The Algebraic Properties of `if-then-else` with Commutative Three-Valued Tests

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Master of Science in Mathematics

at

The University of Waikato

by

Roger Chi-Wei Su

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2018

Abstract

This thesis studies an algebraic model of computable programs and the `if-then-else` operation. The programs here are considered deterministic, but not assumed to be always halting, so they are modelled by a semigroup of partial functions, with several extra operations in addition to the original binary operation of the semigroup.

The `if-then-else` operation involves not only programs, but logical tests too. Hence, it calls for a separate algebra of tests. Evaluating a test often requires running another program, so the tests are also possibly non-halting. When tests do not always halt, the results of conjunctions (logical 'and') and disjunctions (logical 'or') can differ, depending on whether sequential or parallel evaluation is applied. The parallel evaluation is what this thesis adopts.

The overall 'program algebra' consists of two sorts, one of programs and the other of tests. Each sort has its own operations, and there are hybrid operations such as `if-then-else` which involve both sorts. This thesis establishes the axioms of all these operations by building an embedding from the abstract program algebra into a concrete one. At the end is a discussion on the algebra of tests without the programs, where the differences between the two evaluation paradigms are explored in detail.

Acknowledgements

To my supervisor, Dr. Tim Stokes, am I most grateful. Throughout this past year, he has shared his vast knowledge and given me insightful guidance on the academic subjects, and has provided me much friendly advice on the practice of research. In addition, his courses I took in my undergraduate days were among the sparks which ignited my academic passion, and I thank him for this too.

I am much obliged to Assoc. Prof. Stephen Joe. Ever since I started university five years ago, I somehow kept getting myself embroiled in enrolment issues, but he has patiently cleared away those troubles with deep expertise and experience with the university regulations. I also thank him for his powerful LaTeX package, which has given this thesis its elegant design with little of my effort.

Mrs. Glenys Williams also deserves special mention. After 25 years as the secretary of the Department of Mathematics, she has recently left her post and embarked afresh. I am indebted to her for all her kindness and administrative help, especially for letting me from time to time escape the dreadful windowless students' study, and shelter in the sunny breezy meeting room. I wish her all the best with her new phase in life!

Of course, my gratitude extends to everyone in the department (and the faculty) too. Many thanks for having taught me, or just being friendly company!

On the institutional level, I thank the Faculty of Computing and Mathematical Sciences for the espresso machine and the endless supply of tea and milk and coffee, and I am also grateful to the University of Waikato for providing financial assistance with the Research Masters Scholarship.

Finally, I sincerely express my gratitude to my family for their confidence in me and their constant support.

Contents

Chapter 1

Introduction

The 1950s saw the birth of high-level programming languages such as `FORTRAN` and `ALGOL`. These languages contain the conditional statement of `if-then-else`, a feature which has prevailed to this day. An `if-then-else` statement consists of a *test* (or *guard*) α, which is simply a logical proposition, and two programs p and q. The statement

$$\texttt{if } (\alpha) \texttt{ then } \{p\} \texttt{ else } \{q\}$$

says that 'if α is true, apply p; otherwise, apply q', and it will be denoted as $\alpha[p, q]$ throughout this thesis. This suggests that modelling `if-then-else` involves bridging the world of tests and the world of programs. The tests, being logical propositions, have operations such as 'and', 'or' and 'not'. The programs, on the other hand, have the concatenation operation which executes one program after another.

In the practical setting, programs do not always successfully terminate, and they may continue running indefinitely. Evaluating a test often invokes another program, so just like programs, tests may not halt either. Hence, the result of a test may be neither true nor false, so rather than the classical two-valued logic, algebraic models of programs often use three-valued tests (which will be explained in detail later).

Not long after the emergence of these programming languages, in 1963, McCarthy (himself one of the designers of `ALGOL`) proposed a set of axioms for `if-then-else` in [27]. He treated both programs and tests as the same type (*i.e.* one-sorted), with the values 'true' T and 'false' F being distinguished elements. This model is rather limited because it only involves `if-then-else` , and neither logical connectives on the tests nor concatenation of programs are treated.

Later on in 1983, Bloom and Tindell picked up this subject in [4], and studied the axiomatisation of four one-sorted variants of `if-then-else` . Similar to McCarthy, Bloom and Tindell only focused on the `if-then-else` operation. Afterwards, their work was further generalised and elaborated by Guessarian and Meseguer [13], Melker and Nelson [28], and Pigozzi [30]. These works all carry a logical flavour, with an emphasis on terms, proof systems, syntactic consequence ($\vdash$), semantic entailment ($\vDash$) and so on.

In the 1990s, Bergman [1] and Manes [26] initiated two lines of research. Bergman used sheaf theory to study the action of a Boolean algebra (tests) on a set (programs), and Manes used category theory to study `if-then-else` with both two- and three-valued tests. Both of these two works focus on how the logical connectives interact with the `if-then-else` operation, but do not cover program concatenation.

Around 2000, Kozen [23] and Desharnais *et al.* [9] took a relational perspective. They viewed both programs and tests as binary relations, and modelled them by a Kleene algebra. They not only treated logical connectives and program concatenation; with the closure operator in a Kleene algebra, they also modelled the iterative `while-do` construct. However, this approach yielded no finite axiomatisation.

Finally, there is the two-sorted algebraic approach started by Jackson and Stokes [17, 19]. These authors first studied a version similar to those of Bergman and Manes, with logical connectives on two-valued tests but no program concatenation. The novelty of their works was that they established the axioms purely algebraically. Using the same method, they also studied a more furnished version, where both logical connectives (still on two-valued tests) and program concatenation were treated.

More recently, Panicker *et al.* [29] generalised Jackson and Stokes' work [17] to three-valued tests, and they viewed the binary connectives 'and' and 'or' as being evaluated sequentially.

Sequential evaluation, also known as short-circuit evaluation, starts by evaluating the first argument, and proceeds to evaluate the second argument only if the first is not enough to determine the overall value. If the first argument has not halted, the entire statement would not halt either, even if the second argument alone can determine the result. Therefore, in the presence of non-halting tests, the result of sequential evaluation depends on the order of the arguments, and the binary connectives become non-commutative. This sequential evaluation strategy is the paradigm widely used in real-life programming languages.

On the other hand, the binary connectives can also be evaluated in parallel. This strategy looks at both arguments, and the evaluation would finish as soon as one of the arguments halts and is sufficient to determine the overall result. The order of the arguments does not matter when parallel evaluation strategy is employed, so the binary connectives are commutative.

This thesis will follow the algebraic approach by Jackson and Stokes, and generalise their works to three-valued tests. However, opposite to Panicker *et al.*, this thesis will view the connectives as being evaluated in parallel.

Organisation of this Thesis This thesis begins with the preliminaries in Chapter 2. Then, Chapter 3 introduces the algebraic structures and operations which will be studied, and explains the motivations behind them.

Next come the chief parts of this thesis. Chapter 4 studies the algebra of the programs alone, and Chapter 5 brings in the tests and the `if-then-else` operation. Afterwards, Chapter 6 covers the equality test, a particular type of tests which is commonly seen and hence deserves attention.

Finally, Chapter 7 ends the thesis by looking at the algebra of tests on its own, without the presence of the programs. It covers the historical background of different systems of three-valued logic, and then establishes the type of algebras which correspond to the commutative three-valued logic used in this thesis.

Chapter 2

Preliminaries

This chapter will begin with some fundamental concepts of relations and functions. Then, it will outline the basics of universal algebra, semigroups and lattices. At the end, this chapter will also discuss many-sorted algebras and axiomatisation.

Definition 2.1. A **(binary) relation** θ between the sets X and Y is a set of ordered pairs:

$$\{(x, y) \mid x \in X, y \in Y\}.$$

Elements $x \in X$ and $y \in Y$ are said to be related by θ if and only if $(x, y) \in \theta$, which can also be written as $x\ \theta\ y$.

A relation *on* X means that it is defined between X and X. There are two important special types of relations on a single set X — equivalence relations and partial orders.

Definition 2.2. The relation $\sim$ on X is an **equivalence relation** when it satisfies the following three criteria for all $x, y, z \in X$:

- $x \sim x$; (Reflexive)
- $x \sim y$ if and only if $y \sim x$; (Symmetric)
- $x \sim y$ and $y \sim z$ implies $x \sim z$. (Transitive)

The most trivial example of an equivalence relation is equality.

When the set X has an equivalence relation $\sim$, the **equivalence class** of some $x \in X$ is defined to be $\overline{x} = \{y \in X \mid x \sim y\}$. All of the equivalence classes are disjoint, and their union is the whole of X, so the set of equivalence classes in fact forms a **partition** of the original set X:

$$X/{\sim} = \{\overline{x} \mid x \in X\}.$$

Definition 2.3. The relation $\leq$ on X is a **partial order** when it satisfies the following three criteria for all $x, y, z \in X$:

- $x \leq x$; (Reflexive)
- $x \leq y$ and $y \leq x$ imply $x = y$; (Anti-Symmetric)
- $x \leq y$ and $y \leq z$ imply $x \leq z$. (Transitive)

A set X with a partial order $\leq$ is called a **partially ordered set** $\langle X, \leq \rangle$, or a **poset**.

The most common example of a partial order is the 'less-than-or-equal-to' relation on the set of real numbers.

Definition 2.4. The relation f between X and Y is called a **(partial) function** from X to Y when it satisfies the following criterion for all $x \in X$ and $y_1, y_2 \in Y$:

$$(x, y_1) \in f \text{ and } (x, y_2) \in f \text{ imply } y_1 = y_2.$$

When $(x, y) \in f$, this criterion enables y to be unambiguously written as xf. Furthermore, an ordered pair (x, y) in a function is also called a maplet.

Given a function f from X to Y, the **domain** of f, denoted $dom(f)$, is the following subset of X:

$$\{x \in X \mid \exists y \in Y \ : \ (x, y) \in f\}.$$

Definition 2.5. The function g between X and Y is a **total function** (or **transformation**) when $dom(g) = X$.

Finally, the following are some notes on the notations used in this thesis.

Notation 2.6. Let X be a set. Then,

- $\mathcal{P}(X)$ denotes the set of all (partial) functions from X to itself;
- $\mathcal{T}(X)$ denotes the set of all total functions from X to itself.

Consequently, $\mathcal{T}(X) \subseteq \mathcal{P}(X)$, and in this thesis, the term *function* includes partial functions.

Notation 2.7. In this thesis, function applications are written on the right. For example, xf means 'f applied to x', and fg means 'first f, then g'.

2.1 Universal Algebra

This section summarises the basic concepts in universal algebra which are needed in the later parts of this thesis. Universal algebra is a rich and deep subject in its own right, and a detailed exposition can be found in the classic text by Grätzer [12], or in the book by Denecke and Wismath [8].

Definition 2.8. An **n-ary operation** ρ on a set A is a total function which takes n elements of A and returns a single element of A. This number n is called the **arity** of ρ.

Different arities are given different names as follows.

- Nullary: zero argument (constant).
- Unary: one argument (*e.g.* the 'inverse' of a real number).
- Binary: two arguments (*e.g.* the 'addition' of numbers).
- Ternary: three arguments (*e.g.* `if-then-else`).

Notation 2.9. In contrast with functions *between* sets or algebras in Notation 2.7, operations are written in *prefix* or *infix* notation. For example, an n-ary operation ρ applied to $x_1, \ldots, x_n$ is written as $\rho(x_1, \ldots, x_n)$. If the operation ρ_2 is binary, then $\rho_2(x_1, x_2)$ is also written as $x_1\ \rho_2\ x_2$.

Definition 2.10. An **algebra** is a tuple $\langle A; \rho_1, \rho_2, \dots, \rho_k \rangle$, where A is called the base set (or the carrier set), and the ρ_i's are the operations. Each of the operations has its own arity n_i, and the tuple $(n_1, n_2, \dots, n_k)$ is called the **type** of the algebra.

Examples A group $\langle G; \cdot, x^{-1}, 1 \rangle$ is an algebra of type $(2, 1, 0)$, and a lattice $\langle L; \wedge, \vee \rangle$ is an algebra of type $(2, 2)$.

Subalgebras and Quotient Algebras

Definition 2.11. A non-empty subset $B \subseteq A$ is a **subalgebra** of A when it has the same type as A, and is closed under every operation, *i.e.* for every n-ary operation ρ:

$$\forall x_1, \dots, x_n \in B \ : \ \rho\big(x_1, \dots, x_n\big) \in B.$$

The beginning of this chapter introduced the concept that an equivalence relation partitions a set into equivalence classes; in an algebra A with an arbitrary equivalence relation θ, however, A/θ may not form an algebra because a valid operation may not be definable. For A/θ to be a well defined algebra, θ needs to be a congruence.

Definition 2.12. A **congruence** on an algebra A is an equivalence relation θ which satisfies the following. For every n-ary operation ρ, and for all $a_1, \dots, a_n$ and $b_1, \dots, b_n \in A$, if $a_1 \ \theta \ b_1 \ , \dots , \ a_n \ \theta \ b_n$, then

$$\rho(a_1, \dots, a_n) \ \theta \ \rho(b_1, \dots, b_m).$$

Definition 2.13. When θ is a congruence on the algebra A, the **quotient algebra** is the algebra which has the same type as A, has A/θ as its base set, and each of its operations ρ (assumed n-ary) is defined to be:

$$\forall \overline{x_1}, \dots, \overline{x_n} \in A/\theta \ : \ \rho(\overline{x_1}, \dots, \overline{x_n}) = \overline{\rho(x_1, \dots, x_n)}.$$

Homomorphisms and Embeddings

From now on, let A and B be two algebras with the same type.

Definition 2.14. A **homomorphism** is a function $\phi : A \to B$ which 'preserves every operation'. In other words, for every operation ρ and $x_1, \dots, x_n \in A$,

$$\big(\rho_A(x_1, \dots, x_n)\big)\phi = \rho_B(x_1\phi, \dots, x_n\phi).$$

Various special types of homomorphisms are given special names. In this thesis, a particularly important type is defined below.

Definition 2.15. An **embedding** is an injective homomorphism.

2.2 Semigroups

The algebraic study of total functions with function composition gave rise to the notion of the semigroup.

Definition 2.16. A **semigroup** is an algebra $\langle S\ ;\ \cdot\ \rangle$ with one binary operation which satisfies the associativity axiom: for all $x, y, z \in S$,

- $(xy)z = x(yz)$.

For example, both $\mathcal{P}(X)$ and $\mathcal{T}(X)$ are semigroups, with function composition as their binary operations. (See Notation 2.6.)

Modelling the identity function gives rise to the notion of a *monoid*.

Definition 2.17. A **monoid** is an algebra $\langle S\ ;\ \cdot\ , 1\rangle$ with one binary operation and an **identity** (nullary operation), which satisfy the following axioms. For all $x, y, z \in S$:

- $(xy)z = x(yz)$;
- $x1 = 1x = x$.

For example, both $\mathcal{P}(X)$ and $\mathcal{T}(X)$ are monoids, with function composition as before, and the full identity function as the identity.

Other than the identity, another very common nullary operation is the **zero**, denoted as 0, and it satisfies $\forall x \in S : x0 = 0x = 0$.

