathbb{R} \longrightarrow [0,1]$, we exhibit an open preimage $\alpha^{-1}[U]$ for any open set $U = (a,b) \subset [0,1]$ where $0 < a < b < 1$. This is obvious from Case 2 of the definition of α; which is simply the identity function. Thus

$$\alpha^{-1}[(a,b)] = (a,b)$$

and by the continuity of composition of functions, $(\alpha \circ g') \in C(K,[0,1])$. ∎

To show that $g = A(g')$ is a satisfactory approximation of ϕ we do a case analysis.

Case 1 If $g'(z) > 1$ then $\alpha(g'(z)) = 1$ for all $z \in K$, and so we know that $|(\alpha \circ g')(z)| \le |g'(z)|$. Furthermore, we have the inequalities $1 < g'(z) < 1 + \frac{\epsilon}{2}$ due to the fact that g' is a suitable approximation of ϕ. By the triangle inequality we write

$$\begin{aligned} \sup_{z\in K}|\phi(z) - g(z)| &\le \sup_{z\in K}|\phi(z) - g'(z)| + \sup_{z\in K}|g'(z) - (\alpha \circ g')(z)| \\ &< \frac{\epsilon}{2} + \sup_{z\in K}|g'(z) - 1| \\ &< \frac{\epsilon}{2} + \left|\left(1 + \frac{\epsilon}{2}\right) - 1\right| \\ &< \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon. \end{aligned}$$

Case 2 If $g'(z) \in [0,1]$ then $\alpha(g'(z)) = g'(z)$. This case is trivial since

$$\begin{aligned} \sup_{z\in K}|\phi(z) - g(z)| &= \sup_{z\in K}|\phi(z) - (\alpha \circ g')(z)| \\ &= \sup_{z\in K}|\phi(z) - g'(z)| \\ &< \frac{\epsilon}{2} \\ &< \epsilon. \end{aligned}$$

Case 3 If $g'(z) < 0$ then $\alpha(g(z)) = 0$ for all $z \in K$. But g' is a suitable approximation for ϕ, the inequality

$$0 < \sup_{z\in K}|g'(z)| < \frac{\epsilon}{2} \tag{7.8}$$

is true, and so by the triangle inequality:

$$\begin{aligned}
\sup_{z\in K}|\phi(z)-g(z)| &\leq \sup_{z\in K}|\phi(z)-g'(z)|+\sup_{x\in Z}|g'(z)-g(z)| \\
&< \frac{\epsilon}{2}+\sup_{z\in K}|g'(z)-(\alpha\circ g')(z)| \\
&< \frac{\epsilon}{2}+\sup_{z\in K}|g'(z)| \\
&< \frac{\epsilon}{2}+\frac{\epsilon}{2} \text{ by Inequality 7.8} \\
&< \epsilon.
\end{aligned}$$

By the above cases the function $g = A(g')$ in $C(K,[0,1])$ has the property

$$\|\phi - g\|_\infty < \epsilon$$

and is a sufficient approximation to ϕ, and thus is the required approximating function. ■

7.4.2 The Stone-Weierstrass Theorem Over $m + n$ Dimensions

Spatial objects in the Constructive Volume Geometry framework often have the form

$$\phi : K \to [0,1]^m \times \mathbb{R}^n,$$

where K is a compact space of E^3 and $[0,1]^m \times \mathbb{R}^n$ represent *attributes* such as colour, opacity or any measurable physical quantity. This section describes a general Stone-Weierstrass Theorem that yields an approximating function for any spatial object of the form $\phi : K \to [0,1]^m \times \mathbb{R}^n$.

Before extending the Stone-Weierstrass Theorem, we make an observation and consider some technical details.

Observation 7.4.1 *The function $\phi \in C(K,[0,1]^m \times \mathbb{R}^n)$ can be written as an $(m+n)$-tuple. For each $x \in K$*

$$\phi(x) = (\phi_1(x), \ldots, \phi_{m+n}(x)).$$

When we wish to single out and work with a specific coordinate function ϕ_i of ϕ we use *projection functions.*

Definition 7.4.2 *Let $\phi : K \to [0,1]^m \times \mathbb{R}^n$. Define the projection function $\pi_i : [0,1]^m \times \mathbb{R}^n \to \mathbb{R}$ as*

$$\pi_i(a_1, \ldots, a_{m+n}) = a_i.$$

The ith coordinate function $\phi_i : K \to \mathbb{R}$ of ϕ is the composition $\pi_i \circ \phi : K \to [0,1]^m \times \mathbb{R}^n$.

Using the observation and definition, we state a lemma for which the proof can be found in [47](p.18).

Lemma 7.4.1 *A function $\phi : K \to [0,1]^m \times \mathbb{R}^n$ is continuous if, and only if each coordinate function ϕ_i is continuous.*

PROOF $\Rightarrow$ Suppose ϕ is continuous. Then by the Composition Theorem 6.4.2, $\pi_i \circ \phi$ is continuous, for $1 \leq i \leq m+n$.

$\Leftarrow$ Suppose each coordinate function ϕ_i is continuous, for $1 \leq i \leq m+n$. Then for open sets $U_1, \ldots, U_m \subseteq [0,1]$ and open sets $U_{m+1}, \ldots, U_{m+n} \subseteq \mathbb{R}$ each preimage $(\pi_i \circ \phi)^{-1}[U_i]$ is open for $1 \leq i \leq m+n$.

But $\phi^{-1}[\pi^{-1}[U_i]] = (\pi \circ \phi)^{-1}[U_i]$; so $(\pi \circ \phi)$ maps open sets to open sets by the continuity of $(\pi \circ \phi)$, and thus ϕ is continuous. ■

To measure the distance between spatial objects we use the Euclidean norm defined as

$$\|\mathbf{z}\|_2 = \sqrt{\sum_{i=1}^{m+n} |z_i|^2},$$

where $\mathbf{z} \in [0,1]^m \times \mathbb{R}^n$. The sup norm over $C(K, [0,1]^m \times \mathbb{R}^n)$ is then

$$\|\phi\|_\infty = \sup_{x \in K}\{\|\phi(x)\|_2\}$$

for $\phi \in C(K, [0,1]^m \times \mathbb{R}^n)$.

This Corollary is in essence a coordinatewise extension to the original Stone-Weierstrass Theorem. Our reasoning follows closely to that of [47](p.166).

