A Topological Framework for Program Semantics

by

Ingrid Moira Rewitzky

AUGUST 1995

A Thesis presented for the Degree of Doctor of Philosophy
Supervisor: Prof Chris Brink

Department of Mathematics and Applied Mathematics
Faculty of Science
University of Cape Town
South Africa

Contents

Abstract

Program semantics can be viewed relationally as in relational semantics, algebraically as in predicate transformer semantics, logically as in information systems and order-theoretically as in denotational semantics. This can be compared to a common situation in non-classical logics. Namely, a logic can often be presented as a formal deductive system, as an algebra and as a relational structure, with each of the presentations derivable from each of the other two. The central hypothesis of this thesis is that this situation can serve as a paradigm for unifying the various versions of program semantics. Starting with a relational semantics based on certain ordered topological spaces, called Priestley spaces, and invoking the techniques of Priestley duality, an algebraic, a logical and an order-theoretic presentation of program semantics are derived. Each of these four presentations are also derivable from each of the other three. The topological model of program semantics based on Priestley spaces thus serves as a unifying framework for other versions of program semantics, essentially as in the logic-algebra-semantics paradigm.

Acknowledgements

I would like to express my sincere thanks to my supervisor Prof Chris Brink for his guidance and encouragement during my PhD studies, and for making the Laboratory for Formal Aspects and Complexity in Computer Science (FACCS-Lab) at the University of Cape Town a pleasant and stimulating work environment in which to carry out my research. This thesis has benefitted from his valuable input and helpful criticisms.

Many other people have taken an active interest in my work. I thank, in particular, Dr Jean Pretorius, Dr Maarten de Rijke and Dr Japie Vermeulen for informative discussions and helpful comments.

I gratefully acknowledge the financial assistance received from the Foundation for Research Development, the University of Cape Town, and the FACCS-Lab during my PhD studies. For the Junior Research Fellowships offered to me in 1994 and 1995, I thank the Department of Mathematics and Applied Mathematics.

I thank the Department of Mathematics and Applied Mathematics for the use of departmental facilities, and for all the opportunities offered to me.

During a study tour of England, Germany and France in 1994 I had the opportunity to discuss my work with numerous people and to present a paper at an international conference. I thank the Foundation for Research Development, FACCS-Lab and the organisers of RelMiCS'94 for making this financially possible. For their hospitality, I thank Dr Jean Pretorius, Dr Hilary Priestley, Prof Ewa Orlowska, and the staff at Schloß Dagstuhl.

Most of all I would like to express my deepest gratitude to my parents and my sister for their understanding and encouragement. As a token of my appreciation I dedicate this thesis to them.

Acknowledgements

I would like to express my sincere thanks to my supervisor Prof Chris Brink for his guidance and encouragement during my PhD studies, and for making the Laboratory for Formal Aspects and Complexity in Computer Science (FACCS-Lab) at the University of Cape Town a pleasant and stimulating work environment in which to carry out my research. This thesis has benefitted from his valuable input and helpful criticisms.

Many other people have taken an active interest in my work. I thank, in particular, Dr Jean Pretorius, Dr Maarten de Rijke and Dr Japie Vermeulen for informative discussions and helpful comments.

I gratefully acknowledge the financial assistance received from the Foundation for Research Development, the University of Cape Town, and the FACCS-Lab during my PhD studies. For the Junior Research Fellowships offered to me in 1994 and 1995, I thank the Department of Mathematics and Applied Mathematics.

I thank the Department of Mathematics and Applied Mathematics for the use of departmental facilities, and for all the opportunities offered to me.

During a study tour of England, Germany and France in 1994 I had the opportunity to discuss my work with numerous people and to present a paper at an international conference. I thank the Foundation for Research Development, FACCS-Lab and the organisers of RelMiCS'94 for making this financially possible. For their hospitality, I thank Dr Jean Pretorius, Dr Hilary Priestley, Prof Ewa Orlowska, and the staff at Schloß Dagstuhl.

Most of all I would like to express my deepest gratitude to my parents and my sister for their understanding and encouragement. As a token of my appreciation I dedicate this thesis to them.

Preface

Broadly, speaking, program semantics is the endeavour of representing the 'meaning' of programs. More specifically, a presentation of program semantics is a mathematical structure of some kind. Programs are treated as some kind of mathematical objects (e.g. functions, multifunctions, or relations) within this structure. Typically, the representation of a program depends on the kind of terminating and nondeterministic behaviour that needs to be captured.