Finally, if every element of a monoid has an inverse, the result corresponds to the notion of the group. The elements of a group model bijections.

Definition 2.18. A **group** is an algebra $\langle S \ ; \ \cdot \ , \ \cdot^{-1}, 1 \rangle$, where $\cdot$ is a binary operation, $\cdot^{-1}$ is a unary operation called the *inverse*, and 1 is a nullary operation called the *identity*. The axioms of a group are the following. For all $x, y, z \in S$:

- $(xy)z = x(yz)$;
- $x1 = x = 1x$;
- $xx^{-1} = 1 = x^{-1}x$.

Functions are not the only motivation behind semigroups, and there is the notion of a semilattice.

Definition 2.19. A semigroup $\langle S \ ; \ \cdot \ \rangle$ is called a **semilattice** if it satisfies the following additional properties. For all $x, y \in S$:

- $xy = yx$;
- $xx = x$.

Semilattices are closely related to lattices, which are the subject of Section 2.4.

Earlier in this chapter, Section 2.1 introduced the concepts of sub-algebras and congruences. Here, the remainder of this present section will describe these concepts in the context of a semigroup S.

Definition 2.20. A **subsemigroup** H is a non-empty subset of S which is closed under the semigroup operation, *i.e.* for all $x, y \in H$, $(xy) \in H$. This ensures that H is itself a semigroup.

Rephrasing Definition 2.12 in the context of semigroups, a congruence on a S is an equivalence relation θ which satisfies

$$\forall a_1, a_2, b_1, b_2 \in S \ : \ a_1 \ \theta \ b_1 \ , \ a_2 \ \theta \ b_2 \implies a_1a_2 \ \theta \ b_1b_2.$$

Similarly for Definition 2.13, when θ is a congruence on S, the set of θ-classes forms a well-defined semigroup, with the operation defined to be

$$\forall \overline{x}, \overline{y} \in S/\theta \;:\; \overline{x} \cdot \overline{y} = \overline{xy},$$

and this semigroup S/θ is called the **quotient semigroup**.

2.3 Left Restriction Semigroups

There is also a special type of semigroup which has a unary operation representing the *domain*. In $\mathcal{P}(X)$, this domain operation D takes a function f to the restriction of the identity function to $dom(f)$, *i.e.*

$$D(f) = \{(x, x) \mid x \in dom(f)\}.$$

A semigroup with such a domain operation is called a *left restriction semigroup.*

Definition 2.21. A **left restriction semigroup** is an algebra $\langle S \,;\, \cdot\,, D\rangle$ where $\cdot$ satisfies the associativity law, and D satisfies the following axioms. For all $x, y \in S$.

(R1) $D(x)x = x$

(R2) $D(x)D(y) = D(y)D(x)$

(R3) $D\big(D(x)\big) = D(x)$

(R4) $D(xy) = D(x)D(xy)$

(R5) $xD(y) = D(xy)x$

The concept of left restriction semigroups has many different guises; these guises, in addition to the generalisation of the left restriction, are explored by Gould in [11].

The notation and results in this thesis originate from the paper *An Invitation to C-Semigroups* [16] by Jackson and Stokes, and this approach is also followed in these authors' other works [18, 19].

As the title of [16] suggests, the object studied there is the *C-semigroup*. It is a semigroup equipped with a unary operation C, which satisfies axioms similar but different to those of the left restriction semigroup. However, a left restriction semigroup $\langle S\ ; \cdot, D\rangle$ is in fact equivalent to a *twisted C-semigroup* $\langle S\ ; \times, C\rangle$ (see [16, Section 3]) by letting $a \cdot b = b \times a$ for all $a, b \in S$.

The next proposition contains some useful properties which can be further derived from these axioms. These properties have been previously established in [16, Proposition 1.2].

Proposition 2.22. *Let S be a left restriction semigroup, then the following are true for all $x, y \in S$.*

(R6) $D(x)D(y) = D\big(D(x)D(y)\big)$

(R7) $D(x)D(x) = D(x)$

(R8) $D(x)D(y) = D\big(D(x)y\big)$

(R9) $D(xy) = D\big(xD(y)\big)$

An important subset of a left restriction semigroup is the set of **domain elements**:

$$\begin{aligned} D(S) &= \{D(x) \mid x \in S\} \\ &= \{y \in S \mid D(y) = y\}. \end{aligned}$$

These two definitions are equivalent due to (R3).

Proposition 2.23. *For a semigroup S, the subset $D(S)$ forms a subsemigroup, which is a semilattice.*

This subsemigroup of domain elements has also been previously established in [16].

If the left restriction semigroup has 0 as one of its operations, then the following additional axiom is required:

(R0) $D(0) = 0$.

This axiom asserts that $0 \in D(S)$, which is true in the cases of $\mathcal{P}(X)$ because the empty function is the restriction of the identity function to the empty set.

On the other hand, if the left restriction semigroup contains the identity 1, then

$$\begin{aligned} 1 &= D(1) \cdot 1 && \text{(R1)} \\ &= D(1), && \text{(identity law)} \end{aligned}$$

so $1 \in D(S)$ follows directly.

The next two propositions are useful in a later part of this thesis.

Proposition 2.24. *Let $x, a, b \in S$ with $a, b \in D(S)$. Then, $D(xab) = D(xa) \cdot D(xb)$.*

Proof. Starting from the right-hand side:

$$\begin{aligned} D(xa) \cdot D(xb) &= D\big(D(xa) \cdot xb\big) \cdot D(xa) && \text{(R5)} \\ &= D\big(D(xa)x \cdot b\big) \cdot D(xa) && \\ &= D\big(xD(a) \cdot b\big) \cdot D(xa) && \text{(R5)} \\ &= D(xab) \cdot D(xa) && \text{(because } D(a) = a\text{)} \\ &= D(xa) \cdot D(xab) && \text{(R2)} \\ &= D(xab) && \text{(R4)} \end{aligned}$$

□

Proposition 2.25. *Let $x, y, a, b \in S$ same as above. Then,*

$$D(xay) \cdot D(xby) = D(xaby).$$

Proof. Starting from the left-hand side:

$$\begin{aligned}
D(xay) \cdot D(xby) &= D\big(D(xay) \cdot D(xby)\big) && \text{(R6)}\\
&= D\big(D(xay) \cdot xby\big) && \text{(R9)}\\
&= D\big(D(xay)x \cdot by\big) && \\
&= D\big(xD(ay) \cdot by\big) && \text{(R5)}\\
&= D\big(x \cdot D(ay)b \cdot y\big) && \\
&= D\big(x \cdot bD(ay) \cdot y\big) && \text{(R2)}\\
&= D\big(xb \cdot D(ay) \cdot y\big).
\end{aligned}$$

Now, note that

$$\begin{aligned}
D(ay) &= D\big(a \cdot D(y)\big) && \text{(R9)}\\
&= D\big(D(a) \cdot D(y)\big) && \text{(because } a = D(a)\text{)}\\
&= D(a) \cdot D(y) && \text{(R6)}\\
&= a \cdot D(y). && \text{(because } a = D(a)\text{)}
\end{aligned}$$

So, back to the main argument:

$$\begin{aligned}
D\big(xb \cdot D(ay) \cdot y\big) &= D\big(xb \cdot aD(y) \cdot y\big) && \\
&= D\big(xba \cdot D(y)y\big) && \\
&= D\big(xaby\big) && \text{(R2) on } ab \text{ and (R1)}
\end{aligned}$$

□

2.4 Lattices

This section outlines the theory of lattices, covering the basic definitions and standard results required in the later parts of this thesis. More detailed discussions on lattices can be found in the book by Davey and Priestley [7].

Definition 2.26. A **lattice** is an algebra $\langle L; \wedge, \vee \rangle$ with two binary operations, and satisfying the following axioms. For all $x, y, z \in L$:

(L1) $(x \wedge y) \wedge z = x \wedge (y \wedge z)$; (Associativity)

$(x \vee y) \vee z = x \vee (y \vee z)$;

(L2) $x \wedge y = y \wedge x$; (Commutativity)

$x \vee y = y \vee x$;

(L3) $x \wedge x = x$; (Idempotence)

$x \vee x = x$;

(L4) $x \wedge (x \vee y) = x$; (Absorption)

$x \vee (x \wedge y) = x$.

The operation $\wedge$ is called *meet*, and $\vee$ is called *join*.

Note that each of the above axioms contains two equations. One of the two equations is called the *dual* of the other. Given one equation, the dual can be obtained simply by replacing each $\wedge$ with $\vee$, or *vice versa*.

Note that $\langle L; \wedge \rangle$ and $\langle L; \vee \rangle$ are both semilattices. Hence, if $\langle S; \wedge \rangle$ and $\langle S; \vee \rangle$ are semilattices which satisfy $\forall x, y \in S : x \wedge (x \vee y) = x \vee (x \wedge y) = x$, then $\langle S; \wedge, \vee \rangle$ is a lattice.

Lattices are not only purely algebraic structures, but are also closely related to partially ordered sets which were introduced in the beginning of Chapter 2.

Given a poset $\langle L, \leq \rangle$, the *least upper bound* of $x, y \in L$ is denoted as $lub\{x, y\}$, and is the smallest element in L which satisfies $x \leq lub\{x, y\}$ and $y \leq lub\{x, y\}$. On the other hand, the *greatest lower bound*, $glb\{x, y\}$, is defined dually. If every pair of elements in a poset L has a least upper bound and a greatest lower bound, then this poset is a lattice by defining

$$x \wedge y = glb\{x, y\} \text{ and } x \vee y = lub\{x, y\}.$$

A routine check can easily verify that this definition of $\wedge$ and $\vee$ fulfills the axioms of a lattice.

Conversely, any lattice has an intrinsic partial order, as demonstrated by the following *connecting lemma* [7, Lemma 2.8, page 39].

Lemma 2.27. *Let L be a lattice and $x, y \in L$. Then the following are equivalent:*

(i) $x \leq y$;

(ii) $x \wedge y = x$;

(iii) $x \vee y = y$.

In a partially ordered set, there can be elements which are not comparable. However, when every two elements are comparable, the relation is called a total order.

Definition 2.28. A **total order** on a set L is a partial order $\leq$ which satisfies

$$\forall x, y \in L \ : \ (x, y) \in L \text{ or } (y, x) \in L.$$

A totally ordered set is also called a **chain**.

We often encounter lattices with additional properties, and below are some of these.

Definition 2.29. A lattice is **bounded** when it has a top element 1 and a bottom element 0, which satisfy the identity laws $x \wedge 1 = x$ and $x \vee 0 = x$ for all $x \in L$.

Using the absorption law (L4), the above identity laws are equivalent to the following alternative rules.

$$x \vee 1 = (x \wedge 1) \vee x = 1 \text{ and}$$

$$x \wedge 0 = (x \vee 0) \wedge x = 0.$$

Definition 2.30. A **distributive lattice** is a lattice L which satisfies one of the following distributivity laws. For all $x, y, z \in L$:

(DL1) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$;

(DL2) $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

In fact, these two distributivity laws are equivalent in the presence of the absorption law, as the following proposition shows [7, Lemma 4.3].

Proposition 2.31. *A lattice L satisfies (DL1) if and only if it satisfies (DL2).*

Two of the most commonly-met lattice-like structures are sets with intersection and union, and propositional logic with 'and' and 'or'. Other than join and meet, both of these have an additional operation of *complement*.

Definition 2.32. Let L be a bounded lattice. An element $b \in L$ is a **complement** of $a \in L$ when $a \wedge b = 0$ and $a \vee b = 1$.

This is the most general definition, where an element of a lattice may have no complement. The following is a stronger notion.

Definition 2.33. A **complemented lattice** is a bounded lattice in which every element has at least one complement.

Complements are unique in the presence of the distributivity laws.

Proposition 2.34. *In a distributive complemented lattice, complements are unique.*

Now, we can finally define the notion of Boolean algebras, which captures the properties of classical propositional logic and the algebra of sets.

Definition 2.35. A **Boolean algebra** is a complemented distributive lattice. In other words, it is an algebra $\langle B; \wedge, \vee, \neg, 0, 1\rangle$ where

- $\langle B; \wedge, \vee, 0, 1\rangle$ forms a distributive lattice, and
- $\neg$ satisfies $x \vee \neg x = 1$ and $x \wedge \neg x = 0$.

2.5 Filters and Ideals

Filters and ideals are special kinds of sublattices which are useful in the applications of lattice theory. Let $\langle L; \wedge, \vee \rangle$ be a lattice.

Definition 2.36. A **filter** $F \subseteq L$ is a non-empty subset which satisfies:

(F1) for all $a, b \in F$, $a \wedge b \in F$;

(F2) for all $a \in F$ and $x \in L$, $a \leq x$ implies $x \in F$.

'Ideal' is simply the dual concept of 'filter'.

Definition 2.37. An **ideal** $J \subseteq L$ is a non-empty subset which satisfies:

(I1) For all $a, b \in J$, $a \vee b \in F$;

(I2) For all $a \in J$ and $x \in L$, $x \leq a$ implies $x \in J$.

Since filters and ideals are dual concepts, it is sufficient to investigate one or the other, so the remainder of this section focuses on filters only. In any case, only filters are used in later parts of this thesis. There are several special and important types of filters, and these are introduced below.

Definition 2.38. For any specific element $a \in L$, the **principal filter generated by** a is defined to be

$$\uparrow a = \{x \in L \mid x \geq a\}.$$

This notion can be generalised from a single element to a subset of the lattice, although in the latter case, the resultant set may not be a filter.

Definition 2.39. For a subset $Q \subseteq L$, the **up-set generated by** Q is defined to be

$$\uparrow Q = \{x \in L \mid \exists q \in Q : x \geq q\}.$$

Here is a lemma for a later part of this thesis.

Lemma 2.40. *Let F be a filter of a lattice L. Given some $a \in L \setminus F$, define $G = \uparrow \{a \wedge c \mid c \in F\}$. Then, G is a filter which includes a and properly contains F.*

Proof. Firstly, we show that G is a filter. Let $x, y \in G$, *i.e.* $x \geq a \wedge d_x$ and $y \geq a \wedge d_y$ for some $d_x, d_y \in F$. Then,

$$\begin{aligned} x \wedge y &\geq (a \wedge d_x) \wedge (a \wedge d_y) \\ &= a \wedge (d_x \wedge d_y). \end{aligned}$$

As $(d_x \wedge d_y) \in F$, $x \wedge y$ indeed belongs to G.

In addition, let $x \in G$ and $z \geq x$. Then, as $x \geq a \wedge d_x$, transitivity of our partial order directly implies that $z \geq a \wedge d_x$, and this simply means that $y \in G$.

Finally, every $f \in F$ satisfies $f \geq a \wedge f$, so $f \in G$ and hence $F \subseteq G$. Moreover, $a \geq a \wedge g$ for any $g \in F$, so $a \in G$. Since $a \notin F$, it follows that F is a proper subset of G. □

Definition 2.41. A **maximal filter** F in L is a filter which satisfies the following: if G is a filter which strictly contains F, then $G = L$. A maximal filter is also called an **ultrafilter**.