Corollary 7.4.2 *Let K be a compact space and $C(K, [0,1]^m \times \mathbb{R}^n)$ the collection of continuous spatial objects with the norm $\|\phi\|_\infty = \sup_{x \in K}\{\|\phi(x)\|_2\}$. If H is a subset of $C(K, [0,1]^m \times \mathbb{R}^n)$ which*

- *separates the points of space K,*
- *contains a non-zero constant spatial object,*

then the Σ_{SW}-subalgebra generated by repeated application of Σ_{SW}-operations on the spatial objects in H is dense in $C(K, [0,1]^m \times \mathbb{R}^n)$. That is, given $\epsilon > 0$ and $\phi \in C(K, [0,1]^m \times \mathbb{R}^n)$, we can produce a $g \in \langle H \rangle_{\Sigma_{SW}}$ such that

$$\|\phi - g\|_\infty < \epsilon.$$

PROOF By Observation 7.4.1, the function we wish to approximate is of the form $\phi = (\phi_1, \ldots, \phi_m, \phi_{m+1}, \ldots, \phi_{m+n})$. We prove the Corollary by finding approximating functions for each $\phi_i \in C(K, [0,1])$ for $1 \leq i \leq m$ and $\phi_j \in C(K, \mathbb{R})$ for $m+1 \leq j \leq m+n$.

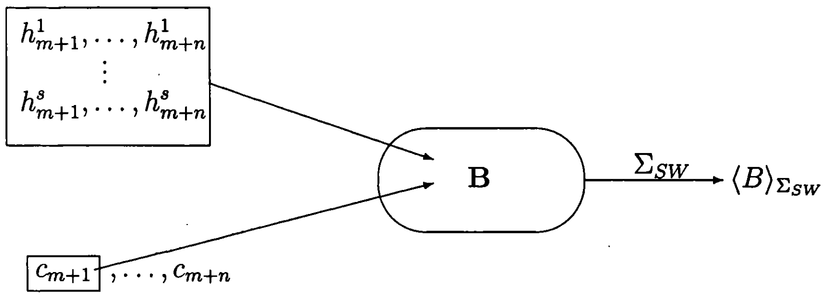
$h^1_{m+1}, \ldots, h^1_{m+n}$
$h^s_{m+1}, \ldots, h^s_{m+n}$
B
Σ_{SW}
$\langle B \rangle_{\Sigma_{SW}}$
$c_{m+1}, \ldots, c_{m+n}$

Figure 7.2: Producing a dense subset of $C(K, \mathbb{R})$

The set H contains separating functions, say,

$$h^1, \ldots, h^s$$

each h^i of the form

$$h^i = (h^i_1, \ldots, h^i_m, h^i_{m+1}, \ldots, h^i_{m+n})$$

for $1 \leq i \leq s$. Furthermore H contains non-zero constant functions, say,

$$c = (c_1, \ldots, c_m, c_{m+1}, \ldots, c_{m+n}),$$

where each coordinate function of c is a non-zero constant.

We build a subset $A \subset C(K, [0,1])$ containing

- the necessary separating functions from $h^1, \ldots, h^s$. That is, each of the coordinate functions $h^i_1, \ldots, h^i_m \in C(K, [0,1])$ for $1 \leq i \leq s$.
- one non-zero constant coordinate function, say, $c_1 \in C(K, [0,1])$ from the function c.

Similarly, we build a set $B \subset C(K, \mathbb{R})$ with the separating functions from $h^1, \ldots, h^s$ and a non-zero constant function c_{m+1}, as shown in Figure 7.2.

By Corollary 7.4.1 each of the coordinate functions $\phi_1, \ldots, \phi_m$ in $C(K, [0,1])$ can be approximated by functions in $\langle A \rangle_{\Sigma_{SW}}$, and each of the coordinate functions $\phi_{m+1}, \ldots, \phi_{m+n}$ can be approximated by functions in $\langle B \rangle_{\Sigma_{SW}}$ by the original Stone-Weierstrass Theorem 7.3.2.

That means that given any $q = m + n$ positive numbers

$$\frac{\epsilon_1}{\sqrt{q}}, \ldots, \frac{\epsilon_q}{\sqrt{q}} < \epsilon$$

we produce the functions $g_1, \ldots, g_m \in \langle A\rangle_{\Sigma_{SW}}$ and $g_{m+1}, \ldots, g_{m+n} \in \langle B\rangle_{\Sigma_{SW}}$ such that for all x in K

$$\|\phi_1 - g_1\|_\infty < \frac{\epsilon_1}{\sqrt{q}}, \cdots, \|\phi_n - g_q\|_\infty < \frac{\epsilon_q}{\sqrt{q}}.$$

Now, define the function $g : K \to C(K, [0,1]^m \times \mathbb{R}^n)$ as $g = (g_1, \ldots, g_{m+n})$, which is continuous by Lemma 7.4.1 and is in $\langle H\rangle_{\Sigma_{SW}}$. We have constructed $g \in C(K, [0,1]^m \times \mathbb{R}^n)$ in such a way that

$$\begin{aligned}
\|\phi - g\|_\infty &= \sup_{x\in K}\{\|\phi(x) - g(x)\|_2\} \\
&= \sup_{x\in K}\{\|(\phi_1(x), \ldots, \phi_q(x)) - (g_1(x), \ldots, g_q(x))\|_2\} \\
&= \sup_{x\in K}\{\|(\phi_1(x) - g_1(x), \ldots, \phi_q(x) - g_q(x))\|_2\} \\
&= \sup_{x\in K}\left\{\left[\sum_{i=1}^{q} |\phi_i(x) - g_i(x)|^2\right]^{\frac{1}{2}}\right\} \\
&< \left[\sum_{i=1}^{q}\left(\frac{\epsilon_i}{\sqrt{q}}\right)^2\right]^{\frac{1}{2}} \\
&< \frac{q \cdot \epsilon}{q} = \epsilon.
\end{aligned}$$

Hence for the given function $\phi \in C(K, [0,1]^m \times \mathbb{R}^n)$ and $\epsilon > 0$, we can approximate ϕ within ϵ using g. We therefore conclude $\langle H\rangle_{\Sigma_{SW}}$ is dense in $C(K, [0,1]^m \times \mathbb{R}^n)$. ■

7.4.3 The Stone-Weierstrass Over A Normed Vector Space

In addition to attributes of real numbers or tuples of real numbers, it will be useful to consider spatial objects that map points in space to data in a vector space. We define the vector space algebra as follows, where T_{Vector} are the vector space axioms listed in Chapter 3.