In this thesis I shall be concerned with four versions of program semantics. They are:

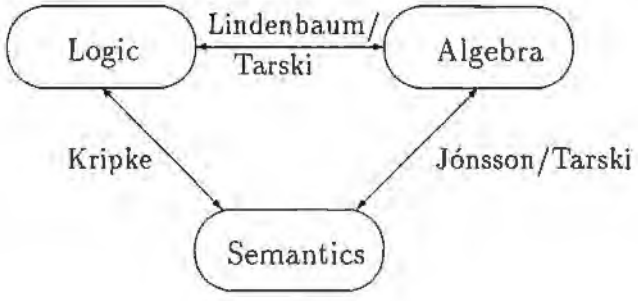
- *Relational semantics*, in which programs are represented as binary input-output relations over a set of states.
- *Predicate transformer semantics*, in which programs are characterised in terms of their action on predicates (properties of states).
- *Information systems*, in which finite units of information about states are structured with a Gentzen-style inference relation and programs are characterised as information-respecting relations.
- *Denotational semantics*, in which states are ordered under an information ordering and programs are represented as domain morphisms.

Note that program semantics can thus be viewed *relationally* as in relational semantics, *algebraically* as in predicate transformer semantics, *logically* as in information systems, or *order-theoretically* as in denotational semantics.

The aim of this thesis is to offer a unifying framework for program semantics. Such a framework allows us to present program semantics in whichever way seems most natural, and then to translate our presentation into other (possibly more familiar) presentations.

To motivate the approach of this thesis, consider the following common situation in non-classical logics. From a given formalisation of, for example, a modal logic, we obtain by the Lindenbaum-Tarski construction an algebra (typically some kind of lattice). Also, by the techniques of Kripke semantics, the logic can be captured by a certain type of relational structure, thought of as a set of worlds endowed with one or more accessibility relations. Moreover, not only does the logic have both an algebraic and a relational representation, but by a technique due to Jónsson and Tarski the algebra and the relational structure also mutually characterise each other.

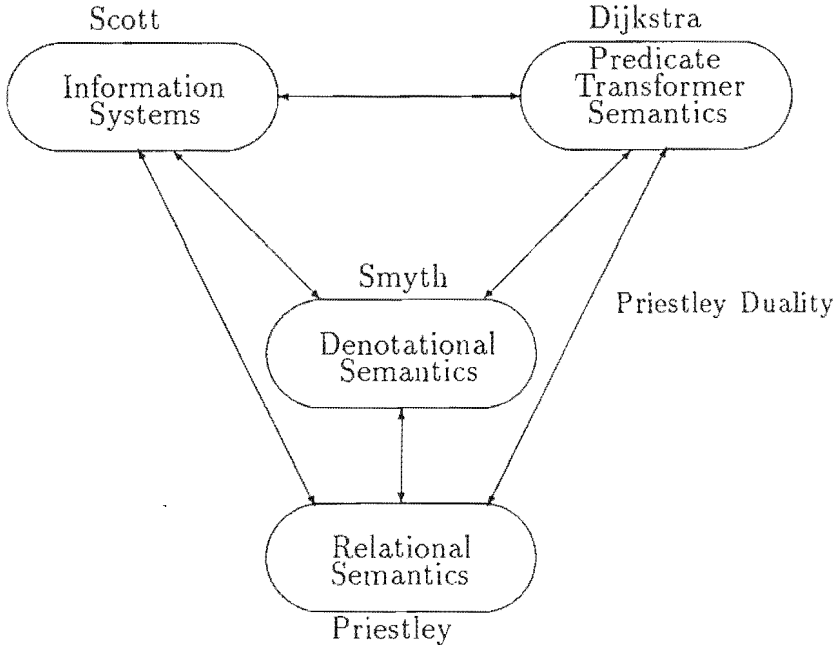
This situation can be pictured in what I shall refer to as the *paradigm triangle* of logic-algebra-semantics, namely:



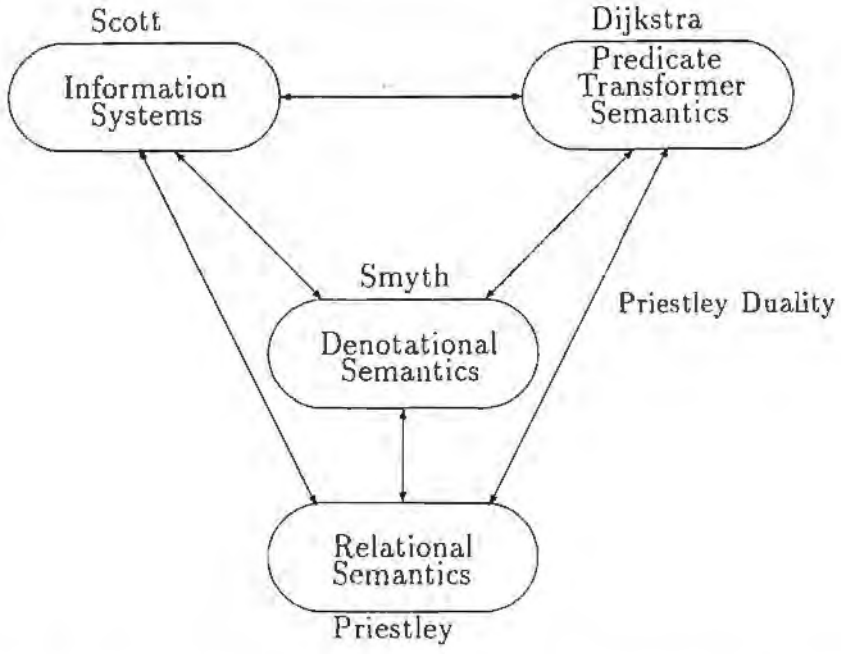
This brings us to the central hypothesis of this thesis: the idea that the triangle of logic-algebra-semantics can also serve as a paradigm for unifying the various presentations of *program semantics.*

Taking the relational semantics as our starting point, we have at the most basic level a set of states endowed with a number of binary relations, each representing a program. This would correspond on the logical side to a multimodal logic, such as Propositional Dynamic logic, and on the algebraic side to a Boolean algebra with unary operators. Although this presentation of relational semantics provides a positive answer to our question, it leaves us unable to express denotational semantics. So a more sophisticated relational semantics is required. A possible extension of the basic relational semantics involves topologising the state space. Topologising reveals the logical and information-theoretic aspects of a state space — open sets are properties of states and the specialisation ordering of the topology places states in a hierarchy of increasing information content. Scott domains provide examples of such state spaces. Although this extension is sufficiently sophisticated to be related to an order-theoretic presentation of program semantics, it may be argued that the logical and information-theoretic aspects of a state space are conceptually different. I give effect to this idea of a state space having a topology and a *separate* order by using certain ordered topological spaces called *Priestley spaces.* Programs are represented as structure-preserving binary relations over Priestley spaces called *opposite Priestley relations.*

Modelling relational semantics as a Priestley space with opposite Priestley relations, and invoking the techniques of *Priestley duality,* I show that the paradigm triangle of logic-algebra-semantics can indeed be applied to program semantics. The algebraic, logical, and order-theoretic presentations of program semantics derived from the chosen relational presentation will stand to each other as pictured in the following adaptation of the paradigm triangle of logic-algebra-semantics, which I shall refer to as the *program semantics triangle*:



The work consists of five chapters. Chapter 1 serves as an introduction to program semantics and the approach followed in this thesis, and provides the technicalities of Priestley duality. In Chapters 2 to 4 I introduce and unify the four above-mentioned presentations of program semantics: relational, algebraic, logical and order-theoretic. The concluding Chapter 5 combines the various results and presents the positive answer to the question raised at the outset.



The work consists of five chapters. Chapter 1 serves as an introduction to program semantics and the approach followed in this thesis, and provides the technicalities of Priestley duality. In Chapters 2 to 4 I introduce and unify the four above-mentioned presentations of program semantics: relational, algebraic, logical and order-theoretic. The concluding Chapter 5 combines the various results and presents the positive answer to the question raised at the outset.

x

Chapter 1

Introduction and Preliminaries

The purpose of this Chapter is to expand on the ideas introduced in the Preface, and to provide the basic terminology and techniques used in this thesis. Section 1 explains some basic intuitions and facts about four versions of program semantics: relational semantics, predicate transformer semantics, denotational semantics and information systems. The approach followed in this thesis is then motivated by a discussion of the relationships between a logic, its algebra and its semantics. Sections 2, 3 and 4 provide the technical definitions and results relevant respectively to Chapters 2, 4 and 3. The notion of *relational Priestley space*, to be used as a basis for a relational model of program semantics in (Chapter 2), is introduced in Sections 5 and 6. These Sections also outline the technique of *Priestley duality* to be used as a tool for translating between the various presentations of program semantics considered in this thesis.