Definition 2.42. A **prime filter** P in L is a filter which satisfies the following: for all $a, b \in L$, $a \vee b \in P$ implies $a \in P$ or $b \in P$.

In distributive lattices, the maximality and primeness properties of filters are closely related, and the following theorem explores this connection. (See [7, Theorem 10.11, page 233].)

Theorem 2.43. *Let L be a distributive lattice with bottom element 0. Then, every maximal filter in L is prime.*

Zorn's lemma is a vital tool for inferring the existence of maximal filters, and it often requires the following theorem, with which this section closes.

Theorem 2.44. *Let L be a lattice, and $\mathcal{C}$ a chain of filters in L, ordered by set inclusion. Then, the set*

$$U = \bigcup_{F \in \mathcal{C}} F$$

is still a filter of L.

Proof. Let $x, y \in U$. Then, there are some $F_1, F_2 \in \mathcal{C}$ such that $x \in F_1$ and $y \in F_2$. Since $\mathcal{C}$ is a chain, it can be assumed without loss of generality that $F_1 \subseteq F_2$, and hence $x, y \in F_2$. As F_2 is a filter, $x \wedge y$ is contained in F_2; therefore, $x \wedge y \in U$.

Let $x \in U$ and $z \in L$ with $z \geq x$. Then, there is an $F \in \mathcal{C}$ which contains x. Since F is a filter, z is contained in F and hence in U.

Therefore, U is indeed a filter. $\square$

2.6 Many-Sorted Algebras

The program algebra studied in this thesis is two-sorted, as it has a *test sort* K and a *program sort* S. Each of these two sorts has its own operations: K has the usual logical connectives, and S has functional composition. In addition, there are hybrid operations such as the ternary `if-then-else`, which has type $K \times S \times S \to S$.

Another familiar example of a two-sorted algebra is the vector space. One of its sorts is the field of scalars $\mathbb{F}$, which has the usual operations and axioms of a field. The other sort is the vectors V, whose operations include vector addition $V \times V \to V$. The scalar multiplication is a hybrid operation, which has type $\mathbb{F} \times V \to V$. All of these operations satisfy a certain set of axioms.

The two examples above belong to the theoretical framework of *many-sorted* (or *heterogeneous*) algebras, which is the generalisation of the single-sorted (or homogeneous) universal algebra briefly summarised in Section 2.1. Nevertheless, as Birkhoff and Lipson laid out in [3], the concepts of single-sorted algebras carry over into many-sorted ones in a natural way.

2.7 Axiomatisation

An abstract algebra is often a model of some concrete objects in the real world, and the deducible theorems in the abstract algebra should correspond to truths about the concrete objects. For instance:

- a group is a model of a set of permutations with composition and inversion,
- a left restriction semigroup is a model of a set of partial functions with composition and domain, and
- a Boolean algebra models a set of subsets of a set with union, intersection and complementation [7].

As an example, how are the axioms of a group determined? Firstly, a set of permutations satisfies the axioms, *i.e.* the set of permutations is itself a group. This step merely involves routine checking. Next, we show that every group is a permutation group by constructing an embedding from any group G to the group of permutations on some set. This is in fact the well-know Cayley's Theorem.

Theorem. *Every group G is isomorphic to a subgroup of $Sym(G)$, the symmetric group acting on G. (Alternatively, G is* embeddable *in $Sym(G)$.)*

The isomorphism map $\phi : G \to Sym(G)$ required to prove this theorem is defined by $\forall g \in G \ : \ g\phi = \psi_g$, where $\psi_g : G \to G$ and $\forall x \in G \ : \ x\psi_g = xg$. Then, it remains to be shown that:

- Each ψ_g is really a permutation;
- $\forall g, h \in G \ : \ g\phi \cdot h\phi = (gh)\phi$;
- 1ϕ is really the identity in $Sym(G)$;
- $\forall g \in G \ : \ (g^{-1})\phi = \big(g\phi\big)^{-1}$;
- ϕ is injective.

This definition of ϕ mapping g to 'right-multiplication by g' is called the Cayley-style representation. This same representation also works for semigroups with identity, and it embeds every semigroup S into $\mathcal{P}(S)$.

In the more general setting of relational algebras, the steps required to axiomatise are outlined by Schein [31, page 53]:

> *'At first, one must "guess" the right axioms (the guess is not quite arbitrary, and a good method simplifies the task very considerably). Secondly, one must verify the necessity of the axioms. This step is the easiest: one checks the axioms on proper relation algebras. The most difficult step is the third one: the proof of sufficiency. There is given an algebraic system satisfying the axioms, and one must construct a proper relation algebra of the given class isomorphic to the given system.'*

Most generally, how do we determine the set of axioms of a particular algebra? The axioms are what 'characterise' the abstract algebra, so they should enable a direct correspondence between the abstract and the concrete algebras. This involves two directions:

1. the concrete algebra satisfies the axioms of the abstract algebra,
   i.e. concrete $\implies$ abstract;

2. the abstract algebra is isomorphic to a concrete algebra,
   i.e. abstract $\implies$ concrete.

The first direction is sometimes called the *soundness* of the axioms, and the second the *completeness*. Moreover, in the second point, the isomorphism from the abstract to the concrete is called a **representation**.

Chapter 3

Operations

This chapter will introduce the algebraic structures and operations which will model the programs and the tests. At the end, it will give a preview of all the axioms which will be studied in the later chapters.

3.1 Programs as Functions

Every program takes a start state from the state space, and arrives at an end state if the program is well defined on this input. In this thesis, programs are assumed to be deterministic, so the end state is unique if it exists. This suggests that programs can be viewed as functions.

Given certain inputs, a program may continue running indefinitely without halting. When a program is viewed as a function, this situation will be considered as the function being undefined at these inputs. Since a program may not be defined for every possible input, programs will be viewed specifically as partial functions.

Another possible bad situation is when a program terminates without completing a valid computation, *i.e.* the program 'crashes' or 'aborts'. This thesis will not consider the situation of 'abort', and will focus only on non-halting programs.

3.2 Operations Purely on Programs

An important operation on programs is *concatenation*, which executes one program straight after another. Since programs are viewed as functions, this operation is simply the usual function composition, which is associative by nature. Hence, programs with composition form a semigroup.

In addition to composition, there are two nullary operations which represent the special programs of `skip` and `loop`. The `skip` program is the one that always outputs the same value as its input. It makes no effect when composed with other programs, so it will be the *identity* element which satisfies $1 \cdot p = p \cdot 1 = p$ for all programs p. On the other hand, the `loop` program is the one that never halts. A possible form of such a program is

```
while (true) { }.
```

When `loop` is composed with other programs, no matter the order, the resulting program is still always non-halting, so `loop` behaves like the *zero* constant 0, which satisfies $0 \cdot p = p \cdot 0 = 0$ for all programs p.

3.3 Tests

Now, logical tests will be brought into the picture.

A halting test is a predicate on the state space X, and a (total) predicate is a total function from X to the truth set $\{T, F\}$. Extending this idea, a possibly non-halting test is a partial predicate, which is a partial function from X to $\{T, F\}$, and the states at which a partial predicate is undefined represent where the evaluation of a test has not halted. Partial predicates possess all the usual operations which total predicates have, namely the connectives ‘and’ $\wedge$, ‘or’ $\vee$ and ‘not’ $\neg$, as well as the constants ‘true’ T and ‘false’ F.

When the tests have the possibility of not halting, the results of binary connectives $\wedge$ and $\vee$ become dependent on how these connectives are evaluated: whether sequentially or in parallel. The differences between these two evaluation strategies were introduced in

Chapter 1, and will be explained in more detail in Chapter 7. Meanwhile, the following tables show how the connectives are defined under parallel evaluation, which this thesis employs. Note that U denotes the situation when a test 'has not halted'.

$\wedge$	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

$\vee$	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

$\neg$	
T	F
F	T
U	U

A convenient means to specify partial predicates is the notion of *disjoint pairs.* A partial predicate α has a 'true' part $\alpha_T = \{x \in X \mid x\alpha = T\}$, as well as a 'false' part $\alpha_F = \{x \in X \mid x\alpha = F\}$. Both of these are subsets of X, and $\alpha_T \cap \alpha_F = \emptyset$ because nothing can be both true and false at once.

Definition 3.1. The set of **disjoint pairs** over a set X is

$$\{\langle A, B\rangle \mid A, B \subseteq X \; ; \; A \cap B = \emptyset\},$$

and the operations on these are defined as follows.

- $\langle A, B\rangle \wedge \langle C, D\rangle = \langle A \cap C, B \cup D\rangle$
- $\langle A, B\rangle \vee \langle C, D\rangle = \langle A \cup C, B \cap D\rangle$
- $\neg\langle A, B\rangle = \langle B, A\rangle$
- $T = \langle X, \emptyset\rangle$ and $F = \langle \emptyset, X\rangle$

Furthermore, two disjoint pairs $\langle A, B\rangle$ and $\langle C, D\rangle$ are equal if and only if both $A = C$ and $B = D$.

Since the algebra of subsets of a set correspond directly to a Boolean algebra, a set of disjoint pairs can also be defined on a Boolean algebra.

There is in fact an isomorphism between partial predicates and disjoint pairs, stated as the next proposition.

Proposition 3.2. *Let Γ be a mapping from the set of partial predicates on X to the set of disjoint pairs, and define Γ by $\alpha\Gamma = \big\langle A_T, A_F \big\rangle$, where*

$$A_T = \{x \in X \mid x\alpha = T\} \text{ and}$$
$$A_F = \{x \in X \mid x\alpha = F\}.$$

Then, this mapping Γ is an isomorphism.

This can be verified by very routine checking, so the proof is omitted.

3.4 The `if-then-else` Operation

The first hybrid operation which involves both tests and programs is `if-then-else`, which was introduced in the very start of Chapter 1. Formally, if S denotes the programs and K denotes the tests, then the `if-then-else` is a ternary operation

$$\cdot\,[\,\cdot\,,\,\cdot\,]\;:\;K \times S \times S \to S.$$

When the test halts and produces either T or F, the behaviour of `if-then-else` is

$$T[p,q] = p, \text{ and}$$
$$F[p,q] = q.$$

If the test does not halt, the `if-then-else` statement would be trying to evaluate the test indefinitely, so the whole statement behaves like `loop`. Hence, $U[p,q] = 0$.

3.5 The `Halt` Test and the Domain Operation

Since a program may be undefined at certain inputs, it is natural to ask 'whether a program terminates'. This is the `halt` test, denoted H, and it takes a program p as input and returns a test $H(p)$ as output. When p halts, the result $H(p)$ will be T; however, when p has not halted, there is no way to tell whether or not p will halt at a later time, so $H(p)$ will continue running indefinitely. This is one instance of how a non-halting test can arise.

Note that this `halt` test can never return false. Asking whether $H(p)$ is false is the same as asking whether p is non-halting. This is exactly the halting problem, which is well known not to be computable. Since this thesis models computable programs, such non-computable statements must not be expressible.

In the language of functions and partial predicates, for a function f on the set X, the `halt` test is defined as follows for all $x \in X$:

$$x\big(H(f)\big) = \begin{cases} T & \text{if } xf \text{ is defined;} \\ F & \text{nowhere;} \\ U & \text{if } xf \text{ is undefined.} \end{cases}$$

The `halt` test $H(p)$ gives rise to the *domain* operation on programs via the expression $\big(H(p)\big)[1,0]$, which represents 'the restriction of the identity function to where p is defined'. The domain operation $D(p)$ is thus defined by $\big(H(p)\big)[1,0]$. Conversely, H can be implicitly expressed in terms of D: $H(p)$ is exactly the test α which satisfies $\alpha[1,0] = D(p)$ and $\alpha[0,1] = 0$. Therefore, in order to study H, investigating D is necessary.

Now, given two programs p and q, both $H(p) \wedge H(q)$ and $H(p) \vee H(q)$ are also tests, and $\big(H(p) \wedge H(q)\big)[1,0]$ and $\big(H(p) \vee H(q)\big)[1,0]$ should represent the restriction of the identity function to 'where both p and q are defined' and 'where either p or q is defined' respectively. This brings the need of lattice operations *join* and *meet* on the set of identity restrictions. In the previous chapter, Proposition 2.23 showed that *meet* on identity restrictions is exactly functional composition. On the other hand, however, *join* needs to be studied as an additional operation. Hence, the algebra of programs will have an extra operation $\vee$ called the *domain join.*

3.6 Testing for Equality

Very often, the test used in an `if-then-else` statement involves testing the equality of two values, both of which can be results of other programs. This brings about the equality test, which takes two programs p and q and results in a test $(p = q)$. The equality test determines whether two programs produce the same output, given the same input. If both programs halt, we can proceed to test the equality between the two outputs. However, if one of the programs does not halt, then the equality test would not halt either.

In terms of functions and partial predicates, the equality test of functions f and g on the set X is defined as follows for all $x \in X$:

$$x(f = g) = \begin{cases} T & \text{if both } xf \text{ and } xg \text{ are defined, and } xf = xg; \\ F & \text{if both } xf \text{ and } xg \text{ are defined, but } xf \neq xg; \\ U & \text{if either } xf \text{ or } xg \text{ is undefined.} \end{cases}$$

3.7 Summary

To summarise, the programs are modelled by an algebra $\langle S\ ;\ \cdot\ , \vee, D, 1, 0 \rangle$, and this algebra satisfies:

- $\langle S\ ;\ \cdot\ , 1 \rangle$ is a monoid (Definition 2.17);
- $\langle S\ ;\ \cdot\ , D \rangle$ is a left restriction semigroup (Definition 2.21);
- 0 is the zero element (after Definition 2.17).

Moreover, $\langle S\ ;\ \cdot\ , \vee, D \rangle$ satisfies the following. For all $a, b, c \in D(S)$ and $x, y \in S$:

(D1) $(a \vee b) \vee c = a \vee (b \vee c)$

(D2) $a \vee b = b \vee a$

(D3) $a \vee a = a$

(D4) $a \vee (ab) = a \cdot (a \vee b) = a$

(D5) $a \vee (bc) = (a \vee b) \cdot (a \vee c)$

(D6) $D\big(x(a \vee b)\big) = D(xa) \vee D(xb)$

(D7) $ax = ay$ and $bx = by$ imply $(a \vee b)x = (a \vee b)y$

Then, let $\langle K; \wedge, \vee, \neg, T, F \rangle$ be the algebra of tests, and $\cdot\,[\,\cdot\,,\,\cdot\,] : K \times S \times S \to S$ the `if-then-else` operation. Together, the two sorted algebra which consists of S, K and the `if-then-else` operation is called the *program algebra* in the remainder of this thesis, and it satisfies all the axioms of S above, plus the following.