Algebra	V
Import	$\mathbb{R}$
Carriers	V
Operations	$+ : V \times V \to V$ $\cdot : \mathbb{R} \times V \to V$
Axioms	T_{Vector}

This section proves a density result due to [11](p.314) for spatial objects in the set

$$C(X, V) = \{\phi \mid \phi : X \to V \text{ is continuous}\}$$

which map points in a compact space X to a set of vectors V.

To approximate such objects, we will of course require a norm function $\| - \| : V \to \mathbb{R}$ on $C(X, V)$. To this end, we will require V to be a *normed vector space* (Recall the definition 3.8.19 in Preliminaries Section 3.8.4). Thus we define

$$\|v\| = \sup_{x \in K} \|v(x)\|$$

as usual.

7.4.3.1 Continuous Partitions of Unity

Definition 7.4.3 *[11](p.185) Let X be a topological space and let f be a real-valued function defined on X. The support of f, denoted Supp(f), is the smallest closed set $S \subset X$ such that $f(x) = 0$ for all $x \notin S$.*

Definition 7.4.4 *[11](p.186) Given a family $(A_i)_{i \in I}$ of subsets of a topological space X, a family $(u_i)_{i \in I}$ of real-valued functions defined on X is said to be subordinate to the family $(A_i)_{i \in I}$ if Supp(u_i) $\subset A_i$ for each index $i \in I$.*

Definition 7.4.5 *[11](p.186) A continuous partition of unity on X is any family $(u_i)_{i \in I}$ of positive real-valued continuous functions on X whose supports form a locally finite family and are such that $\sum_{i \in I} u_i(x) = 1$ for all points x in X.*

Lemma 7.4.2 *Given any open covering $(A_i)_{i \in I}$ of a paracompact set K there exists a continuous partition of unity $(f_i)_{i \in I}$ on x, subordinate to the covering $(A_i)_{i \in I}$*

PROOF This result is proved in [11](p.187). ∎

The following theorem and proof are due to [11](p.315).

Theorem 7.4.1 *Let K be a compact space, V a normed space over $\mathbb{R}$ and H a subset of $C(K, \mathbb{R})$. If H is dense in $C(K, \mathbb{R})$, then every continuous function $\phi : X \to V$ can be uniformly approximated by polynomials of the functions in H with coefficients in V.*

PROOF Given an $\epsilon > 0$, for each $x \in K$ there exists an open set of x in which the oscillation[1] of the given function ϕ is less than ϵ.

[1]The oscillation of a function ϕ on a subset $A \subset K$ is defined as $\sup_{x,y \in A} \|\phi(x) - \phi(y)\|$, [11](p.150) and intuitively means the largest distance between any two points of K

By the compactness of K there exists a finite open covering $\mathscr{A} = \{A_1, \ldots A_n\}$ of K such that the oscillation of ϕ in each A_i for $1 \leq i \leq n$ is less than ϵ. In symbols,

$$\sup_{x,y \in A_i} \|\phi(x) - \phi(y)\| < \epsilon, \text{ for } 1 \leq i \leq n.$$

Let $\mathbf{a}_i \in V$ be the value of ϕ at a point in A_i, $1 \leq i \leq n$, and the family $(u_i) = \{u_1, \ldots, u_n\}$ is a continuous partition of unity subordinate to the covering $\mathscr{A}$ (by Lemma 7.4.2), where $u_i : X \to \mathbb{R}$ with the following properties: Let z be any point in K. For every index $1 \leq i \leq n$, we have

$$\begin{aligned} z \notin A_i &\Rightarrow u_i(z) = 0, \\ z \in A_i &\Rightarrow \|\phi(z) - \mathbf{a}_i\| < \frac{\epsilon}{2n}. \end{aligned} \tag{7.9}$$

Then

$$\begin{aligned} \|\phi(x) - \sum_{i=1}^{n} \mathbf{a}_i u_i(z)\| &= \|\sum_{i=1}^{n}(\phi(z) - \mathbf{a}_i)u_i(z)\|, \text{ by algebraic properties} \\ &\leq \frac{\epsilon n}{2n}\sum_{i=1}^{n} u_i(x), \text{ by Equation 7.9} \\ &= \frac{\epsilon}{2}, \text{ by Definition 7.4.5.} \end{aligned}$$

By our assumption, H is dense in $C(K, \mathbb{R})$, and so there is a function v_i in H such that

$$|u_i(x) - v_i(x)| < \frac{\epsilon}{2n \cdot \|\sum_{j=1}^{n} \mathbf{a}_j\|}$$

for all $x \in K$ and $1 \leq i \leq n$. Observe that the values $\|\mathbf{a}_j\|$ for $1 \leq j \leq n$ are all positive real numbers, and so the summation is always positive. By the triangle inequality,

$$\begin{aligned} \|\phi(x) - \sum_{i=1}^{n} \mathbf{a}_i v_i(x)\| &\leq \|\phi(x) - \sum_{i=1}^{n} \mathbf{a}_i u_i(x)\| + \|\sum_{i=1}^{n} \mathbf{a}_i u_i(x) - \sum_{i=1}^{n} \mathbf{a}_i v_i(x)\| \\ &\leq \frac{\epsilon}{2} + \|\sum_{i=1}^{n}\Big(\mathbf{a}_i(u_i(x) - v_i(x))\Big)\| \\ &\leq \frac{\epsilon}{2} + \frac{n\epsilon\|\sum_{i=1}^{n} \mathbf{a}_i\|}{2n\|\sum_{j=1}^{n} \mathbf{a}_j\|} \\ &= \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ &= \epsilon. \end{aligned}$$

Thus we have shown that the polynomial $\sum_{i=1}^{n} \mathbf{a}_i v_i(x)$ of functions of H with coefficients of V can approximate ϕ to an arbitrary degree of precision, completing the proof. ■

7.5 Stone-Weierstrass Applied to CVG

We apply the main result of our work to Constructive Volume Geometry. That is, for some operations $F_1, \ldots, F_k$ that implement addition, multiplication and scalar multiplication and given a set B of basic CVG spatial objects and we would like to be able to describe (possibly by approximation) all other continuous CVG objects. For us to implement the Stone-Weierstrass Theorem 7.3.2, we must ensure that B

1. contains a non-zero constant spatial object,
2. is able to separate the points of E^3.