1.1 Introduction

A *program* is a sequence of commands written in some programming language which can be submitted as a unit to a computer. Values stored in computer memory during program execution are denoted in the program by *program variables*. I assume a computer has an unlimited memory capacity, but that at any point in time we may only ever know or access some finite number of values. A (partial) assignment of values to program variables is called a *state*, and the set of all states is called the *state space*. I assume that the input of a program is reflected in the choice of an *initial state*, while a *final state* reflects the output of the program. A *property* about states is a description of the relationships between variables and their values. An *execution* of a program can be viewed as a sequence of states, starting with an initial state and terminating (if at all) in

a final state. If a program terminates it may do so *cleanly* which means that it does so in some state, or *messily* which means that upon termination it is not in one of the states making up the state space. In the latter case the program is said to *abort* [Dij75, Dij76, Gri81, DiS90]. If a program does not terminate it may do so due to some erraneous loop condition or due to some time constraints. In the former case the program is said to abort [Dij75, Dij76, Gri81, DiS90]. In general, I assume programs are *nondeterministic*, in the sense that from any initial state a program may end up in any one of a set of possible final states. The argument for considering nondeterministic programs can be found in (e.g.) ([Gri81], p 111) ([DiS90], p 125). A program is called *finitely nondeterministic* if a finite number of final states is possible from every state, and *unboundedly nondeterministic* if an infinite number of final states is possible from at least one state. The meaning, or *semantics*, of a program is a mathematical description of its behaviour. I now outline four such descriptions of the input-output behaviour of programs: relational semantics, predicate transformer semantics, denotational semantics, and information systems.

In *relational semantics* the idea is that an input-output pair (s, t) of states is related by a program α if program α may reach final state t from initial state s. Since programs, in general, are nondeterministic, the meaning of a program is given by the collection of all its possible input-output pairs. This collection is a binary (input-output) relation on states. Several relational semantics have been proposed to capture the input-output behaviour of programs [HoL74, Plo76, Wan77, Smy78, JaG85, Hoa87, Nel89]. In some of these, (e.g.) [JaG85, Hoa87], it is postulated that a relational semantics should be given in terms of a restricted set of binary relations. The constraints imposed on relations are intended to capture specific manifestations of terminating and nondeterministic behaviour of programs. In order to deal explicitly with nontermination the state space is extended to include an artificial 'bottom' state, denoted by $\perp$. This represents a 'state of nontermination' in the sense that it caters for infinite execution, abortion, 'overflow', 'underflow' (due to a value being out of range), 'break' (due to a deliberate break in program execution), 'undefinedness' (due to say division by zero), etc. Then a pair $(s, \perp)$ is related by a program α if at least one execution of α from the state s does not terminate. An advantage of the relational approach to program semantics is the existence of a calculus of binary relations such as that of Tarski [Tar41] for reasoning about programs. However, some relational operations (for example, complementation) have rather artificial interpretations in the context of program semantics.

In *predicate transformer semantics*, the aim is to capture the meaning of a program as a total function between properties about states. A property about states is called a *predicate*, and a total mapping between predicates a *predicate transformer*. It is common practice to identify a predicate with the set of states in which it is true, and to regard predicate transformers as mappings between sets of states. Constraints called *healthiness conditions* are axiomatised on predicate transformers to capture the terminating and nondeterministic be-

haviour of programs. Since such axiomatisations are equational it is appropriate to think of predicate transformer semantics as an *algebraic presentation* of program semantics. A predicate can be thought of either as a *precondition* for a program, imposed as a constraint on its initial states; or as a *postcondition* for a program, imposed as a constraint on its final states. Then a predicate transformer can be thought of either as a *precondition predicate transformer* mapping postconditions to preconditions for a program, or as a *postcondition predicate transformer* mapping preconditions to postconditions for a program [Dij75, Dij76, Bac81, JaG85, DiS90, BaW89]. A postcondition predicate transformer may seem a more natural description of program behaviour, but in practice it is often difficult to predict the final states of a program given just the program and an initial state. For this reason the precondition predicate transformers are more widely used, the most prominent being Dijkstra's *weakest precondition predicate transformer* introduced in [Dij75, Dij76], and explained and extended in [Gri81, DiS90]. I introduce different kinds of predicate transformers in Section 1.2 in an algebraic context of Boolean algebra, and in Chapter 2 in a topological context.

The *denotational approach* to program semantics is based on the idea that states can be built up from atomic units of information and that the input-output behaviour of a nondeterministic program can be completely described by the input-output behaviour of its deterministic subprograms. The state space is typically endowed with a partial order so as to form a *domain* (an ω-algebraic complete partial order) [Sco70, Sco72], and deterministic programs are represented as mappings between domains and are called *domain morphisms*. (An example of a domain of states is provided in Chapter 1.) To deal with nondeterminism it is required that the power set of the state space also forms a domain — this is the *powerdomain* [Mai87, GuS90]. Then nondeterministic programs are represented as mappings from a domain to its powerdomain. These mappings are called *state transformers* to indicate that initial states are mapped to (sets of) final states for a program. There are a number of different powerdomains, each capturing different kinds of nondeterministic and terminating behaviour of programs. I outline the technicalities of domains and powerdomains in Section 1.3. Standard references on denotational semantics are [Sto77, Gor79, Sch86], and the survey paper [Mos90].