(T1) $(\alpha \vee \beta)[1,0] = \alpha[1,0] \vee \beta[1,0]$

$(\alpha \wedge \beta)[1,0] = \alpha[1,0] \cdot \beta[1,0]$

(T2) $(\alpha \vee \beta)[0,1] = \alpha[0,1] \cdot \beta[0,1]$

$(\alpha \wedge \beta)[0,1] = \alpha[0,1] \vee \beta[0,1]$

(T3) $(\neg\alpha)[s,t] = \alpha[t,s]$ for all $s, t \in S$

(T4) $\alpha[1,0] \cdot \alpha[0,1] = 0$

(T5) $\alpha[1,0], \alpha[0,1] \in D(S)$

(T6) $\alpha[1,0] = \beta[1,0]$ and $\alpha[0,1] = \beta[0,1]$ implies that $\alpha = \beta$

(G1) $\alpha[1,0] \cdot s = \alpha[1,0] \cdot \alpha[s,t]$

$\alpha[0,1] \cdot t = \alpha[0,1] \cdot \alpha[s,t]$

(G2) $(\alpha[1,0] \vee \alpha[0,1]) \cdot \alpha[s,t] = \alpha[s,t]$

These axioms will be proven sound and complete in the subsequent two chapters. Chapter 4 will deal with axioms (D1) to (D7), and Chapter 5 will study (T1) to (T6) and (G1) to (G2).

The equality test will be treated in isolation later in Chapter 6.

Chapter 4

Programs

This chapter will study the algebra of programs S, which is a left restriction semigroup with domain join, 1 and 0. It will establish the soundness and completeness of the axioms (D1) to (D7) in Section 3.7 by constructing a representation from this algebra of programs to an algebra of functions.

4.1 Domain Join

Earlier, Section 3.5 explained the need for an additional operation $\vee$, called the *domain join*. Let $\vee : D(S) \times D(S) \to D(S)$ be a binary operation on $D(S)$, which satisfies the following axioms. For all $a, b, c \in D(S)$ and $x, y \in S$:

(D1) $(a \vee b) \vee c = a \vee (b \vee c)$;

(D2) $a \vee b = b \vee a$;

(D3) $a \vee a = a$;

(D4) $a \vee (ab) = a \cdot (a \vee b) = a$;

(D5) $a \vee (bc) = (a \vee b) \cdot (a \vee c)$;

(D6) $D\big(x(a \vee b)\big) = D(xa) \vee D(xb)$;

(D7) $ax = ay$ and $bx = by$ imply $(a \vee b)x = (a \vee b)y$.

This partial operation on $D(S)$ can then be extended into a total operation on the whole of S, by setting $x \vee y$ to be $D(x) \vee D(y)$ for any two $x, y \in S$.

Identity restrictions, being functions, can be viewed as sets of maplets. However, compared to arbitrary functions, the maplets of an identity restriction have the special form of (x, x). Hence, every identity restriction f can be mapped to the subset $dom(f)$, and every subset A can be mapped to the identity restriction $\{(x, x) \mid x \in A\}$. This is a direct correspondence between identity restrictions and subsets, so the existing facts about sets establish the soundness of the first five axioms.

The axiom (D6) is a stronger version of distributivity in $D(S)$, and is required in a later proof. Let α and β be identity restrictions, and f an arbitrary partial function, all of which operate on some set X. Since both sides of the fourth axiom are identity restrictions, it is enough to show that the domains of both sides are equal, *i.e.*

$$dom\big(f \cdot (\alpha \vee \beta)\big) = dom\big(f\alpha\big) \cup dom\big(f\beta\big).$$

Given two partial functions g and h, $x \in dom(gh)$ if and only if $x \in dom(g)$ and $xg \in dom(h)$. Using this fact,

$$\begin{aligned}
x \in dom\big(f \cdot (\alpha \vee \beta)\big) &\iff x \in dom\big(f\big) \text{ and } xf \in dom\big(\alpha \vee \beta\big) = dom\big(\alpha\big) \cup dom\big(\beta\big) \\
&\iff \Big(x \in dom\big(f\big) \text{ and } xf \in dom\big(\alpha\big)\Big) \\
&\quad\ \text{or } \Big(x \in dom\big(f\big) \text{ and } xf \in dom\big(\beta\big)\Big) \\
&\iff x \in dom\big(f\alpha\big) \text{ or } x \in dom\big(f\beta\big) \\
&\iff x \in dom\big(f\alpha\big) \cup dom\big(f\beta\big),
\end{aligned}$$

and this justifies the soundness of (D6).

In fact, (D6) implies the following simpler version of distributivity:

(D6') $\forall a, b, c \in D(S) \;:\; a(b \vee c) = ab \vee ac,$

and using (D1) to (D5), together with (D6'), it follows by definition that $D(S)$ forms a distributive lattice.

Moreover, by taking the constants 1 and 0 into account, $\langle D(S) \;;\; \cdot\, , \vee, 1, 0 \rangle$ forms a bounded distributive lattice.

Finally, for (D7), let α and β be identity restrictions, and f and g arbitrary partial functions on X. Suppose $(x, y) \in (\alpha \vee \beta)f$; then, $x \in \mathit{dom}(\alpha) \cup \mathit{dom}(\beta)$ and $xf = t$. Without loss of generality, assume that $x \in \mathit{dom}(\alpha)$; then $(x, y) \in \alpha f$. Now, by one of the premises $\alpha f = \alpha g$, it must be the case that $(x, y) \in \alpha g$ too, and then it follows that $(x, y) \in (\alpha \vee \beta)g$.

Using the same argument for the other direction, the soundness of (D7) can be established.

4.2 Determinative Pair in General

The goal of this chapter is to construct a representation of the algebra of programs, which at its core is a semigroup. Earlier, Section 2.7 introduced the Cayley-style representation, which works for groups and semigroups with 1. However, this representation technique does not always work when the semigroup (with 1) has additional operations, such as D and $\vee$ in this thesis, or other operations which arise from different studies of function semigroups. To deal with this obstacle, a new representation technique of *determinative pairs* was developed by Schein in [32].

This section introduces the principle of determinative pairs, and demonstrates that general determinative pairs preserves the semigroup operation, 1 and 0. The remaining operations D and $\vee$ will be treated in the next section, which studies a specific determinative pair.

First, below are some basic notions on a semigroup S.

Definition 4.1. A relation $\sim$ on a semigroup S is **right-regular** when it satisfies the following:

$$a \sim b \iff \forall x \in S : ax = bx.$$

A right-regular equivalence relation is called a **right congruence**, as in [15, p.22].

Definition 4.2. A **right ideal** is a subsemigroup $J \subseteq S$ which satisfies the following:

$$\forall a \in J, x \in S : ax \in J.$$

Then, a determinative pair is defined in terms of these basic notions.

Definition 4.3. Let S be a semigroup. A **determinative pair** is of the form $\langle \epsilon, W \rangle$, where

- ϵ is a right congruence on S, and
- $W \subset S$ is a union of ϵ-classes, and is also a right ideal.

Now, a general determinative pair induces a semigroup homomorphism from an arbitrary semigroup S to a semigroup of partial functions on a quotient semigroup of S.

Theorem 4.4. *Let S be a semigroup, and $\langle \epsilon, W \rangle$ a determinative pair. Then, define the base set X to be $\big(S \setminus W\big)/\epsilon$. For every $x \in S$, define $\psi_x \in \mathcal{P}(X)$ by*

$$\forall \overline{s} \in X \ \ : \ \overline{s}\psi_x = \begin{cases} \overline{sx} \textit{ if } sx \notin W; \\ \textit{undefined otherwise.} \end{cases}$$

Then, the map $\phi : S \to \mathcal{P}(X)$ which maps x to ψ_x is a semigroup homomorphism.

Proof. First, given $x \in S$, we show that ψ_x is a well defined function. Consider $s, t \in S$ such that $\overline{s} = \overline{t}$. Hence, $s \ \epsilon \ t$, and by the right-regular property, $sx \sim tx$.

$$\overline{s}\psi_x = \overline{sx} = \overline{tx} = \overline{t}\psi_x.$$

Hence, the function ψ_x is indeed well defined.

Next, we show that f is a homomorphism, *i.e.* $\forall x, y \in S : (xy)\phi = x\phi \cdot y\phi$. By the definition of f, this is equivalent to showing $\psi_{xy} = \psi_x\psi_y$.

Suppose $\psi_x\psi_y$ is undefined at $\overline{s} \in X$, and this means either that $\overline{s}$ is undefined for ψ_x, or that $\overline{s}\psi_x = \overline{sx}$ is undefined for ψ_y. Thus, either $sx \in W$ or $(sx)y \in W$. In the first case, since W is a right ideal, $sx \in W$ implies that $(sx)y \in W$. Then, given $sxy \in W$, we can conclude that $\overline{s}$ is undefined for ψ_{xy}.

Now we have shown that the domains of ψ_{xy} and $\psi_x\psi_y$ are equal, we can consider $\overline{s} \in X$ where $\overline{s}$ is defined for both sides. Then,

$$\begin{aligned}\overline{s}\psi_{xy} &= \overline{s(xy)} \\ &= \overline{(sx)y} \\ &= (\overline{sx})\psi_y \\ &= (\overline{s}\psi_x)\psi_y \\ &= \overline{s}\psi_x\psi_y,\end{aligned}$$

so indeed $(xy)\phi = x\phi \cdot y\phi$, and this completes the proof. □

In fact, the map ϕ in the previous theorem preserves the constants 1 and 0 too.

Corollary 4.5. *If 1 is the identity in S, then $1\phi = \psi_1$ is the total identity function on X.*

Proof. Note that a right ideal such as W never contains 1: if $1 \in W$, we can let $x \in (S \setminus W)$. As $1 \cdot x = x \notin W$, the condition of a right ideal is violated, so it must be the case that $1 \notin W$.

Using this fact, for any $\overline{s} \in X$ (*i.e.* $s \notin W$), $s \cdot 1 = s \notin W$, so ψ_1 is defined at every $\overline{s} \in X$. Then, $\overline{s}\psi_1 = \overline{s1} = \overline{s}$, so ψ_1 is indeed the total identity function. □

Corollary 4.6. *If 0 is the zero in S, then $0\phi = \psi_0$ is the empty function on X.*

Proof. Note that a right ideal such as W always contains 0: if $0 \notin W$, then let $a \in W$ and $0 \in S$. As $a \cdot 0 = 0 \notin W$, the condition of a right ideal is violated, so it must be the case that $0 \in W$.

Using this fact, for any $\overline{s} \in X$, $s \cdot 0 = 0 \in W$, so ψ_0 is undefined at any $\overline{s} \in X$. Therefore, ψ_0 is indeed the empty function. □

So far, a general determinative pair provides a homomorphism which preserves composition, 1 and 0, but not $\vee$ and D. Also, the homomorphism induced by a general determinative pair is not necessarily injective. The operations $\vee$ and D, as well as injectivity, depend on which specific determinative pair is used, and this is the subject of the remainder of this chapter.

4.3 Determinative Pair in Context

After the general principle of determinative pairs was introduced, this section applies it to the current context by specify a suitable right congruence and right ideal. It turns out that the determinative pair used by Jackson and Stokes in [19] works well in the context of this thesis, so the definitions and results in this section and the next are adopted from [19] and included for completeness.

Definition 4.7. Let F be a proper filter of $D(S)$. Define a binary relation θ_F on S by

$$x \; \theta_F \; y \iff \exists \mathtt{e} \in F : \mathtt{e}x = \mathtt{e}y.$$

In addition, define $W_F = \{a \in S \mid D(a) \notin F\}$.

Notation 4.8. For the purpose of clarity, the using of symbols will follow the convention below. Given a left restriction semigroup S and a filter $F \subseteq D(S)$,

- x, y, z and s, t denote general elements of S,
- a, b, c denote elements of $D(S)$, and
- $\mathtt{e}, \mathtt{f}, \mathtt{g}$ denote elements of F.

Lemma 4.9. *$\langle \theta_F, W_F \rangle$ forms a determinative pair.*

Proof. The following statements need to be established:

(i) θ_F is an equivalence relation.

(ii) θ_F is right-regular.

(iii) W_F is a union of equivalence classes in S/θ_F.

(iv) W_F is a right ideal.

And these will be proven one by one.

(i) Since every $\mathtt{f} \in F$ and every $x \in S$ satisfy $\mathtt{f}x = \mathtt{f}x$, it is true that $x \; \theta_F \; x$, so θ_F is indeed reflexive. Also, the symmetry of equality directly implies the symmetry of θ_F.

Finally, for transitivity, suppose that $x\theta_F y$ and $y\theta_F z$. This means that there exist some $\mathtt{f}, \mathtt{g} \in F$ such that $\mathtt{f}x = \mathtt{f}y$ and $\mathtt{g}y = \mathtt{g}z$. Now, as the semigroup operation is associative, and is commutative in $F \subset D(S)$,

$$\begin{aligned}
(\mathtt{fg})x &= \mathtt{g}(\mathtt{f}x) \\
&= \mathtt{g}(\mathtt{f}y) \\
&= \mathtt{f}(\mathtt{g}y) \\
&= \mathtt{f}(\mathtt{g}z) \\
&= (\mathtt{fg})z,
\end{aligned}$$

But F being a filter means that $(\mathtt{fg}) \in F$, so $x\theta_F z$.

(ii) Let $x, y \in S$ such that $x\theta_F y$, *i.e.* $\mathtt{f}x = \mathtt{f}y$ for some $\mathtt{f} \in F$. For any $s \in S$, $(\mathtt{f}x)s = (\mathtt{f}y)s$, and by associativity, $\mathtt{f}(xs) = \mathtt{f}(ys)$, so $xs\theta_F ys$.

(iii) This statement is the same as the following: Given $x, y \in S$, if $x \notin W_F$ and $x\theta_F y$, then $y \notin W_F$.

Suppose that $x \in W_F$ and hence $D(x) \notin F$. Assume also that $\mathsf{e}x = \mathsf{e}y$ for some $\mathsf{e} \in F$.

$$\begin{aligned}
D(y) &\geq \mathsf{e}D(y) && \text{the property of meet}\\
&= D(\mathsf{e})D(y) && \text{because } \mathsf{e} \in D(S)\\
&= D(\mathsf{e}y) && \text{(R4)}\\
&= D(\mathsf{e}x) && \text{by assumption}\\
&= D(\mathsf{e})D(x) && \text{(R4)}\\
&= \mathsf{e}D(x) && \text{because } \mathsf{e} \in D(S)
\end{aligned}$$

Both e and $D(x)$ belong to the filter F, so $\mathsf{e}D(x) \in F$. Hence $D(y) \in F$ too, and this is equivalent to $y \notin W_F$.

(iv) Let $a \in W_F$, and this means that $D(a) \notin F$. Now, take any $x \in S$ and consider $D(ax)$. By Axiom (R4), $D(ax) = D(a)D(ax)$, so $D(ax) \leq D(a)$.

If $D(ax) \in F$, this inequality would imply that $D(a) \in F$ because F is a filter. However, $D(a) \notin F$, so it must be the case that $D(ax) \notin F$. Therefore, $ax \in W_F$, which means that W_F is indeed a right ideal. □

This section shows that a proper filter of $D(S)$ gives rise to a particular determinative pair, without much detail on the filter itself. The next section is devoted to the specifics of how such a filter is chosen.

4.4 Separating Congruences and Filters

Earlier, Theorem 4.4 established a homomorphism from a semigroup (with 1 and 0) into $\mathcal{P}(X)$ for some set X. However, this homomorphism may not be injective. Hence, this section provides a remedy by introducing the notion of the (s,t)-separating property, which turns out to be crucial to achieving injectivity.