7.5.1 Volumetric Objects

In practical graphics computing, the objects which we describe must be finite, in the sense that their visual geometry is defined within some finite region of E^3.

We say that a scalar field $F : E^3 \longrightarrow S$ is *bounded* (compact) if there exists a bounded (compact) set $K \subset E^3$ such that

$$x \in E^3 - K \Rightarrow F(x) = c$$

where c is a scalar in S. In the case of the opacity scalar field, we set $c = 0$.

A spatial object is a *volumetric object* if there is a bounded set $K \subset E^3$ such that

$$x \in E^3 - K \Rightarrow O(x) = 0$$

7.5.2 The 4-Colour Channel Model

We show how the theory of Chapter 7 can be applied to Constructive Volume Geometry, specifically the 4-Colour Channel Model.

Our space, denoted E^3, is the three dimensional set $\mathbb{R}^3$ equipped with the Euclidean metric, defined as

$$|x - y| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

for points x, y in E^3. We generally refer to E^3 as the Euclidean space.

We define a norm on the set of attributes Opacity, Red, Green and Blue by

$$\|\mathbf{z}\|_2 = \sqrt{(z_o)^2 + (z_r)^2 + (z_g)^2 + (z_b)^2}$$

where $\mathbf{z} = (z_o, z_r, z_g, z_b)$ is a vector in the attribute space $[0,1] \times \mathbb{R}^3$.

Thus we are able to define the set

$$\mathscr{C}(\Sigma_{4cc}) \subseteq O(\Sigma_{4cc})$$

to be the set of all continuous CVG spatial objects mapping points in the Euclidean space E^3 to points in the attribute space $[0,1] \times \mathbb{R}^3$.

We define an algebra A_{SW} which contains the necessary operations for the Stone-Weierstrass Theorem. Each operation is pointwise lifted from the operations on the unit interval $[0,1]$ or the real numbers $\mathbb{R}$.

Algebra	A_{sw}
Import	A_{SO}
Carriers	$K = [a,b]^3 \subset E^3$, $C(K,[0,1])$, $C(K,\mathbb{R})$
Operations	
	$+_I : C(K,[0,1]) \times C(K,[0,1]) \to C(K,[0,1])$
	$\times_I : C(K,[0,1]) \times C(K,[0,1]) \to C(K,[0,1])$
	$\cdot_I : [0,1] \times C(K,[0,1]) \to C(K,[0,1])$
	$+ : C(K,\mathbb{R}) \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$
	$\times : C(K,\mathbb{R}) \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$
	$\cdot : \mathbb{R} \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$

We define a signature which incorporates the Stone-Weierstrass operations and the operations on the CVG spatial objects. The carrier set will be a subset of $\mathbf{O}(\Sigma_{4cc})$; namely the continuous CVG volumetric objects

$$\mathscr{C}(\Sigma_{4cc}).$$

We need our CVG objects to be specified within some bounded region $[-n,n]$ of E^3 for $n \geq 1$, since we will be applying the Stone-Weierstrass Theorem. The algebra and signature are defined as follows:

Signature	Σ_{4cc}^{sw}
Import	Σ_{4cc}, Σ_{sw}
Sorts	$\mathbf{C}(\Sigma_{4cc}) \subseteq \mathbf{O}(\Sigma_{4cc})$

Algebra	A_{4cc}^{sw}
Import	A_{4cc}, A_{sw}
Carriers	$\mathscr{C}(\Sigma_{4cc})$

Now we need some set B of basic spatial objects that contain a non-zero constant object, and one that separates the points of E^3. The constant function will simply be for every $z \in E^3$

$$o(z) = (1, 1, 1, 1)$$

which has the constant value 1 for each data attribute of Opacity, Red, Green and Blue.

To separate the points of E^3, the user of CVG has many possibilities. Let $z = (x_1, x_2, x_3)$ be a point in E^3. We could have, for example, the function

$$\phi(z) = (\frac{1}{1 + \sqrt{{x_1}^2 + {x_2}^2 + {x_3}^2}}, x_1, x_2, x_3).$$

This particular separating is rather strong. The user of our CVG framework need only guarantee that for any unique points $z \neq z'$ there is a function such that $\phi(z) \neq \phi(z')$, and so we could have many functions to separate the points, instead of just a single function[2].

We can use Corollary 7.4.2 for the CVG spatial objects in $\mathscr{C}(\Sigma_{4cc})$:

Corollary 7.5.1 *Let $[-n, n]^3$ be a closed and bounded cube of E^3, $n \geq 1$ and $\mathscr{C}(\Sigma_{4cc})$ the collection of continuous CVG spatial objects. If $B \subset \mathscr{C}(\Sigma_{4cc})$*

- *separates the points of space $[-n, n]^3$,*
- *contains a non-zero constant spatial object,*

then using the Σ_{4cc}^{sw}-operations on the CVG spatial objects in B we can approximate all CVG objects in $\mathscr{C}(\Sigma_{4cc})$ to any degree of precision, w.r.t the sup *norm.*

[2]The problem of selecting basic spatial objects and separating functions is important in CVG. We give a statement of this and related CVG problems that remain open in the concluding remarks of the thesis.

7.6 Adequacy

We have seen that the operations of addition, multiplication, scalar multiplication were required in the proof of Theorem 7.3.2. In fact, we can use any signature

$$\Sigma = (\Sigma; Data, SO; F_1, \dots, F_k)$$

such that the necessary Σ_{SW}-operations are Σ-term definable. i.e. expressible by the terms in $T(\Sigma, Y)$, where Y is a set of variables. Specifically, by Definition 4.3.5, this means that for a Σ-algebra A, the operation $+ : A \times A \to A$ is Σ-term definable if there exists a term $s(y_1, y_2) \in T(\Sigma, Y)$ such that for all $a_1, a_2 \in A$

$$a_1 + a_2 = [\![s(y_1, y_2)]\!]_A(a_1, a_2),$$

and similarly for the other Σ_{SW} operations.