As explained in [DaP90], the starting point for the notion of an *information system*, as first proposed by [Sco82], is the idea of identifying a state with a set of properties true in it and adequate to define it. These properties can be thought of as logical propositions, each describing a finite amount of information. An information system has three constituents: a set H of propositions, a family of finite subsets of H representing consistent information, and a relation $\vdash$ of entailment identifying implied or superfluous information. Axioms are imposed on such structures to formalise commonsense features of entailment and consistency. This axiomatisation makes information systems Gentzen-style sequent systems where multiple conclusions are deduced from multiple premises [Zha94]. Pro-

grams are represented as certain information-respecting relations over consistent information, and are called *approximable mappings* [Sco82, DaP90]. Constraints are imposed on these relations to formalise the preservation of consistency and information. Scott's work has inspired the introduction of other information systems providing logical presentations of various domains [DaP90, DrG90, EdS93], topological spaces [DrG90, EdS93], algebraic structures [EdS93] and ordered topological spaces (in Chapter 3 of this thesis). I outline Scott's information systems in Section 1.3, and introduce a generalisation of these information systems in Chapter 3.

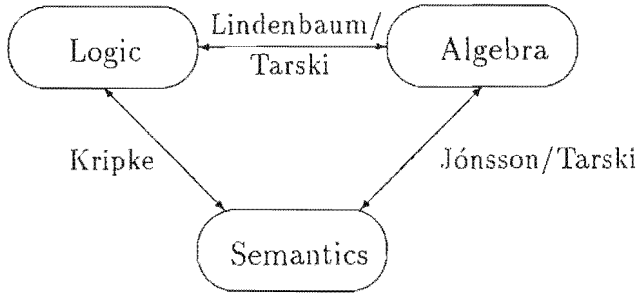
Each of the approaches described in this Section provides a distinctive presentation of the input-output behaviour of (nondeterministic) programs: relational semantics provides a *relational presentation*, predicate transformer semantics an *algebraic presentation*, information systems a *logical presentation* (similar to the so-called *Hoare logic* of [Hoa69] and *program logics* of [Pra76, MiS87]), and denotational semantics an *order-theoretic presentation*. Each can and has been independently investigated, but the full elegance of each presentation can only be obtained by relating it to the others. The question of translating between predicate transformer semantics and relational semantics has been addressed in (e.g) [Gue80, MaC80, Gue82, JaG85, ReB95, Hoa95], and will be addressed in Section 1.2 and Chapter 2 of this thesis. There is also a demonstrated relationship between Dijkstra's predicate transformer semantics and powerdomains [Plo80, Smy83, BoK93, BoK94, BJK95]. A bijective correspondence between domains and information systems is set up in (e.g.) [DrG90, DaP90]. In Chapter 5 I will relate my findings with recent results from the literature.

To motivate the approach I shall use to unify program semantics, consider the following common situation in non-classical logics. A logic, for example modal logic, can be formalised as a logical system using a Hilbert-style axiomatisation or a Gentzen-style axiomatisation. By the Lindenbaum/Tarski [Tar35] construction, the logic can be presented as an algebra of equivalence classes of logical formulae with logical connectives becoming algebraic operations, logical axioms becoming algebraic laws, and logical operators becoming algebraic operators. Properties expressed by logical formulae correspond to conditions on algebraic operators. Then theorems showing the algebra validates all and only the theorems of the logic establish the correspondence between the logic and its algebra.

The logic can be captured by a so-called *Kripke* [Kri63] or *possible-world semantics* given in terms of a relational structure consisting of a set of possible worlds endowed with one or more binary relations between them. A logical formula then corresponds to the set of possible worlds in which it is true. Properties expressed by logical formulae are first- or second-order properties of relations. Soundness and completeness theorems showing that the semantics validates all and only the theorems of the logic establish the correspondence between the logic and its semantics.

Moreover, the algebra and the relational structure corresponding to the logic mutually characterise each other via Jónsson-Tarski duality [JoT51, JoT52]. This result effects a one-one correspondence between binary relations over a set and certain unary operations over the power set of that set. Properties of binary relations translate into properties of the unary operations, and *vice versa.* (Examples of such translations of properties can be found in [Bri93].)