Definition 4.10. Let A be an algebra, and $s, t \in A$ with $s \neq t$. Then, an (s,t)-separating equivalence relation σ is one that satisfies $(s,t) \notin \sigma$.

In Definition 4.7, the right congruence is specified by a filter. If an (s,t)-separating right congruence is to be specified using a filter in the same way, then what properties must this filter have? This leads to the definition of an (s,t)-separating filter.

Definition 4.11. Let S be a left restriction semigroup, and $s,t \in S$ with $s \neq t$. Then, a filter F in $D(S)$ is (s,t)-separating when

- $D(s) \in F$, and
- $\neg \exists \mathsf{e} \in F : \mathsf{e}s = \mathsf{e}t$.

Then, it immediately follows that given an (s,t)-separating filter F, the corresponding θ_F is an (s,t)-separating right congruence.

For conciseness, an (s,t)-separating congruence or filter will be simply referred to as a separating congruence or filter, when no ambiguity can arise.

After defining the correct concept of separating filters, we must also show that these filters do exist. This is the next lemma, which is adopted from the passage preceding [19, Lemma 2.4].

Lemma 4.12. *Let S be a left restriction semigroup, and $s,t \in S$ with $s \neq t$. Then there exists an (s,t)-separating filter in $D(S)$.*

Proof. Since $s \neq t$, we can let $s \nleq t$ without loss of generality. Consider the principal filter generated by $D(s)$:

$$\uparrow D(s) = \{\mathsf{e} \in D(S) \mid D(s) \leq \mathsf{e}\},$$

where the order is induced by viewing multiplication as meet, *i.e.* $x \leq y \iff xy = x$.

Is this filter (s,t)-separating? Suppose for contradiction that it is not, which means that there exists an $\mathsf{e} \in {\uparrow}D(s)$ such that $\mathsf{e}s = \mathsf{e}t$. Then,

$$\begin{aligned}
s &= D(s) \cdot s && \text{(R4)} \\
&= \big(D(s)\mathsf{e}\big) \cdot s && \mathsf{e} \in {\uparrow}D(s) \implies \mathsf{e} \geq D(s) \\
&= D(s) \cdot (\mathsf{e}s) \\
&= D(sa) \cdot (\mathsf{e}t) \\
&= \big(D(s)\mathsf{e}\big) \cdot t \\
&= D(s) \cdot t,
\end{aligned}$$

and this means that $s \leq t$, which contradicts our assumption of $s \nleq t$. Hence, $\uparrow D(s)$ is indeed an (s,t)-separating filter, and this finishes the proof. $\square$

In addition, the *maximality* of these separating filters turns out to be a crucial property that enables the representation to work.

Definition 4.13. A *maximally (s,t)-separating filter* is an (s,t)-separating filter which is not properly contained in any other (s,t)-separating filter.

Note that a maximally separating filter is not necessarily a maximal filter. A maximally separating filter may well be contained in another proper filter, with this other proper filter not separating.

Building on Lemma 4.12 and using Zorn's lemma, we can infer the existence of a maximally separating filter. Like Lemma 4.12, the next lemma is also adopted from the passage preceding [19, Lemma 2.4].

Lemma 4.14. *Let S be a left restriction semigroup, and $s, t \in S$ with $s \neq t$. Then there exists a* maximally *(s,t)-separating filter in $D(S)$.*

Proof. Using the fixed s and t mentioned in the statement, consider the set $\mathcal{F}$ of all (s,t)-separating filters in $D(S)$, ordered by set inclusion. Zorn's lemma will be employed, and it requires that

- this set is non-empty, and
- every chain has an upper bound in $\mathcal{F}$.

The first statement is already covered by Lemma 4.12.

Secondly, let $\mathcal{C}$ be a chain in $\mathcal{F}$. Consider the union of all filters in $\mathcal{C}$,

$$U = \bigcup_{F \in \mathcal{C}} F.$$

By Theorem 2.44, the set U is certainly a filter. Since all of the filters in $\mathcal{C}$ are (s,t)-separating, by definition, none of them contains $D(s)$. Also, none of them contains any element $\mathtt{e}$ such that $\mathtt{e}s = \mathtt{e}t$, so their union U cannot contain such an element. Therefore, U is (s,t)-separating.

Moreover, every $F \in \mathcal{C}$ is a subset of U, so U is indeed an upper bound of the chain $\mathcal{C}$.

Therefore, as both criteria (i) and (ii) are satisfied, Zorn's lemma implies that the set $\mathcal{F}$ has a maximal element, *i.e.* a maximally (s,t)-separating filter does exist. □

This section culminates in the following lemma, which demonstrates the importance of the maximality of separating filters: maximally separating filters are prime. (See Definition 2.42.)

Lemma 4.15. *Let S be a left restriction semigroup, $s, t \in S$ with $s \nleq t$, and $F \subset D(S)$ a maximally (s,t)-separating filter. Then, F is prime.*

Proof. Let $a, b \in D(S)$, and assume for contradiction that $a \vee b \in F$ with $a \notin F$ and $b \notin F$. Since $a \notin F$, we can consider $G_a = \uparrow\{a\mathtt{f} \mid \mathtt{f} \in F\}$. According to Lemma 2.40,

G_a is a filter which properly contains a and F. Hence, G_a cannot be a maximally (s,t)-separating filter because F already is one, and $F \subset G_a$. This means that there exists some $\mathtt{g}_a \in G_a$ such that $\mathtt{g}_a s = \mathtt{g}_a t$. Moreover, $\mathtt{g}_a \geq a\mathtt{f}_a$ for some $\mathtt{f}_a \in F$, so $a\mathtt{f}_a = a\mathtt{f}_a\mathtt{g}_a$, and

$$\begin{aligned}(a\mathtt{f}_a)s &= (a\mathtt{f}_a\mathtt{g}_a)s \\ &= a\mathtt{f}_a(\mathtt{g}_a s) \\ &= a\mathtt{f}_a(\mathtt{g}_a t) \\ &= (a\mathtt{f}_a\mathtt{g}_a)t \\ &= (a\mathtt{f}_a)t.\end{aligned}$$

Similarly for b, we can obtain $(b\mathtt{f}_b)s = (b\mathtt{f}_b)t$ for some $\mathtt{f}_b \in F$.

Then, multiplying both sides of $(a\mathtt{f}_a)s = (a\mathtt{f}_a)t$ by $\mathtt{f}_b$, and dually for b, we get

$$\begin{aligned}(a\mathtt{f}_a\mathtt{f}_b)s &= (a\mathtt{f}_a\mathtt{f}_b)t \\ (b\mathtt{f}_a\mathtt{f}_b)s &= (b\mathtt{f}_a\mathtt{f}_b)t.\end{aligned}$$

Now, consider $a\mathtt{f}_a\mathtt{f}_b \vee b\mathtt{f}_a\mathtt{f}_b$. By distributivity of (D6'), this equals to $(a \vee b)\mathtt{f}_a\mathtt{f}_b$, which is in F because all of $(a \vee b)$, $\mathtt{f}_a$ and $\mathtt{f}_b$ are in F.

However, (D7) implies that

$$\big((a \vee b)\mathtt{f}_a\mathtt{f}_b\big)s = \big((a \vee b)\mathtt{f}_a\mathtt{f}_b\big)t,$$

and this contradicts our assumption of F being (s,t)-separating.

Therefore, given $a \vee b \in F$, at least one of a and b must be in F, so F is indeed a prime filter. $\square$

Finally, Lemmas 4.14 and 4.15 combine to give the following concise statement.

Corollary 4.16. *Given $s, t \in S$ with $s \not\leq t$, there exists a prime filter F of $D(S)$ which is maximally (s,t)-separating.*

4.5 Final Embedding

This section is the crux of this chapter. Bringing the previous sections together, it defines the ultimate mapping, which is later proven to be injective and preserves all of the operations.

Let F be a prime filter of $D(S)$. By Definition 4.3, this F induces a determinative pair. Then, by Theorem 4.4, this determinative pair gives rise to a semigroup homomorphism from S into $\mathcal{P}(X_F)$ for some set X_F. Denote this semigroup homomorphism by letting it take every $x \in S$ to ψ_x^F, where

$$\forall \overline{z} \in X_F \ : \ \overline{z}\psi_x^F = \begin{cases} \overline{zx} \text{ if } D(zx) \in F; \\ \text{undefined otherwise.} \end{cases}$$

Then, define Ψ_x to be $\bigcup_{Fprime} \psi_x^F$, the union of all ψ_x^F where F ranges through all the different prime filters.

An intuitive picture of this potentially obscure concept is the following. Given a prime filter F, every $x \in S$ is taken to ψ_x^F, which is a set of maplets on a set of blocks X_F. Given another prime filter G, every $x \in S$ is taken to ψ_x^G, which is another set of maplets on another set of blocks X_G.

By 'pasting' the functions $x \mapsto \psi_x^F$ and $x \mapsto \psi_x^G$ together, the resulting function is one that takes every $x \in S$ to $\psi_x^F \cup \psi_x^G$, which is itself a set of maplets on the combined set of blocks $X_F \cup X_G$.

This combined set of maplets on $X_F \cup X_G$ remains a valid function because:

1. each of ψ_x^F and ψ_x^G is itself a function;
2. X_F and X_G are disjoint.

When we need to show that a statement holds for Ψ_x, it is enough to show that the same statement holds for ψ_x^G, where G is any arbitrary prime filter of $D(S)$.

Finally, define our ultimate embedding candidate $\Phi : S \to \bigcup_{Fprime} \mathcal{P}(X_F)$ by letting it map every $x \in S$ to Ψ_x.

This map Φ is in fact the same one used in [19], and some parts of the proofs below are adopted from the proofs of [19, Lemma 2.6 and Theorem 2.7].

Theorem 4.17. *The above map Φ is a homomorphism of a semigroup with $\vee$, D, 1 and 0.*

Proof. The following need to be shown. For all $x, y \in S$:

(i) $x\Phi \cdot y\Phi = (xy)\Phi$;

(ii) $x\Phi \vee y\Phi = \big(x \vee y\big)\Phi$;

(iii) $D(x\Phi) = \big(D(x)\big)\Phi$;

(iv) 1Φ is the full identity function on the base set $\bigcup_{F prime} \mathcal{P}(X_F)$;

(v) 0Φ is the empty function on the base set.

The first statement is equivalent to $\Psi_x \cdot \Psi_y = \Psi_{xy}$, and as mentioned earlier, it is enough to show that $\psi_x^F \cdot \psi_y^F = \psi_{xy}^F$ for an arbitrary prime filter F. This principle applies to the other statements too.

(i,iv,v) These statements are the same as ψ^F preserving the semigroup operation, 1 and 0, which was already established by Theorem 4.4 and Corollaries 4.5 and 4.6.

(iii) To prove that $\psi_{D(x)}^F = D(\psi_x^F)$, firstly their domains must be equal. This is true because

$$\begin{aligned} \overline{z} \in dom\big(\psi_{D(x)}^F\big) &\iff D\big(zD(x)\big) \in F \\ &\iff D(zx) \in F \\ &\iff \overline{z} \in dom\big(\psi_x^F\big) = dom\big(D(\psi_x^F)\big). \end{aligned}$$

Then, note that $D(\psi_x^F)$ is an identity restriction; what does $\psi_{D(x)}^F$ do as a function? Let $\psi_{D(x)}^F$ be defined at $\overline{z} \in X_F$; then $\overline{z}\psi_{D(x)}^F = \overline{zD(x)}$ and $D(zx) \in F$. Now, focusing on the relationship between $\overline{z}$ and $\overline{zD(x)}$,

$$\begin{aligned} D(zx) \cdot z &= zD(x) && \text{(R5)} \\ &= D\big(zD(x)\big) \cdot zD(x) && \text{(R1)} \\ &= D(zx) \cdot zD(x). && \text{(R9)} \end{aligned}$$

The above reasoning shows that $z \ \theta_F \ zD(x)$, so $\overline{z} = \overline{zD(x)}$. As $\overline{z}\psi^F_{D(x)} = \overline{zD(x)} = \overline{z}$, $\psi^F_{D(x)}$ is an identity restriction.

Therefore, since $\psi^F_{D(x)}$ and $D(\psi^F_x)$ have the same domain and are both identity restrictions, these two functions are indeed equal.

(ii) Firstly, recall that $x \vee y$ is defined to be $D(x) \vee D(y)$ when x and y are arbitrary elements in S. Hence, let a denote $D(x)$ and b denote $D(y)$, where both $a, b \in D(S)$.

Then, the left-hand side of (ii) can be rewritten as

$$\begin{aligned} x\Phi \vee y\Phi &= D\big(x\Phi\big) \vee D\big(y\Phi\big) \\ &= \big(D(x)\big)\Phi \vee \big(D(y)\big)\Phi && \text{(item (iii) above)} \\ &= a\Phi \vee b\Phi. \end{aligned}$$

Similarly, the right-hand side of (ii) can be rewritten as

$$\begin{aligned} (x \vee y)\Phi &= \big(D(x) \vee D(y)\big)\Phi \\ &= (a \vee b)\Phi. \end{aligned}$$

Therefore, in order to prove (ii), we must show that $\forall a, b \in D(S) : a\Phi \vee b\Phi = (a \vee b)\Phi$, which in turn requires us to prove $\psi^F_{a\vee b} = \psi^F_a \cup \psi^F_b$.

Since both sides of this latest statement are identity restrictions, it is enough to show that the domains of both sides are equal.

$$\begin{aligned} \overline{z} \in dom\big(\psi_{a\vee b}\big) &\iff D\big(z(a \vee b)\big) \in F \\ &\iff D\big(za\big) \vee D\big(zb\big) \in F. \qquad \text{(D4)} \end{aligned}$$

Here, F is a prime filter, so $D\big(za\big) \cup D\big(zb\big) \in F$ means that either $D\big(za\big) \in F$ or $D\big(zb\big) \in F$. Therefore, either $\overline{z} \in dom\big(\psi_a\big)$ or $\overline{z} \in dom\big(\psi_b\big)$; in other words,

$$\begin{aligned} \overline{z} &\in dom\big(\psi_a\big) \cup dom\big(\psi_b\big) \\ &= dom\big(\psi_a \cup \psi_b\big). \end{aligned}$$

The above argument works in the reverse direction too, so we can now conclude that $dom(\psi_{a\vee b}) = dom(\psi_a \cup \psi_b)$, and hence $\psi_{a\vee b} = \psi_a \cup \psi_b$. $\square$

Now that a homomorphism is achieved, the next theorem establishes injectivity.

Theorem 4.18. *The above map Φ is injective.*

Proof. Let $s, t \in S$ with $s \not\leq t$ (and hence $s \neq t$), where the order is induced by viewing the semigroup operation as *meet*, *i.e.* $a \leq b$ if and only if $a \cdot b = a$. Also, let F be a prime filter which separates s and t. Use this F to define ψ_x^F where $x \in S$. It will be shown that for this F, $\psi_s^F \neq \psi_t^F$, and hence $s\Psi \neq t\Psi$.