Corollary 7.6.1 *Let $C(K, \mathbb{R})$ be the set of all continuous spatial objects where K is a compact space. Define the signature*

$$\Sigma = (\Sigma; Data, SO; F_1, \dots, F_k)$$

such that the operations $+$, $\times$ and $\cdot$ are Σ-definable. Then for any subset $B \subseteq C(K, \mathbb{R})$ which

1. *separates points in K, and*

2. *contains a non-zero constant spatial object,*

the Σ-subalgebra $\overline{\langle B \rangle}_{\Sigma_{SW}}$ is dense in $C(K, \mathbb{R})$.

PROOF Suppose that the operations $+$, $\times$ and $\cdot$ are Σ-term definable. Then by Definition 4.3.5 there exists terms $s(y_1, y_2)$, $t(y_1, y_2)$ and $r(y_1, y_2)$ in $T(\Sigma, Y)$ such that

$$\begin{aligned} +(\phi_1, \phi_2) &= [\![s(y_1, y_2)]\!]_{C(K,\mathbb{R})}(\phi_1, \phi_2), \\ \times(\phi_1, \phi_2) &= [\![t(y_1, y_2)]\!]_{C(K,\mathbb{R})}(\phi_1, \phi_2), \\ \cdot(\lambda, \phi_2) &= [\![r(y_1, y_2)]\!]_{C(K,\mathbb{R})}(\lambda, \phi_2). \end{aligned}$$

for all $\phi_1, \phi_2 \in C(K, \mathbb{R})$ and $\lambda \in \mathbb{R}$. Define a Σ'_{SW}-algebra

Algebra	A_{SW}
Import	$\mathbb{R}$
Carriers	K, B
Operations	
	$+ : C(K,\mathbb{R}) \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$
	$\times : C(K,\mathbb{R}) \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$
	$\cdot : \mathbb{R} \times C(K,\mathbb{R}) \to C(K,\mathbb{R})$
Definitions	$+(\phi_1,\phi_2) = [\![s(y_1,y_2)]\!]_{C(X,A)}(\phi_1,\phi_2)$
	$\times(\phi_1,\phi_2) = [\![t(y_1,y_2)]\!]_{C(K,\mathbb{R})}(\phi_1,\phi_2)$
	$\cdot(\lambda,\phi_2) = [\![r(y_1,y_2)]\!]_{C(K,\mathbb{R})}(\lambda,\phi_2)$

containing the imported data set $\mathbb{R}$, carriers K and $B \subset C(K,\mathbb{R})$ of spatial objects satisfying conditions 1 and 2 of the Theorem and the new operations based on the evaluation of Σ-terms s, t and r. By the Stone-Weierstrass Theorem 7.3.2 $\langle B'\rangle_{\Sigma'_{SW}}$ is dense in $C(K,\mathbb{R})$ and furthermore, by Lemma 4.3.3

$$\langle B'\rangle_{\Sigma'_{SW}} \subseteq \langle B\rangle_{\Sigma}$$

and therefore $\langle B\rangle_{\Sigma}$ dense in $C(K,\mathbb{R})$. ∎

7.7 Dense subsets of $C(X,\mathbb{R})$, X non-compact

The Stone-Weierstrass Theorem shows that, assuming certain properties of the set B containing the basic spatial objects, $\langle B\rangle_{\Sigma_{SW}}$ is dense in $C(X,\mathbb{R})$ when X is a compact space.

We would like to approximate objects over non-compact spaces. Take for example $X = E^3$: can we find a dense subset of $\mathscr{C}(E^3,\mathbb{R}^3)$? The answer is yes, and we use the theory developed in Chapter 6 to formulate and prove this result.

7.7.1 Homomorphisms and the Inverse Limit

Recalling the inverse limit construction of the previous chapter, we defined an inverse limit on the family $C(\mathscr{K},\mathbb{R})$ using the directed set $\mathscr{K}$ of all compact subspaces of X and showed that the compact-open topology behaves "the same" on $\varprojlim C(\mathscr{K},\mathbb{R})$ as it does on $C(X,\mathbb{R})$. This was achieved in Theorem 6.8.1 by showing there exists a homeomorphism mapping the inverse limit element (f) to the union of all restricted functions $\bigcup_{K\in\mathscr{K}} f_K : X \to \mathbb{R}$. We will use its inverse

$$\Phi : C(X,\mathbb{R}) \to \varprojlim C(\mathscr{K},\mathbb{R})$$

where $\Phi(f) = (f)$ maps a function f to a sequence (f) in the inverse limit where $f_{|K} = (f)(K)$. That is, each coordinate K of the element (f) is the restricted function $f_{|K}$ to $C(K,\mathbb{R})$. Now consider a function

$$\Phi_K : C(X,\mathbb{R}) \to C(K,\mathbb{R})$$

defined by the composition $\Phi_K = (\phi_K^\infty \circ \Phi)$ where ϕ_K^∞ is the homomorphic projection from the inverse limit to the coordinate space $C(K,\mathbb{R})$. This function is simply a restriction function restricting functions in $C(X,\mathbb{R})$ to $C(K,\mathbb{R})$ and is clearly a homomorphism since it is the composition of homomorphisms. We display Φ_K as a commutative diagram in Figure 7.3.

$$\begin{tikzcd}
C(X,\mathbb{R}) \arrow[r, "\Phi"] \arrow[dr, "\Phi_K"'] & \varprojlim C(\mathscr{K},\mathbb{R}) \arrow[d, "\phi_K^\infty"] \\
 & C(K,\mathbb{R})
\end{tikzcd}$$

Figure 7.3: A homomorphism from $C(X,\mathbb{R})$ to $C(K,\mathbb{R})$

7.7.2 Approximation on $C(X,\mathbb{R})$, X non-compact

Now $\mathbb{R}$ will be a topological Σ_{SW}-algebra of the real numbers with the Stone-Weierstrass operations as specified by Σ_{SW}. We shall use Φ_K to prove a density result for $C(X,\mathbb{R})$ when X is not compact.

Our first concern is how the basic spatial objects satisfying the necessary conditions for the Stone-Weierstrass Theorem behave under the map Φ_K. Recall the properties a set $B = \{b_1,\dots,b_n\}$ must satisfy for the Stone-Weierstrass Theorem are

1. separating the points of space X,

2. containing a non-zero constant spatial object,

and we must show that the map Φ_K preserves these properties.