As a general paradigm, this situation can be pictured in what I shall refer to as the *paradigm triangle* of logic-algebra-semantics, namely:



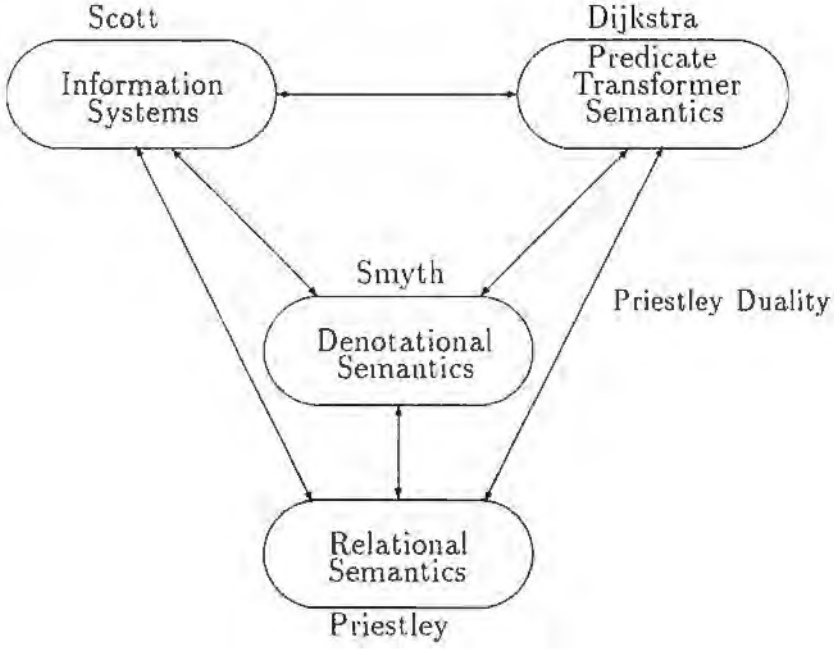
The correspondences between a logic, an algebra and a semantics have been investigated since the 1940s. For example, the modal logic S4 [HuC68, BuS84] corresponds on the algebraic side to a closure algebra [Tar44], and on the semantics side to a quasi-order [JoT51]. Intuitionistic logic [RaS63, Ras74, Gol84] corresponds on the algebraic side to a Heyting algebra [Ras74], and on the semantics side to a partial order. The general picture of such relationships is due to C. Brink and was first presented in [BGO95].

This leads us to the question to be addressed in this thesis: *Can the logic-algebra-semantics triangle serve as a paradigm for unifying program semantics?*

In earlier work [ReB95, DOR94], a positive answer to this question is obtained using the techniques of Jónsson/Tarski duality [JoT51, JoT52]. The chosen relational semantics has an explicit formalisation of nontermination and is based on the restricted set of binary relations given in [Hoa87]. It turns out that the constraints translate via Jónsson/Tarski duality into familiar healthiness conditions of predicate transformers. The resulting algebraic presentation of program semantics is given in terms of a Boolean algebra with certain unary operators. The logic, which has its Kripke semantics given in terms of this relational semantics, is essentially an extension of Propositional logic. It is obtained by introducing a special propositional constant for reasoning about the property of nontermination, and by adding modal-like prefixes indexed by programs and axioms corresponding to the conditions on the relations used to represent programs. However the chosen relational model is relatively unsophisticated in the sense that it leaves us unable to express denotational semantics and provides an over-simplified view of properties of states.

In this thesis I therefore propose a more sophisticated relational structure,

namely a *Priestley space with opposite Priestley relations*, as a relational model of programs. By exhibiting an example of such a structure, and using the techniques of *Priestley duality* I argue that this is a good relational model of programs in the sense that it offers a unifying framework for four versions of program semantics. This framework can be pictured in the following adaptation of the logic-algebra-semantics triangle, which I shall refer to as the *program semantics triangle*:



1.2 Predicate transformers as power operations

In this Section I give two presentations of program semantics: a relational presentation based on an extended state space endowed with a collection of restricted binary relations, and an algebraic presentation based on a Boolean algebra with certain unary operators. I outline how these presentations can be obtained from each other using a construction initiated by Jónsson and Tarski [JoT51]. The ideas outlined in this Section arise from joint work with C. Brink and the technical details can be found in [ReB95].

For the relational semantics, the state space is taken to be the set $U = \mathcal{S} \cup \{\perp\}$ where $\mathcal{S}$ denotes the set of all states and $\perp$ denotes the special 'state of nontermination'. Programs are represented as certain binary relations over U called *execution relations*. Namely:

1.2.1 Definition A binary relation $R \subseteq U \times U$ is an *execution relation* if it satisfies:

(a) for every $s \in U$, $R(s) \neq \emptyset$, where $R(s) = \{t \mid (s,t) \in R\}$ (that is, R is total), and

(b) $R(\bot) = \{\bot\}$ (that is, R is strict).