Firstly, $D(s) \in F$ by Definition 4.11, and

$$\begin{aligned} D\big(D(s) \cdot s\big) &= D(s) \\ &\in F, \end{aligned}$$

so ψ_s^F is defined at $\overline{D(s)}$, and hence $\psi_s^F\big(\overline{D(s)}\big) = \overline{s}$.

Now, we show that ψ_t^F is not equal to ψ_s^F.

Case 1: Assume $D(t) \in F$, and note that

$$\begin{aligned} D\big(D(s) \cdot t\big) &= D\big(D(s) \cdot D(t)\big) && \text{(R9)} \\ &= D(s) \cdot D(t) && \text{(R6)} \\ &\in F, \end{aligned}$$

so ψ_t^F is defined at $\overline{D(s)}$. Then, are $\psi_s^F\big(\overline{D(s)}\big)$ and $\psi_t^F\big(\overline{D(s)}\big)$ equal? Assume (for contradiction) that they are, and this leads to

$$\overline{s} = \overline{D(s)t}.$$

By the definition of our congruence θ_F, there would be an $\mathsf{e} \in F$ which satisfies

$$\mathsf{e} \cdot s = \mathsf{e} \cdot D(s)t.$$

However, $\mathsf{e} \cdot s = \mathsf{e} \cdot D(s)s$, so we have

$$\mathsf{e}D(s)s = \mathsf{e}D(s)t,$$

where $\mathsf{e}D(s) \in F$ because both e and $D(s)$ are in F. This would contradict the premise that F is an (s,t)-separating filter, so it must be the case that $\psi_s^F \neq \psi_t^F$ as they disagree on $\overline{D(s)}$.

Case 2: Assume $D(t) \notin F$.

The property of filters implies that any element 'smaller' than $D(t)$ cannot be in F. We know from the Case 1 that $D\big(D(s)t\big) = D(s)D(t) \leq D(t)$, so $D\big(D(s)t\big)$ cannot be in F, and hence ψ_t^F is undefined at $\overline{D(s)}$. Therefore, $\psi_s^F \neq \psi_t^F$.

In conclusion, as ψ_x^F is a subset of Ψ_x for all $x \in S$, the above has shown that $\psi_s^F \neq \psi_t^F$, which implies $s\Phi \neq t\Phi$. Therefore, Φ is injective. $\square$

Finally, Theorems 4.17 and 4.18 together lead to the following corollary.

Corollary 4.19. *The above map Φ is an embedding from the algebra of programs (*i.e. *a semigroup with $\vee$, D, 1 and 0) into a same algebra of partial functions (*i.e. *$\mathcal{P}(X)$ for some set X).*

This statement, being the culmination of this entire chapter, asserts that there is a representation of the algebra of programs. On this important note, this chapter concludes.

Chapter 5

Tests

After the previous chapter established a representation for the algebra of programs, this chapter will proceed to study the tests and the `if-then-else` operation by proving that the axioms (T1) to (T6) and (G1) to (G2) are sound and complete.

In this chapter, let S be a semigroup (with $\vee$, D, 1 and 0) as before, and let K be an algebra of tests with the operations $\wedge$, $\vee$, $\neg$, T and F.

5.1 Components of a Test

Earlier, Section 3.3 showed that non-halting tests are partial predicates, which can be written as disjoint pairs. The components of a disjoint pair are subsets of the state set, which correspond directly to identity restrictions, as Section 4.1 explained. Therefore, a disjoint pair can in fact be represented by mapping its components to domain elements (*i.e.* elements of $D(S)$), and then the existing semigroup embedding Φ of Corollary 4.19 can be used.

Firstly, the 'true' and 'false' parts of an arbitrary test are identified using the `if-then-else` operation. The 'true' part of a test α is $\alpha[1, 0]$, and its 'false' part is $\alpha[0, 1]$. The following are the axioms involving these parts.

(T1) $(\alpha \vee \beta)[1,0] = \alpha[1,0] \vee \beta[1,0]$

$(\alpha \wedge \beta)[1,0] = \alpha[1,0] \cdot \beta[1,0]$

(T2) $(\alpha \vee \beta)[0,1] = \alpha[0,1] \cdot \beta[0,1]$

$(\alpha \wedge \beta)[0,1] = \alpha[0,1] \vee \beta[0,1]$

(T3) $(\neg\alpha)[s,t] = \alpha[t,s]$ for all $s, t \in S$

(T4) $\alpha[1,0] \cdot \alpha[0,1] = 0$

(T5) $\alpha[1,0], \alpha[0,1] \in D(S)$

(T6) $\alpha[1,0] = \beta[1,0]$ and $\alpha[0,1] = \beta[0,1]$ implies that $\alpha = \beta$

Note also that these axioms are all sound under our functional interpretation: (T1) to (T3) follow from the discussion at the end of Section 3.3, (T4) ensures that $\alpha[1,0]$ and $\alpha[0,1]$ do form a disjoint pair, (T5) asserts that these parts are indeed domain elements, and (T6) enables us to uniquely identify a test by its two parts.

Before venturing forth, we define a new notation for conciseness.

Notation 5.1. For every $\alpha \in K$, let $\alpha_T = \alpha[1,0]$ and $\alpha_F = \alpha[0,1]$.

Now, we define the representation of K.

Definition 5.2. Extend the mapping Φ from Chapter 4 to cover the tests K, and for all $\alpha \in K$, define $\alpha\Phi$ to be $\langle \alpha_T\Phi, \alpha_F\Phi \rangle$.

Since Φ is already a semigroup homomorphism (Theorem 4.17),

$$\begin{aligned} \alpha_T\Phi \cdot \alpha_F\Phi &= (\alpha_T \cdot \alpha_F)\Phi \\ &= 0\Phi && \text{(T4)} \\ &= 0, \end{aligned}$$

so $\alpha_T\Phi$ and $\alpha_F\Phi$ are disjoint, and hence the extended Φ does map K to a set of disjoint pairs. Next comes the main theorem of this section.

Theorem 5.3. *The extended mapping Φ on K in Definition 5.2 is an embedding from K to a set of disjoint pairs.*

Proof. Note that

$$(\alpha \wedge \beta)\Phi = \big\langle \big((\alpha \wedge \beta)_T\big)\Phi, \big((\alpha \wedge \beta)_F\big)\Phi \big\rangle, \text{ and}$$

$$\begin{aligned}(\alpha)\Phi \wedge (\beta)\Phi &= \big\langle (\alpha_T)\Phi, (\alpha_F)\Phi \big\rangle \wedge \big\langle (\beta_T)\Phi, (\beta_F)\Phi \big\rangle \\ &= \big\langle (\alpha_T)\Phi \cdot (\beta_T)\Phi, (\alpha_F)\Phi \cup (\beta_F)\Phi \big\rangle.\end{aligned}$$

In order to show that these two are the same, both of their components will be shown equal.

For the first component, we show $(\alpha_T)\Phi \cdot (\beta_T)\Phi = \big((\alpha \wedge \beta)_T\big)\Phi$, which amounts to $\psi^G_{\alpha_T} \cap \psi^G_{\beta_T} = \psi^G_{(\alpha\wedge\beta)_T}$, where G is any prime filter in $D(S)$. Suppose that $\overline{x} \in dom\big(\psi^G_{\alpha_T} \cap \psi^G_{\beta_T}\big)$. This is equivalent to

$$\begin{aligned}\overline{x} \in dom\big(\psi^G_{\alpha_T}\big) \text{ and } \overline{x} \in dom\big(\psi^G_{\alpha_T}\big) &\iff D(x\alpha_T) \in G \text{ and } D(x\beta_T) \in G \\ &\iff D(x\alpha_T) \cdot D(x\beta_T) \in G.\end{aligned}$$

$$\begin{aligned}\text{But, } D(x\alpha_T) \cdot D(x\beta_T) &= D(x\alpha_T\beta_T) \\ &= D\big(x \cdot (\alpha \wedge \beta)_T\big).\end{aligned}$$

$$\begin{aligned}\text{So, } D(x\alpha_T) \cdot D(x\beta_T) \in G &\iff D\big(x \cdot (\alpha \wedge \beta)_T\big) \in G \\ &\iff \overline{x} \in dom\big(\psi^G_{\alpha\wedge\beta}\big).\end{aligned}$$

The above reasoning established that $dom\big(\psi^G_{\alpha_T} \cap \psi^G_{\beta_T}\big) = dom\big(\psi^G_{(\alpha\wedge\beta)_T}\big)$, and since these functions are all identity restrictions, it follows that $\psi^G_{\alpha_T} \cap \psi^G_{\beta_T} = \psi^G_{(\alpha\wedge\beta)_T}$.

On the other hand, for the second component, we show $\psi^G_{\alpha_F} \cup \psi^G_{\beta_F} = \psi^G_{(\alpha\vee\beta)_F}$. Suppose that $\overline{x} \in \mathit{dom}\big(\psi^G_{(\alpha\vee\beta)_F}\big)$, which is equivalent to $D\big(x \cdot (\alpha \vee \beta)_F\big) \in G$.

$$\begin{aligned} D\big(x \cdot (\alpha \vee \beta)_F\big) &= D\big(x \cdot (\alpha_F \vee \beta_F)\big) && \text{(T2)} \\ &= D(x\alpha_F) \vee D(x\beta_F), && \text{(D5)} \end{aligned}$$

so $D(x\alpha_F) \vee D(x\beta_F) \in G$. Since G is prime, either $D(x\alpha_F) \in G$ or $D(x\beta_F) \in G$, which means that $\overline{x} \in \mathit{dom}\big(\psi^G_{\alpha_F}\big) \cup \mathit{dom}\big(\psi^G_{\beta_F}\big)$. The above showed that $\mathit{dom}\big(\psi^G_{(\alpha\wedge\beta)_F}\big) = \mathit{dom}\big(\psi^G_{\alpha_F}\big) \cup \mathit{dom}\big(\psi^G_{\beta_F}\big)$, and as these functions are all identity restrictions, it follows that $\psi^G_{\alpha_F} \cup \psi^G_{\beta_F} = \psi^G_{(\alpha\wedge\beta)_F}$.

Now that the case of $(\alpha \wedge \beta)\Phi$ is done, the dual case of $(\alpha \vee \beta)\Phi$ can be established in a similar way.

What about $(\neg\alpha)\Phi$? Note that

$$(\neg\alpha)\Phi = \big\langle \psi^G_{(\neg\alpha)_T}, \psi^G_{(\neg\alpha)_F} \big\rangle \text{ and } \neg(\alpha\Phi) = \big\langle \psi^G_{\alpha_F}, \psi^G_{\alpha_T} \big\rangle,$$

so we need to show $\psi^G_{\alpha_F} = \psi^G_{(\neg\alpha)_T}$. Again, as both of these are identity restrictions, it is enough to consider their domains. Starting from the right-hand side,

$$\overline{x} \in \mathit{dom}\big(\psi^G_{(\neg\alpha)_T}\big) \iff D\big(x \cdot (\neg\alpha)_T\big) \in G.$$

However, $x \cdot (\neg\alpha)_T = x \cdot \alpha_F$ by extending (T3), so

$$D\big(x \cdot \alpha_F\big) \in G \iff \overline{x} \in \mathit{dom}\big(\psi^G_{\alpha_F}\big).$$

Therefore, it is indeed true that $\psi^G_{\alpha_F} = \psi^G_{(\neg\alpha)_T}$.

As for the constants T and F:

$$T\Phi = \big\langle 1\Phi, 0\Phi \big\rangle = \big\langle 1, 0 \big\rangle$$
$$F\Phi = \big\langle 0\Phi, 1\Phi \big\rangle = \big\langle 0, 1 \big\rangle,$$

so these are correctly represented too.

Finally, we prove that the test homomorphism Φ is injective. Let $\alpha, \beta \in K$ such that $\alpha\Phi = \beta\Phi$. Then,

$$\big\langle \alpha_T\Phi, \alpha_F\Phi \big\rangle = \big\langle \beta_T\Phi, \beta_F\Phi \big\rangle.$$

By the definition of equality of disjoint pairs (Section 3.3), the above equation is equivalent to

$$\alpha_T\Phi = \beta_T\Phi \text{ and } \alpha_F\Phi = \beta_F\Phi.$$

These latest equations only involve programs, and since Φ is a program embedding, it follows that

$$\alpha_T = \beta_T \text{ and } \alpha_F = \beta_F.$$

Now, by axiom (T6) and Notation 5.1, we can conclude that α and β are equal, and hence Φ is a test embedding. $\square$

5.2 General `if-then-else`

This section defines how Φ acts on `if-then-else` expressions, and proves that Φ preserves the `if-then-else` operation.

An expression like $\alpha[s, t]$ contains both a test and programs, so it is convenient to recast such an expression into one which contains only programs. Intuitively, $\alpha[s, t]$ should be the same as $(\alpha_T \cdot s)$ 'or' $(\alpha_F \cdot t)$, where α_T, α_F, s and t are all programs. However, the abstract program algebra does not have an 'or' or 'union' operation between two arbitrary functions, as functions are not closed under union in general. Hence, the rule of evaluating `if-then-else` expressions is only defined in the image of the representation. Only after the representation maps the abstract program algebra into the realm of concrete functions can we use the set-theoretic union operation.

In short, the first goal of this section is to establish the following equation.

$$(\alpha[s,t])\Phi = (\alpha_T s)\Phi \cup (\alpha_F t)\Phi \tag{5.1}$$

Here, even though $\cup$ is not generally well defined for arbitrary functions, the above equation turns out a special case where the union is in fact valid. To prove the above equation, we show its equivalent:

$$\psi^G_{\alpha[s,t]} = \psi^G_{\alpha_T s} \cup \psi^G_{\alpha_F t}, \tag{5.2}$$

where G is any prime filter in $D(S)$.

The required axioms for this section are:

(G1) $\alpha_T \cdot s = \alpha_T \cdot \alpha[s,t]$

$\alpha_F \cdot t = \alpha_F \cdot \alpha[s,t]$

(G2) $(\alpha_T \vee \alpha_F) \cdot \alpha[s,t] = \alpha[s,t]$

The justification of these axioms' soundness is as follows. The first equation of (G1) means that 'restricting the partial function $\alpha[s,t]$ to where α is true' is the same as 'restricting the partial function s to where α is true'. In other words, let x be an element of the state set such that $x\alpha = T$, *i.e.* $x\alpha_T = x$. Then, $x(\alpha_T \cdot s) = (x\alpha_T)s = xs$, and

$$\begin{aligned} x(\alpha_T \cdot \alpha[s,t]) &= (x\alpha_T)\alpha[s,t] \\ &= x(\alpha[s,t]) \\ &= xs. \end{aligned}$$

On the other hand, if y is an element of the state set such that $y\alpha = F$, then α_T is undefined at y, and hence both sides are undefined at y. Therefore, $\alpha_T \cdot s = \alpha_T \cdot \alpha[s,t]$ is true under our functional interpretation. The soundness of the second equation of (G1) can be achieved using the same argument.

As for (G2), the term $(\alpha_T \vee \alpha_F)$ denotes 'the identity restricted to where α is defined', so the whole equation means that 'restricting $\alpha[s,t]$ to where α halts' is the same as $\alpha[s,t]$ itself. This is again clearly true under our functional interpretation. Therefore, both (G1) and (G2) are sound.