Clearly if a function $b_i \in C(X,\mathbb{R})$ is constant, then it must be constant on a restriction $b_{i|K}$ to a subset K of X. Therefore Φ_K preserves constant functions. Furthermore, if the functions $b_1,\dots,b_n$ separates points of X then we can observe that the restrictions $b_{1|K}, \ldots, b_{n|K}$ separate points of the subset K of X. To summarize, we make the following observation:

Observation 7.7.1 *If B satisfies Properties 1 and 2 on $C(X,\mathbb{R})$, then $\Phi_K(B)$ satisfies these properties on the space $C(K,\mathbb{R})$, where $K \subset X$.*

Bearing this in mind, we turn to the main result of this section:

Theorem 7.7.1 (Approximation Theorem for Spatial Objects) *Let X be a topological space and $C(X,\mathbb{R})$ the collection of continuous total spatial objects with the compact-open topology. If $B = \{b_1, \ldots, b_n\}$ is a subset of $C(X,\mathbb{R})$ which*

1. *separates the points of space X,*
2. *contains a non-zero constant spatial object,*

then the Σ_{SW}-subalgebra $\langle B\rangle_{\Sigma_{SW}}$ generated by repeated application of Σ_{SW}-operations on the spatial objects in B is dense in $C(X,\mathbb{R})$.

Proof Choose any basic open set $W(K,U)$ of the compact-open topology on $C(X,\mathbb{R})$. We show that $\langle B\rangle_{\Sigma_{SW}} \cap W(K,U) \neq \emptyset$.

For the given compact $K \subset X$, we have the homomorphism Φ_K and by Observation 7.7.1 the set $\Phi_K(B)$ satisfies Properties 1 and 2 on $C(K,\mathbb{R})$. By the Stone-Weierstrass Theorem 7.3.2 we conclude

$$W(K,U) \cap \langle \Phi_K(B)\rangle_{\Sigma_{SW}} \neq \emptyset$$

on $C(K,\mathbb{R})$. This means that there is a term $t \in \langle \Phi_K(B)\rangle_{\Sigma_{SW}}$ such that when evaluated on $C(K,\mathbb{R})$ yields a function

$$[\![t]\!]_{C(K,\mathbb{R})}(\Phi_K(b_1), \ldots, \Phi_K(b_n))$$

in $W(K,U)$ for the basic spatial objects $b_1, \ldots, b_n$ in B restricted to K by Φ_K. But the homomorphic properties of Φ_K gives the equation

$$[\![t]\!]_{C(K,\mathbb{R})}(\Phi_K(b_1), \ldots, \Phi_K(b_n)) = \Phi_K([\![t]\!]_{C(X,\mathbb{R})}(b_1, \ldots, b_n)).$$

This equation means that the term t is evaluated to the same points in $C(X,\mathbb{R})$. That is, $[\![t]\!]_{C(X,\mathbb{R})}[K] \subset U$ and hence is a spatial object in $W(K,U)$, completing the proof. ∎

The reader should note that the sup norm on $C(X,\mathbb{R})$ may not be defined. However, when we are approximating, we are approximating locally. That is, on some compact subspace K of X. Therefore, we are able to meaningfully use the Stone-Weierstrass Theorem.

7.8 Notes and Sources

The Stone-Weierstrass Theorem is an important topological result and gives conditions necessary for a set to be dense in $C(K, \mathbb{R})$. Thus it is a natural tool for us to answer the expressiveness and completeness problem.

Simmons [52] gives a version of the original Weierstrass Theorem for real valued functions which shows that special polynomials called *Bernstein Polynomials* can be used to approximate any continuous real valued function defined on a closed interval to an arbitrary degree of accuracy.

The full Stone-Weierstrass Theorem is given in Simmons which approximates real valued functions on a compact Hausdorff space X using the properties of lattices and this is the approach I have used for my thesis. Other approaches to the Stone-Weierstrass Theorem are found in Kelley [36] and Dugundji [26].

The Stone-Weierstrass Theorem is extended in my thesis to the case of functions that return n-tuples of real numbers [47], functions on the unit interval, and functions that are of the type $f : X \rightarrow [0,1]^m \times \mathbb{R}^n$.

Bourbaki [11] gives a Stone-Weierstrass result for which functions are of the form $f : X \rightarrow V$, where V is a normed vector space. This result is particularly useful for Constructive Volume Geometry, where the CVG spatial objects map points in E^3 to data that contains a function to describe a physical phenomenon at each point in the space, such as reflection.

Chapter 8

Conclusion

This thesis provides a mathematical foundation for the theory of spatial data types. A framework was defined which mathematically models spatial objects by total functions. Such an approach was shown to be useful in many situations. The example we have focused on in this work is Constructive Volume Geometry. An algebraic framework for the specification of CVG objects and operations was defined, and we specified some particularly useful classes of CVG objects.

Spatial objects were shown to be a helpful tool in defining models of computation. An abstract machine model was developed which executed programs over a Σ-algebra A storing values in a set R of registers. Then the set of all register configurations was simply $F(R, A)$, the set of spatial objects where the data A is distributed in the space R.

The value of studying these examples is that both are vastly different applications, but share the same underlying mathematical theory. That is, our mathematical theory of spatial data types can unify continuous and discrete space. In particular, the space we used in CVG is three dimensional Euclidean space E^3 which is continuous and rather complicated compared to the discrete space R of registers we used in the machine example. Both are handled under the same theory.

We solved a general completeness problem for continuous spatial objects in $C(X, A)$, containing all functions mapping a topological space X to a topological space/algebra A. We considered the subalgebra $\langle B\rangle_{\Sigma_{SW}}$ generated by the application of some Σ-operations on basic spatial objects in B. This set was enough to generate all spatial objects of interest, possibly by approximation.

Our method of approximation is the Stone-Weierstrass Theorem, a result from classical analysis. We modified this result and used it to show that $\langle B\rangle_{\Sigma_{SW}}$ is dense in $C(X, A)$. That is, for any spatial object in $C(X, A)$, there is a spatial object in $\langle B\rangle_{\Sigma_{SW}}$ arbitrary close to it. The benefits of such a result is that for a given spatial object, we can *construct* a term such that when evaluated, approximated the given object to an arbitrary degree of accuracy. This result has a strong impact on a theory of computation for spatial objects and we discuss this in the next section.