The constraints imposed on execution relations are not needed to apply Jónsson/Tarski duality but they are convenient, as pointed out in the 'laws of programming' of Hoare *et al* [Hoa87], for reasoning about program behaviour. In particular, restriction (a) means that any program may be activated in any state; while restriction (b) means that execution begun in a 'state of nontermination' never terminates. (Note that no restrictions are placed on the nondeterminism of the programs.) The relational semantics is therefore given in terms of an *execution relational structure* $\mathcal{U} = (U, \mathcal{R})$ consisting of a set U with a special element $\bot$ and endowed with a set $\mathcal{R}$ of execution relations over U.

For the predicate transformer semantics, a program is thought of as a mapping from predicates to predicates. Technically, then, since programs-as-relations have U both as domain and (possibly) as range, it is convenient to model a predicate simply as a subset of U. (This is also done for example in [JaG85] though other papers like [Dij75, Dij76, Plo80] prefer to define a predicate simply as a subset of S, not of U.) As subsets of U, predicates form a Boolean algebra $(\mathcal{P}(U), \bigcup, \bigcap, \neg, \emptyset, U)$, where $\bigcup$, $\bigcap$ and $\neg$ are respectively the set-theoretic operations of union, intersection and complementation, $\emptyset$ is the empty set and represents the predicate *false*, and U represents the predicate *true*. Let $\mathcal{P}(U)_\bot$ be the set of all subsets of U that contain $\bot$ as an element. Generally, any mapping $f : \mathcal{P}(U) \to \mathcal{P}(U)$ is called a *predicate transformer*. Following [BaW93], we say that:

- f is *U-preserving* if $f(U) = U$;
- f is *$\emptyset$-preserving* if $f(\emptyset) = \emptyset$;
- f is *always terminating* if $f(S) = S$;
- f is *strict* if $(Q \in \mathcal{P}(U)_\bot$ iff $f(Q) \in \mathcal{P}(U)_\bot)$;
- f is *conjunctive* if $f(Q_1 \bigcap Q_2) = f(Q_1) \bigcap f(Q_2)$, for any predicates Q_1, Q_2;
- f is *disjunctive* if $f(Q_1 \bigcup Q_2) = f(Q_1) \bigcup f(Q_2)$, for any predicates Q_1, Q_2;
- f is *completely conjunctive* if $f(\bigcap_{i \in I} Q_i) = \bigcap_{i \in I} f(Q_i)$, for any nonempty set of predicates $\{Q_i\}_{i \in I}$;
- f is *completely disjunctive* if $f(\bigcup_{i \in I} Q_i) = \bigcup_{i \in I} f(Q_i)$, for any nonempty set of predicates $\{Q_i\}_{i \in I}$;

- f is *monotonic* if $Q_1 \subseteq Q_2$ implies $f(Q_1) \subseteq f(Q_2)$, for any predicates Q_1, Q_2;
- f is *continuous* if $f(\bigcup_{i\in I} Q_i) = \bigcup_{i\in I} f(Q_i)$, for any increasing (with respect to $\subseteq$) chain of predicates $\{Q_i\}_{i\in I} \subseteq \mathcal{S}$.

Let α be a program. Then Dijkstra's [Dij75, Dij76] weakest precondition predicate transformer for α is, in this context, a predicate transformer $wp_1(\alpha, -) : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ which is $\emptyset$-preserving, conjunctive and continuous. The restriction to subsets of $\mathcal{S}$ means that predicates describing terminating states of α are mapped to predicates describing the states from which α is guaranteed to terminate in a state satisfying Q. The continuity and $\emptyset$-preserving conditions mean that these predicate transformers describe the behaviour of boundedly nondeterministic, terminating programs. Dijkstra's [DiS90] weakest liberal precondition predicate transformer is, in this context, a predicate transformer $wlp(\alpha, -) : \mathcal{P}(U)_\perp \to \mathcal{P}(U)_\perp$ which is U-preserving and conjunctive. Removing the restriction to subsets of $\mathcal{S}$ corresponds to dropping the 'guaranteed to terminate' requirement of $wp_1(\alpha, -)$. Requiring predicates to contain $\perp$ means that these predicate transformers can describe the behaviour of programs for which nontermination from a state is possible. A combination of these two predicate transformers yields a $\emptyset$-preserving, conjunctive predicate transformer, namely $wp_2(\alpha, -) : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ given by:

$$wp_2(\alpha, Q) = wlp(\alpha, Q \cup \{\perp\}) \cap wp_1(\alpha, \mathcal{S})$$

for $Q \subseteq \mathcal{S}$. This predicate transfomer is consistent with Dijkstra and Scholten's [DiS90] notion of weakest precondition predicate transformer, which captures the behaviour of terminating programs including those which are unboundedly nondeterministic. (By consistent, we mean that the defined and the postulated predicate transformers capture the same kind of terminating and nondeterministic behaviour of programs and satisfy the same healthiness conditions.)