Now, a useful proposition can be further derived from these two axioms.

Proposition 5.4. $D(\alpha[s,t]) = D(\alpha_T \cdot s) \vee D(\alpha_F \cdot t)$

Proof. Apply the D operator on both sides of (G2).

$$\begin{aligned} D\big((\alpha_T \vee \alpha_F) \cdot \alpha[s,t]\big) &= D\big(\alpha_T \cdot \alpha[s,t]\big) \vee D\big(\alpha_F \cdot \alpha[s,t]\big) && \text{(D6')} \\ &= D\big(\alpha_T s\big) \vee D\big(\alpha_F t\big) && \text{(G1)} \end{aligned}$$

□

To show that Equation 5.2 holds, the next three lemmas are established first.

Lemma 5.5. *$dom(\psi^G_{\alpha_T s}) \cap dom(\psi^G_{\alpha_F t}) = \emptyset$.*

Proof. Assume that $\overline{x}$ is in $dom(\psi^G_{\alpha_T s}) \cap dom(\psi^G_{\alpha_F t})$, which means that both $D(x\alpha_T s)$ and $D(x\alpha_F t)$ are in G. Consider their product:

$$\begin{aligned} D(x\alpha_T s) \cdot D(x\alpha_F t) &= D\big(x \cdot D(\alpha_T s)\big) \cdot D\big((x \cdot D(\alpha_F t)\big) && \text{(R9)} \\ &= D\big(x \cdot D(\alpha_T s) \cdot D(\alpha_F t)\big) && \text{Proposition 2.24} \\ &= D\big(x \cdot \alpha_T D(s) \cdot \alpha_F D(t)\big) && \text{(R8) and } \alpha_T = D(\alpha_T) \\ &= D\big(x \cdot D(s) \cdot (\alpha_T \cdot \alpha_F) \cdot D(t)\big) \\ &= D\big(x \cdot D(s) \cdot 0 \cdot D(t)\big) && \text{(T4)} \\ &= 0 \end{aligned}$$

The above reasoning shows that $0 \in G$. However, G is assumed to be a proper filter, so it cannot contain 0. Therefore, it must be the case that both $D(x\alpha_T s)$ and $D(x\alpha_F t)$ cannot be in G at once. □

Lemma 5.6. *$dom(\psi^G_{\alpha[s,t]}) = dom(\psi^G_{\alpha_T s}) \cup dom(\psi^G_{\alpha_F t})$.*

Proof. Consider the join of $D(x\alpha_T s)$ and $D(x\alpha_F t)$.

$$\begin{aligned}
D(x\alpha_T s) \vee D(x\alpha_F t) &= D\big(xD(\alpha_T s)\big) \vee D\big(xD(\alpha_F t)\big) && \text{(R9)}\\
&= D\Big(x\big(D(\alpha_T s) \vee D(\alpha_F t)\big)\Big) && \text{Distributivity}\\
&= D\big(xD(\alpha[s,t])\big) && \text{Proposition 5.4}\\
&= D\big(x\alpha[s,t]\big) && \text{(R9)}
\end{aligned}$$

Finally, $D(x\alpha_T s) \vee D(x\alpha_F t) = D(x\alpha[s,t])$ means that $D(x\alpha_T s)$ is in G if and only if either $D(x\alpha_T s)$ or $D(x\alpha_F t)$ is in G, and this is exactly the lemma statement. □

Lemma 5.7. *If $\overline{x} \in dom(\psi^G_{\alpha_T s}) \cup dom(\psi^G_{\alpha_F t})$, then $\overline{x}\big(\psi^G_{\alpha_T s} \cup \psi^G_{\alpha_F t}\big) = \overline{x}\psi^G_{\alpha[s,t]}$.*

Proof. Firstly, assume that $\overline{x} \in dom(\psi^G_{\alpha_T s})$. We prove that $\overline{x}\psi^G_{\alpha_T s} = \overline{x}\psi^G_{\alpha[s,t]}$, *i.e.* $\overline{x\alpha_T s} = \overline{x \cdot \alpha[s,t]}$, by finding some $d \in G$ such that $d\cdot(x\alpha_T s) = d\cdot(x\cdot\alpha[s,t])$. In fact, $d = D(x\alpha_T)$ is a suitable choice. The assumption of $\overline{x} \in dom(\psi^G_{\alpha_T s})$ means that $D(x\alpha_T s) \in G$. Moreover,

$$D(x\alpha_T) \cdot D(x\alpha_T s) = D(x\alpha_T s),$$

so $D(x\alpha_T) \geq D(x\alpha_T s)$, and hence $D(x\alpha_T) \in G$. Now,

$$\begin{aligned}
D(x\alpha_T) \cdot (x\alpha_T s) &= \big(D(x\alpha_T) \cdot x\alpha_T\big)s\\
&= x\alpha_T \cdot s && \text{(R1)}\\
&= x \cdot \alpha_T s\\
&= x \cdot \alpha_T \alpha[s,t] && \text{(G1)}\\
&= x \cdot D(\alpha_T) \cdot \alpha[s,t] && \text{(because } D(\alpha_T) = \alpha_T\text{)}\\
&= D(x\alpha_T) \cdot x \cdot \alpha[s,t], && \text{(R5)}
\end{aligned}$$

so indeed $x\alpha_T s$ is related to $x\alpha[s,t]$.

Finally, assume that $\overline{x} \in dom(\psi^G_{\alpha_F t})$, and we prove that $\overline{x}\psi^G_{\alpha_F t} = \overline{x}\psi^G_{\alpha[s,t]}$. However, this can be established simply using the same argument as above, and therefore the proof is now complete. $\square$

The three lemmas above culminate in the following theorem.

Theorem 5.8. *The embedding* Φ *satisfies* $\big(\alpha[s,t]\big)\Phi = \big(\alpha_T s\big)\Phi \cup \big(\alpha_F t\big)\Phi$.

Proof. As discussed at the start of this section, this theorem amounts to proving Equation 5.2:

$$\psi^G_{\alpha[s,t]} = \psi^G_{\alpha_T s} \cup \psi^G_{\alpha_F t}.$$

Firstly, Lemma 5.5 showed that the two functions on the right-hand side have disjoint domains, so that the functional union is valid.

Then, Lemma 5.6 proved that the domains of both sides are equal, and Lemma 5.7 demonstrated that the functions on both sides 'do the same action'.

Therefore, the functions on both sides of Equation 5.2 are indeed equal, so the theorem statement follows. $\square$

Next comes the second goal of this section, which is to show that Φ preserves the `if-then-else` operation. Let $\alpha \in K$ and $x, t \in S$. Then, the expression $(\alpha\Phi)[s\Phi, t\Phi]$ is legal because Φ is a representation of both tests and programs. Furthermore, using Definition 5.2, $(\alpha\Phi)[s\Phi, t\Phi]$ can be written as $\big\langle \alpha_T\Phi, \alpha_F\Phi \big\rangle [s\Phi, t\Phi]$. From this, we make the following definition.

Definition 5.9. The expression $\big\langle \alpha_T\Phi, \alpha_F\Phi \big\rangle [s\Phi, t\Phi]$ is equal to

$$\big(\alpha_T\Phi \cdot s\Phi\big) \cup \big(\alpha_F\Phi \cdot t\Phi\big).$$

The validity of this definition is justified as follows. Firstly, since Φ is a semigroup homomorphism (Theorem 4.17),

$$\big(\alpha_T\Phi \cdot s\Phi\big) \cup \big(\alpha_F\Phi \cdot t\Phi\big) = (\alpha_T \cdot s)\Phi \cup (\alpha_F \cdot t)\Phi.$$

Secondly, for the right-hand side of the above equation, Lemma 5.5 implies that the union is well defined because the domains are disjoint, so Definition 5.9 is indeed correct.

At last, we state the main theorem of this section.

Theorem 5.10. *The mapping Φ preserves the* `if-then-else` *operation. In other words,*

$$\big(\alpha[s,t]\big)\Phi = (\alpha\Phi)[s\Phi, t\Phi].$$

Proof. Starting with the left-hand side:

$$\begin{aligned}
\big(\alpha[s,t]\big)\Phi &= (\alpha_T s)\Phi \cup (\alpha_F t)\Phi && \text{Theorem 5.8}\\
&= \big(\alpha_T\Phi \cdot s\Phi\big) \cup \big(\alpha_F\Phi \cdot t\Phi\big) && \text{(because } \Phi \text{ is a homomorphism)}\\
&= \big\langle \alpha_T\Phi, \alpha_F\Phi \big\rangle [s\Phi, t\Phi] && \text{Definition 5.9}\\
&= (\alpha\Phi)[s\Phi, t\Phi]. && \text{Definition 5.2}
\end{aligned}$$

□

In conclusion, this chapter established the soundness and completeness of the axioms (T1) to (T6) and (G1) to (G2). Section 5.1 started by arguing that the axioms (T1) to (T6) are sound, and went on to build a representation from the tests to disjoint pairs, using the existing representation from Chapter 4. Then, Section 5.2 defined the representation of `if-then-else` expressions, and demonstrated that this definition preserves the `if-then-else` operation.

Overall, up to this point in the thesis, Chapters 4 and 5 culminated in a two-sorted representation Φ of the program algebra, as this Φ injectively maps both abstract programs and abstract tests to their concrete counterparts, while preserving every operation. Therefore, the set of the axioms listed in Section 3.7 is indeed sound and complete.

Chapter 6

Equality Test

As introduced in Chapter 3, one of the most common tests in `if-then-else` statements is the *equality test*:

$$(\cdot = \cdot) \;:\; S \times S \to K.$$

Given two functions f and g on X and $x \in X$, the equality test of f and g is the partial predicate defined by:

$$x(f = g) = \begin{cases} T & \text{if both } xf \text{ and } xg \text{ are defined, and } xf = xg; \\ F & \text{if both } xf \text{ and } xg \text{ are defined, but } xf \neq xg; \\ U & \text{if either } xf \text{ or } xg \text{ is undefined.} \end{cases}$$

Just like general tests, the equality test is studied by identifying its true and false parts. This further gives rise to two operations on the semigroup S:

$$(f * g) = (f = g)[1, 0] \text{ and}$$
$$(f \neq g) = (f = g)[0, 1],$$

so $(f * g)$ is the identity restriction to where f and g are both defined and equal, and $(f \neq g)$ is that to where f and g are both defined but not equal.

The above shows that $*$ and $\neq$ can be defined in terms of the $=$ operation. In fact, the other direction is also true. Given the operations $*$ and $\neq$, the equality test $(f = g)$ is implicitly defined to be the operation whose 'true' and 'false' parts are $(f * g)$ and $(f \neq g)$ respectively.

Therefore, axiomatising the equality test simply involves studying these two new operations, which in fact was already done by Jackson and Stokes [19, Sections 2.2 and 2.3]. The next two sections respectively list the axioms of both these operations, and the proofs are outlined using stepping stones from [19].

6.1 Operation of Agreement

As briefly mentioned in the beginning of this chapter, 'the restriction of the identity to where both functions are defined and equal' corresponds to a binary operation $*$ on a left restriction semigroup S, and this operation satisfies the following axioms. For all $x, y, z, w \in S$:

(A1) $(x * x)x = x$;

(A2) $x * y = y * x$;

(A3) $(x * y)x = (x * y)y$;

(A4) $(x * y)(z * w) = (z * w)x * y$;

(A5) $x(y * z) = (xy * xz)x$.

Note that the domain operation D can be expressed in terms of $*$ by $D(x) = x * x$.

The proof of completeness requires the following two lemmas, which are [19, Lemma 2.10 and Lemma 2.11] respectively.

Lemma 6.1. *For every $x, y \in S$, the element $(x * y)$ is the maximum of all $a \in D(S)$ (with respect to the ordering induced by the semilattice meet) such that*

- *$a \leq D(x) \cdot D(y)$;*
- *$ax = ay$.*

Lemma 6.2. *If F is any filter of $D(S)$ and $x, y \in S$, then the following are equivalent:*

- $D(x) \in F$ *and* $\exists \mathtt{f} \in F \ : \ \mathtt{f}x = \mathtt{f}y$;
- $(x * y) \in F$.

Note that in the context of determinative pairs (Section 4.3), the first statement in this lemma is the same as 'F not (x, y)-separating', or 'x and y are not related by θ_F'.

Theorem 6.3. *The mapping Φ is a homomorphism with respect to $*$,* i.e. $x\Phi * y\Phi = (x * y)\Phi$.

Proof. As before, it is sufficient to show that $\psi_x^F * \psi_y^F = \psi_{x*y}^F$ for any prime filter $F \subset D(S)$.

Let $\overline{z} \in dom(\psi_{x*y}^F)$; equivalently, $D\big(z(x * y)\big) \in F$. Now,

$$\begin{aligned} D\big(z(x * y)\big) &= D\big((zx * zy)z\big) && \text{(A5)} \\ &= (zx * zy) \cdot D(z) && \text{(R8)} \\ &= \big(D(z)zx\big) * zy && \text{(A4)} \\ &= zx * zy, && \text{(R1)} \end{aligned}$$

so $(zx * zy) \in F$.

Furthermore, by Lemma 6.2, $(zx * zy) \in F$ is equivalent to

$$\begin{aligned} & D(zx), D(zy) \in F \text{ and } \overline{zx} = \overline{zy} \\ \iff \ & \overline{z} \in dom(\psi_x^F) \text{ and } \overline{z} \in dom(\psi_y^F) \text{ and } \overline{z}\psi_x^F = \overline{z}\psi_y^F \\ \iff \ & \overline{z} \in dom(\psi_x^F * \psi_y^F), \end{aligned}$$

and this shows that the domains of $(\psi_x^F * \psi_y^F)$ and ψ_{x*y}^F are the same. Since both of these two functions are identity restrictions, they are indeed equal. □

6.2 Operation of Disagreement

Contrary to the previous section, 'the restriction of the identity to where both functions are defined but not equal' corresponds to another binary operation $\neq$ on a left restriction semigroup S with 0, and it satisfies the following axioms. For all $x, y, z \in S$ and $a \in D(S)$:

(Z1) $D(x \neq y) = (x \neq y)$;

(Z2) $x(y \neq z) = (xy \neq xz)x$;

(Z3) $(x * y)(x \neq y) = 0$;

(Z4) $a(x \neq y) = (ax \neq ay)$;

(Z5) $(x * y) \leq a$ and $(x \neq y) \leq a$ imply $D(x)D(y) \leq a$.

(These axioms are labelled with the letter Z because this operation of disagreement is antithetical to the operation of agreement in the previous section.)

The proof of this section's main theorem requires the following lemma, which is extracted from the proof of [19, Theorem 2.17].

Lemma 6.4. *Let $s, t \in S$ with $s \neq t$. If F is a maximal filter in $D(S)$ with respect to $(s * t) \notin F$, and $x, y \in S$ with $D(x), D(y) \in F$, then $(x * y) \notin F \iff (x \neq y) \in F$.*

Finally, the next theorem establishes the completeness of the axioms (Z1) to (Z5). Its proof uses a similar argument to [19, Theorem 2.17].