8.1 Possible Directions

The demands of our general theory has led us to look for general results of wide applicability. A number of areas are of immediate interest:

Constructive Volume Geometry

Constructive Volume Geometry is an algebraic framework for the specification, representation and manipulation of graphical objects in three dimensions. The area is a rich source of theoretical problems in its own right. For example, we need to find a finite set of operations Σ on some set B of *basic CVG objects* that can generate a set $\langle B\rangle_\Sigma$ which was suitable to approximate any CVG object. These CVG operations must be chosen carefully in a way that they are reasonable to the computer graphics community and are suitable to be used in this application domain.

A rigorous reformulation of Constructive Volume Geometry, adhering to the strict mathematical formalism of this thesis, may generate a number of interesting problems, for example, the development of a theoretical rendering process. That is, study maps of the type

$$r : F(\mathbb{E}^3, A) \to F(\mathbb{Z}^2, A)$$

and find algebraic structures such that r can be defined by structural induction. Also of interest is the method of *digitisation* of continuous data from medical devices (such as Computerised Tomography (CT) scanners). Such a transformation can be modelled algebraically by a mapping

$$D : C(E^3, \mathbb{R}) \to C(\mathbb{Z}^2, \mathbb{Z})$$

from the set of functions in continuous space E^3 to functions in discrete space $\mathbb{Z}^2$. This involves both a translation $d_1 : E^3 \to \mathbb{Z}^2$ of the space and a translation $d_2 : \mathbb{R} \to \mathbb{Z}$ of the data.

Computability

A theory of computation for programming with spatial objects is still very much in its preliminary stages. Spatial objects are an example a *continuous data type*. They are modelled by topological algebras with primitive operations that are continuous on the data on which they are defined.

Broadly speaking, there are two models of computation to consider:

- *Abstract models* are high-level programming models which do not depend on the specific data representation of the Σ-algebra A. Computations are uniform over all Σ-algebras. Once we have chosen some data and operations on that data, we are able to compute with programs. Such models include the While programming language [59] or the SERM programming language of Chapter 5.

- *Concrete models* are models in which computations are dependent on the representation of the data over which we are computing. There are many such models computing on the real numbers. An excellent overview of some common models is found in [53].

In either model computation over spatial objects will not be exact, and so one must consider methods of approximation. Further research will consider various methods of approximation. We aim to control and compare such methods.

Now, we have used Stone-Weierstrass Theorem extensively in this work to approximate spatial objects. An important consideration is the computational aspects of this theorem as a way to extend and further the computability and complexity properties of spatial data types. We could consider an effective version of the Stone-Weierstrass Theorem: Given a spatial object $\phi : K \longrightarrow \mathbb{R}$ and an error margin $\epsilon > 0$ we would like to *compute* a term $t \in T(\Sigma, Y)$ such that

$$\left\|\phi - [\![t(b_1, \ldots, b_n)]\!]_{C(K,\mathbb{R})}\right\|_{\infty} < \epsilon$$

for the basic spatial objects $b_1, \ldots, b_n$. Of particular interest is the constructive Stone-Weierstrass Theorem of Bishop and Bridges [8]. Such a result will be useful for the general theory of spatial objects and to the application of CVG. Furthermore, it is important to study different methods of Stone-Weierstrass to show the impact on results we have developed in this thesis.

Specification

Spatial objects are higher-order objects, and although we have used the algebraic theory of data types, our main task has been to add topologies and consider approximation and computability. We have not considered the axiomatic specification theory. Algebras of spatial objects are higher order algebras and the algebraic specification theory is an advanced and fascinating topic.

It might be interesting to develop a higher order algebraic specification theory of

1. Spatial object algebras in general,

2. CVG algebras in particular.

The basic theory of high-order algebras has been worked out by [40] and some interesting work that has been done for specifying hardware includes [54].

Bibliography

[1] A. Abdul-Rahman. *Physically-Based Rendering and Algebraic Manipulation of Volume Models.* PhD thesis, University of Wales, Swansea, 2006.

[2] A. Abdul-Rahman and M. Chen. Spectral volume rendering based on the Kubelka-Munk theory. *Computer Graphics Forum*, 24(3):413–422, 2005.

[3] R. F. Arens. A topology for spaces of transformations. *Annals of Mathematics*, 47(3):480–495, 1946.

[4] R. F. Arens and J. L. Kelley. Characterizations of the space of continuous functions over a compact Hausdorff space. *Transactions of the American Mathematical Society*, 82(3):499–508, 1947.

[5] J. P. Aubin. *Applied Abstract Analysis.* John Wiley & Sons, 1977.

[6] M. P. Backer and D. Hearn. *Computer Graphics C Version.* Prentice Hall International, international edition, 1997.

[7] B. Banaschewski and C. J. Mulvey. A constructive proof of the Stone-Weierstrass theorem. *Journal of Pure and Applied Algebra*, 116:25–40, 1997.

[8] E. Bishop and D. Bridges. *Constructive Analysis.* Springer-Verlag, 1985.

[9] J. Blanck, V. Stoltenberg-Hansen, and J. V. Tucker. Domain representations of partial functions, with applications to spatial objects and constructive volume geometry. *Theoretical Computer Science*, 284:207–224, 2002.

[10] N. J. Bloch. *Abstract Algebra with Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[11] N. Bourbaki. *General Topology Part 1 and 2.* Springer, 1966.

[12] D. S. Bridges. *Computability: A Mathematical Sketchbook.* Springer-Verlag, New York, 1994.

[13] C. W. Burrill. *Foundations of Real Numbers.* McGraw-Hill, New York, 1967.

[14] M. Chen, A. E. Kaufman, and R. Yagel, editors. *Volume Graphics.* Springer, London, 2000.

[15] M. Chen and A. Leu. Direct rendering algorithms for complex volumetric scenes. *Proc. 16th Eurographics UK Conference, Leeds*, pages 1–15, 1998.

[16] M. Chen and A. Leu. Modelling and rendering graphics scenes composed of multiple volumetric datasets. *Computer Graphics Forum*, 18(2):159–171, 1999.

[17] M. Chen, D. Rodgman, A. S. Winter, and S. M. F. Treavett. Enriching volume modelling with scalar fields. *Data Visualization: the State of the Art*, pages 345–362, 2003.

[18] M. Chen and J. V. Tucker. Constructive volume geometry. Technical report, University of Wales, Swansea, 1998.

[19] M. Chen and J. V. Tucker. Constructive volume geometry. *Computer Graphics Forum*, 19(4):281–293, 2000.

[20] M. Chen, J. V. Tucker, and A. Leu. Trove - a rendering system for constructive representations of volumetric environments. In M. Chen, A. E. Kaufman, and R. Yagel, editors, *Volume Graphics*, chapter 6. Springer, 2000.

[21] M. Chen and A. S. Winter. vlib: A volume graphics API. In Mueller and A. E. Kaufman, editors, *Volume Graphics*, pages 133–147. Springer, 2001.

[22] H. F. Cullen. *Introduction to General Topology.* D.C. Heath and Company, Boston, 1968.

[23] N. J. Cutland. *Computability, An Introduction to Recursive Function Theory.* Cambridge University Press, Cambridge, 1980.

[24] B. A. Davey and H. A. Priestly. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, 1990.

[25] L. Debnathm and P. Mikusiński. *Introduction to Hilbert Spaces with Applications.* Academic Press, San Diego, California, second edition, 1999.

[26] J. Dugundji. *Topology.* Allyn and Bacon Inc., Boston-London-Sydney-Toronto, 1966.

[27] R. Engelking. *General Topology.* Heldermann Verlag, Germany, 1989.

[28] J. D. Foley. *Computer Graphics: Principles and Practices.* Addison-Wesley, second edition, 1997.

[29] H. Friedman. Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory. In *Logic Colloquium '69*, pages 361–389. North-Holland, 1971.

[30] S. A. Gaal. *Point Set Topology.* Academic Press, New York, 1964.

[31] G. Y. Gardner. Simulation of natural scenes using textured quadric surfaces. *ACM/SIGGRAPH Computer Graphics*, 18(3):11–20, 1984.

[32] G. Grätzer. *Universal Algebra.* D. Van Nostrand Company Inc., Princeton, New Jersey, 1968.

[33] G. Grätzer. *Universal Algebra.* Springer-Verlag, Berlin, second edition, 1979.

[34] K. Hoffman. *Analysis in Euclidean Space.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[35] K. Johnson. The theory of spatial data types and constructive volume geometry. In *Bulletin of the European Association for Theoretical Computer Science*, volume 89, page 198, 2006.

[36] J. L. Kelley. *General Topology.* Springer-Verlag, New York, 1975.

[37] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[38] M. Mansfield. *Introduction to Topology.* D. Van Nostrand Company Inc., Princeton, New Jersey, 1966.

[39] N. H. McCoy. *Rings and Ideals.* The Mathematical Association of America, 1948.

[40] K. Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100:385–417, 1992.

[41] K. Meinke. Topological methods for algebraic specification. *Theoretical Computer Science*, 166:263–290, 1996.

[42] K. Meinke and J. Steggles. Specification and verification in higher order algebra: A case study of convolution. In J. Heering, K. Meinke, B. Moller, and T. Nipkow, editors, *Higher Order Algebra, Logic and Term Rewriting, Lecture Notes in Computer Science*, volume 816, pages 189–222. Springer-Verlag, 1994.

[43] K. Meinke and J. V. Tucker. *Universal Algebra*, pages 189–411. Handbook of Logic for Computer Science. Oxford University Press, Oxford, 1992.

[44] C. B. Morrey and M. H. Protter. *A First Course in Real Analysis.* Springer-Verlag, New York, 1977.

[45] J. Myhill. A complete theory of natural, rational, and real numbers. *The Journal of Symbolic Logic*, 15(3):185–196, 1950.

[46] J. Myhill. What is a real number? *The American Mathematical Monthly*, 79(7):748–754, 1972.

[47] G. L. Naber. *Topological Methods in Euclidean Spaces.* Cambridge University Press, Cambridge, 1980.

[48] T. Porter and T. Duff. Compositing digital images. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, New York, NY, USA, 1984. ACM Press.

[49] A. A. G. Requicha. Mathematical models of rigid solid objects. Technical Report 28, University of Rochester, Rochester, New York, November 1977.

[50] A. A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys*, 12(4):437–464, December 1980.

[51] H. Schubert. *Topology.* Macdonald Technical & Scientific, London, 1968.

[52] G. F. Simmons. *Introduction to Topology and Modern Analysis.* Krieger Publishing Company, Malabae, Florida, 1983.

[53] D. Spreen. On some problems in computable topology. In *Logic Colloquium '05, Lecture Notes in Logic*, volume 28. Association for Symbolic Logic, 2007. to appear.

[54] L. J. Steggles. Verifying an infinite systolic algorithm using third-order equational methods. *Journal of Algebraic and Logic Programming*, 69(1-2):75–92, 2006.

[55] K. Stephenson and J. V. Tucker. Data, syntax, and semantics, an introduction to modelling programming languages. University of Wales, Swansea, to appear.

[56] V. Stoltenberg-Hansen and J. V. Tucker. Algebraic and fixed point equations over inverse limits of algebras. *Theoretical Computer Science*, 87(1):1–24, 1991.

[57] V. Stoltenberg-Hansen and J. V. Tucker. Concrete models of computation for topological algebras. *Theoretical Computer Science*, 219(1-2):347–378, 1999.

[58] M. H. Stone. The generalized Weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948.

[59] J. V. Tucker and J. I. Zucker. *Computable functions and semicomputable sets on many sorted algebras*, volume 5 of *Handbook of Logic for Computer Science*, pages 317–523. Oxford University Press, Oxford, 1998.

[60] J. V. Tucker and J. I. Zucker. Computation by while programs on topological partial algebras. *Theoretical Computer Science*, 219(1-2):379–420, 1999.

[61] J. V. Tucker and J. I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computer Logic*, 5(4):611–668, 2004.

[62] J. V. Tucker and J. I. Zucker. Computable total functions, algebraic specifications and dynamical systems. *Journal of Algebraic and Logic Programming*, 62:71–108, 2005.

[63] J. V. Tucker and J. I. Zucker. Abstract versus concrete computability: The case of countable algebras. In V. Stoltenberg-Hansen and J. Vnnen, editors, *Logic Colloquium '03, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, Helsinki, August 2003*, number 24 in Lecture Notes in Logic, pages 377–408, 2006.

[64] W. Wechler. *Universal Algebra for Computer Scientists.* EATCS Monographs on Theoretical Computer Science. Springer, Berlin, 1992.

[65] K. Weihrauch. *Computable Analysis, an Introduction.* Springer, 2000.