Theorem 6.5. *The mapping* Φ *is a homomorphism with respect to* $\neq$, i.e. $x\Phi \neq y\Phi = (x \neq y)\Phi$.

Proof. As before, it is sufficient to show that $\psi_x^F \neq \psi_y^F = \psi_{x\neq y}^F$ for any prime filter $F \subset D(S)$.

$$\begin{aligned}
&\overline{z} \in dom(\psi_x^F \neq \psi_y^F) \\
&\iff \overline{z} \in dom(\psi_x^F) \text{ and } \overline{z} \in dom(\psi_y^F) \text{ and } \overline{z}\psi_x^F \neq \overline{z}\psi_y^F \\
&\iff D(zx), D(zy) \in F \text{ and } \overline{zx} \neq \overline{zy} \\
&\iff (zx * zy) \notin F && \text{Lemma 6.2} \\
&\iff (zx \neq zy) \in F && \text{Lemma 6.4} \\
&\iff D\big(z(x \neq y)\big) \in F && \text{(Z1) and (Z2)} \\
&\iff \overline{z} \in dom(\psi_{x\neq y}^F)
\end{aligned}$$

The above reasoning shows that the domains of both $(\psi_x^F \neq \psi_y^F)$ and $\psi_{x\neq y}^F$ are equal, and since both of these are identity restrictions, the proof is now complete. $\square$

Chapter 7

Three-Valued Logic

This thesis will conclude with a study of three-valued logic on its own, for the algebra of commutative three-valued tests is interesting in itself. In Chapter 5, the tests were studied in the presence of programs, and the laws satisfied by the tests alone were never explicitly stated, as these laws must follow from the laws of the program algebra. However, what are the axioms of the tests in the absence of programs? This is the subject of this chapter.

This chapter will begin with a brief historical survey on the various generalisations from two-valued to three-valued logic, with an emphasis on the differences between the non-commutative and commutative three-valued logics. These two systems stem from the sequential and parallel evaluation paradigms, respectively. Finally, this chapter will establish an algebraic representation of the commutative three-valued logic.

7.1 Earliest Generalisations

The study of three-valued logic emerged at the beginning of the 20th Century, and the earliest work was the 1920 paper [25] by Łukasiewicz. Not long afterwards, Kleene also introduced his three-valued logic in his 1938 paper [21], and later, he elaborated his study of three-valued logic in his 1952 book [22].

The basic logical connectives in Łukasiewicz's logic are identical to those in Kleene's logic, as shown in the following tables. The way these connectives are defined is the same as the parallel evaluation strategy.

$\wedge$	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

$\vee$	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

$\neg$	
T	F
F	T
U	U

Despite the same basic connectives, Łukasiewicz's and Kleene's logics differ in their interpretations of logical implication. Łukasiewicz's logic defines $U \implies U$ to be T, while Kleene's logic defines $U \implies U$ to be U. In this thesis, $U \implies U$ is considered equivalent to $\neg U \vee U$, as in the classical two-valued logic. Since $\neg U \vee U$ evaluates to U, Kleene's logic is adopted by this thesis.

This branch of generalised logic has now grown into a number of vast subject areas such as many-valued logic and fuzzy logic, and a more detailed history and survey can be found (for example) in the book [2] by Bergmann.

7.2 Conditional Logic

Another branch of generalisation of Boolean logic was initiated by McCarthy in his 1963 paper [27]. In this paper, he not only presented some axioms of `if-then-else`, but also gave an interpretation of the three-valued logical connectives using the short-circuit evaluation strategy, which is common in most mainstream programming languages.

The following tables show the behaviour of these short-circuit logical connectives.

$\wedge$	T	F	U
T	T	F	U
F	F	F	F
U	U	U	U

$\vee$	T	F	U
T	T	T	T
F	T	F	U
U	U	U	U

$\neg$	
T	F
F	T
U	U

The main differences between Kleene's logic and conditional logic are the expressions $U \vee T$ and $U \wedge F$. Using the former as an example, suppose that p and q are programs which should return Boolean values, and that p does not halt and q returns T. When a computer with sequential strategy evaluates $p \vee q$, it evaluates p first, and since p does not halt, the evaluation of the entire expression does not halt either. Hence, $U \vee T$ is U under the sequential strategy. On the other hand, using parallel strategy on the same $p \vee q$, both p and q are evaluated; as soon as q returns T, the value of the entire expression is straightaway determined to be T, even though p has not returned an answer. Hence, $U \vee T$ is T under the parallel strategy.

The algebraic equivalent of this non-commutative *conditional logic* is the *C-algebra*, where the 'C' here stands for 'conditional'. The paper [14] by Guzmán and Squire studied C-algebras on their own, and a number of papers on the algebra of programs (such as [26] and [29]) incorporated the C-algebra as their algebra of tests.

7.3 Kleene Algebras

Just like two-valued logic corresponds to Boolean algebras, and conditional logic to C-algebras, the commutative three-valued logic adopted in this thesis should have an algebraic parallel too. Such an algebraic parallel turns out to be the Kleene algebra.

Definition 7.1. A *Kleene algebra* is an algebra $\langle K; \wedge, \vee, ', 0, 1 \rangle$ of type $(2, 2, 1, 0, 0)$ such that $\langle K; \wedge, \vee, 0, 1 \rangle$ forms a distributive bounded lattice, and the pseudo-complement $'$ satisfies:

1. $(a \wedge b)' = a' \vee b'$ (De Morgan's Laws)

   $(a \vee b)' = a' \wedge b'$

2. $a'' = a$ (Involution)

3. $a \wedge a' \leq b \vee b'$ (Kleene Property)

(It is important to note that this Kleene algebra has no relation to the Kleene algebra of regular languages!)

This very algebraic object has in fact been extensively studied, albeit under many different guises. In 1958, Kalman called this the *normal i-lattice* [20]. The 1991 book by Cleave [6, Chapter 7] and the 1982 paper by Dunn [10] both term it the *normal quasi-Boolean algebra*. On the other hand, the 2008 book by Bergmann [2] and the 2017 paper by Kumar and Banerjee [24] both use the name *Kleene algebra*. This thesis adopts this latest, most recent name.

The main difference between the Kleene algebra and the Boolean algebra is that the law of excluded middle ($x \wedge x' = 0$ and $x \vee x' = 1$) may not hold in a Kleene algebra. Still, a Kleene algebra still satisfies all the properties which hold in a bounded distributive lattice.

In fact, both Kleene algebras and Boolean algebras can be placed in the following chain of generalisation.

$$\text{Boolean algebras} \subset \text{Kleene algebras} \subset \text{De Morgan algebras} \subset \text{Ockham algebras}$$

A more thorough explanation of these various generalisations of Boolean algebras can be found in the book *Ockham Algebras* by Blyth and Varlet [5, pages 4-5].

7.4 Representation of Kleene Algebras

This section shows that the Kleene algebra just described can be represented by the disjoint pairs from Section 3.3. Here, let $B^{\wedge}$ denote the set of disjoint pairs over a Boolean algebra B.

In [24], Kumar and Banerjee has shown that for any Kleene algebra K, there is always a Boolean algebra B_K such that K can be embedded into $B_K^{[2]}$, where

$$B_K^{[2]} = \{\ [a,b] \mid a,b \in B_K\ ;\ a \leq b\},$$

and its operations are

- $[a,b] \wedge [c,d] = [a \wedge c, b \wedge d]$
- $[a,b] \vee [c,d] = [a \vee c, b \vee d]$
- $\neg[a,b] = [\neg b, \neg a]$
- $\top = [1,1]$ and $\bot = [0,0]$

Although $B^{[2]}$ and our goal $B^{\wedge}$ are different, they bear a strong similarity. Intuitively, the disjoint pair $\big\langle a, b\big\rangle$ identifies a partial predicate by its 'true' part a and 'false' part b, while an element $[c, d] \in B^{[2]}$ identifies one by its 'true' part c and its 'non-false' part d. This motivates the following isomorphism.

Theorem 7.2. *Let B be a Boolean algebra. Then, the set of disjoint pairs $B^{\wedge}$ is isomorphic to $B^{[2]}$ just defined above, using the isomorphism $\mu : B^{\wedge} \to B^{[2]}$, which maps every $\big\langle a, b\big\rangle \in B^{\wedge}$ to $[a, \neg b] \in B^{[2]}$.*

Proof. We first show that all operations are preserved. Let $\big\langle a, b\big\rangle, \big\langle c, d\big\rangle \in B^{\wedge}$. Then,

$$\begin{aligned}
\big(\big\langle a, b\big\rangle \wedge \big\langle c, d\big\rangle\big)\mu &= \big\langle a \wedge c, b \vee d\big\rangle\mu \\
&= [a \wedge c, \neg(b \vee d)] \\
&= [a \wedge c, \neg b \wedge \neg d] \\
&= [a, \neg b] \wedge [c, \neg d] \\
&= \big\langle a, b\big\rangle\mu \wedge \big\langle c, d\big\rangle\mu,
\end{aligned}$$

so $\wedge$ is preserved, and the same reasoning applies to the dual $\vee$ too.

As for negation,

$$\begin{aligned}
\big(\neg\big\langle a, b\big\rangle\big)\mu &= \big\langle b, a\big\rangle\mu \\
&= [b, \neg a] \\
&= \neg[a, \neg b] \\
&= \neg\big(\big\langle a, b\big\rangle\mu\big).
\end{aligned}$$

Furthermore, $\big\langle 1, 0\big\rangle\mu = [1, \neg 0] = [1, 1]$ and $\big\langle 0, 1\big\rangle\mu = [0, \neg 1] = [0, 0]$, so the top and bottom elements are preserved.

Next, we show that μ is bijective. Firstly, suppose $\big\langle a_1, a_2\big\rangle\mu = \big\langle b_1, b_2\big\rangle\mu$. Then, $[a_1, \neg a_2] = [b_1, \neg b_2]$, so $a_1 = b_1$ and $a_2 = b_2$. Hence, $\big\langle a_1, a_2\big\rangle = \big\langle b_1, b_2\big\rangle$, so μ is indeed injective.

Also, given $[a_1, a_2] \in B^{[2]}$, it follows from definition that $a_1 \wedge a_2 = a_1$, and hence

$$\begin{aligned} a_1 \wedge a_2' &= (a_1 \wedge a_2) \wedge a_2' \quad \text{(replacing } a_1 \text{ by } a_1 \wedge a_2\text{)} \\ &= a_1 \wedge (a_2 \wedge a_2') \\ &= a_1 \wedge 0 \\ &= 0. \end{aligned}$$

This shows that (a_1, a_2') belongs to $B^{\wedge}$, and $\langle a_1, \neg a_2 \rangle \mu = [a_1, a_2]$, so μ is surjective.

Therefore, this completes the proof of $B^{[2]} \cong B^{\wedge}$. $\square$

As K is embeddable into $B_K^{[2]}$ and $B^{[2]} \cong B^{\wedge}$, we can at last conclude that our algebra of partial predicates is indeed a Kleene algebra!

The axioms of a Kleene algebra were not needed by the algebra of test studied in Chapter 5, since other laws were used to construct the representation. Nevertheless, the laws of a Kleene algebra must be consequences of the previous laws.

References

[1] G. M. Bergman. Actions of boolean rings on sets. *Algebra Universalis*, 28(2):153–187, 1991.

[2] M. Bergmann. *An Introduction to Many-Valued and Fuzzy Logic*. Cambridge University Press, 2008.

[3] G. Birkhoff and J. D. Lipson. Heterogeneous algebras. *Journal of Combinatorial Theory*, 8(1):115–133, 1970.

[4] S. L. Bloom and R. Tindell. Varieties of "if-then-else". *SIAM Journal of Computing*, 12(4):677–707, 1983.

[5] T. S. Blyth and J. C. Varlet. *Ockham Algebras*. Oxford University Press, 1994.

[6] J. P. Cleave. *A Study of Logics*. Oxford University Press, 1991.

[7] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[8] K. Denecke and S. L. Wismath. *Universal Algebra and Applications in Theoretical Computer Science*. CRC Press, 2002.

[9] J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Transactions on Computational Logic*, 7(4):798–833, 2006.

[10] J. M. Dunn. A relational representation of quasi-Boolean algebras. *Notre Dame Journal of Formal Logic*, 23(4):353–357, 1982.

[11] V. Gould. Notes on restriction semigroups and related structures; formerly (weakly) left E-ample semigroups. Retrived from `http://www-users.york.ac.uk/~varg1/restriction.pdf`, 2010.

[12] G. Grätzer. *Universal Algebra*. Springer Verlag, 1979.

[13] I. Guessarian and J. Meseguer. On the axiomatization of "if-then-else". *SIAM Journal of Computing*, 16(2):332–357, 1987.

[14] F. Guzmán and C. C. Squire. The algebra of conditional logic. *Algebra Universalis*, 27(1):88–110, 1990.

[15] J. M. Howie. *Fundamentals of Semigroup Theory*. Oxford University Press, 1995.

[16] M. Jackson and T. Stokes. An invitation to C-semigroups. *Semigroup Forum*, 62(2):279–310, 2001.

[17] M. Jackson and T. Stokes. Semigroups with if-then-else and halting programs. *International Journal of Algebra and Computation*, 19(7):937–961, 2009.

[18] M. Jackson and T. Stokes. Modal restriction semigroups. *International Journal of Algebra and Computation*, 21(7):1053–1095, 2011.

[19] M. Jackson and T. Stokes. Monoids with tests and the algebra of possibly non-halting programs. *Journal of Logical and Algebraic Methods in Programming*, 84(2):259–275, 2015.

[20] J. A. Kalman. Lattices with involution. *Transactions of the American Mathematical Society*, 87(2):485–491, 1958.

[21] S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

[22] S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.

[23] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.

[24] A. Kumar and M. Banerjee. Kleene algebras and logic: Boolean and rough set representations, 3-valued, rough and perp semantics. *Studia Logica*, 105(3):439–469, 2017.

[25] J. Łukasiewicz. On three-valued logic. In L. Borkowski, editor, *Selected works*, pages 87–88. North-Holland, 1970.

[26] E. G. Manes. Adas and the equational theory of if-then-else. *Algebra Universalis*, 30(3):373–394, 1993.

[27] J. McCarthy. A basis for a mathematical theory of computation. In *Computer programming and formal systems*, pages 33–70. North-Holland, 1963.

[28] A. H. Mekler and E. M. Nelson. Equational bases for if-then-else. *SIAM Journal of Computing*, 16(3):465–485, 1987.

[29] G. Panicker, K. V. Krishna, and P. Bhaduri. Axiomatization of `IF-THEN-ELSE` over monoids of possibly non-halting programs and tests. *International Journal of Algebra and Computation*, 27(3):273–298, 2017.

[30] D. Pigozzi. Equality-test and if-then-else algebras: axiomatization and specification. *SIAM Journal of Computing*, 20(4):766–805, 1991.

[31] B. M. Schein. Relation algebras and function semigroups. *Semigroup Forum*, 1(1):1–62, 1970.

[32] B. M. Schein. Lectures on semigroups of transformations. In *Twelve Papers in Logic and Algebra*, volume 113 of *AMS Translations Series 2*, pages 123–181. American Mathematical Society, 1979.