The foregoing concepts can also be expressed in the general algebraic context of an arbitrary Boolean algebra $\mathcal{B} = (B, \vee, \wedge, \neg, 0, 1)$. The notions of disjunctive, conjunctive, 0-preserving and 1-preserving appear (in (e.g.) [JoT51]) respectively as *additive*, *multiplicative*, *normal*, and *full*. Strictness can be reformulated in terms of a special set $B_\perp$ in B. A predicate transformer is then a *homomorphism*; a 0-preserving, additive predicate transformer is a *join homomorphism*; and a 1-preserving, multiplicative predicate transformer is a *meet homomorphism*. For example, Dijkstra's weakest liberal precondition predicate transformers are meet homomorphisms, and Dijkstra and Scholten's weakest precondition predicate transformers are normal, multiplicative homomorphisms. A Boolean algebra with predicate transformers is called a *Boolean algebra with unary operators* (in the sense of Jónsson and Tarski [JoT51]). For example, $(\mathcal{P}(U), \mathcal{F}, \mathcal{P}(U)_\perp)$ where $\mathcal{F}$ is the set of all Dijkstra weakest liberal precondition predicate transformers, is a Boolean algebra with meet homomorphisms.

The question addressed in [ReB95] is what kind of predicate transformers correspond to execution relations. For this we used a construction initiated by Jónsson/Tarski [JoT51] and called a *power construction* by Brink [Bri93]. This involves defining for each relation over a universe U a *power operation* over the power set of U. The terminology *power operation* of Brink indicates the process of lifting a structure of *elements* of a set to *subsets* of that set.

Jónsson/Tarski's construction is as follows. Let $(U, \mathcal{R})$ be a relational structure. For each $R \in \mathcal{R}$, a mapping $R^{\dagger} : \mathcal{P}(U) \to \mathcal{P}(U)$ can be defined by

$$R^{\dagger}(Q) = \{x \mid (\exists y \in Q)[(x, y) \in R]\}$$

for $Q \subseteq U$. (This is the notation of [Bri93], where the operation $R^{\dagger}$ is called the *power operation* of R.) Then $\mathcal{P}(U)$ is a Boolean algebra. The operations are normal and completely additive, and hence join homomorphisms. The structure $(\mathcal{P}(U), \{R^{\dagger} \mid R \in \mathcal{R}\})$ is called the *power algebra* [Bri93] (or the *complex algebra* [Gol89]) of the relational structure $(U, \mathcal{R})$. Note that variations of the definition of the power operation of a relation can be obtained: Replacing '$\exists$' with '$\forall$' yields the *dual* of $R^{\dagger}$, replacing '$(x, y) \in R$' by '$(y, x) \in R$' yields the *conjugate* of $R^{\dagger}$, and performing both replacements yields the *residual* of $R^{\dagger}$. Since the resulting operations can be related (using notions of duals [Hal74], conjugates [JoT51] and residuals [BlJ72]) it suffices to consider one translation.

Let $(\mathcal{B}, \mathcal{F})$ be a Boolean algebra with join homomorphisms. Let $\mathcal{X}B$ be the set of all ultrafilters in $\mathcal{B}$. To turn $\mathcal{X}B$ into a relational structure a collection of binary relations over $\mathcal{X}B$ is needed. For each $f \in \mathcal{F}$, a binary relation $\mathcal{X}f \subseteq \mathcal{X}B \times \mathcal{X}B$ is defined in [JoT51] by:

$$\mathcal{X}f = \{(P, Q) \mid Q \subseteq f^{-1}(P)\}.$$

Then the (ultrafilter) structure $(\mathcal{X}B, \{\mathcal{X}f \mid f \in \mathcal{F}\})$ is a relational structure. The Jónsson/Tarski representation theorem ([JoT51] Theorem 3.9) for Boolean algebras with join homomorphisms states that:

1.2.2 Theorem *Every Boolean algebra with join homomorphisms is isomorphic to the power algebra of some relational structure.* □

Using this construction, C. Brink and I have shown [ReB95] that $\emptyset$-preserving, completely disjunctive predicate transformers $f : \mathcal{P}(U) \to \mathcal{P}(U)$ can be viewed as power operations of programs-viewed-as-relations $R \subseteq U \times U$, and that the properties of execution relations translate into conditions of U-preserving and strictness. (Predicate transformers with these properties are consistent with a generalisation of Dijkstra's weakest precondition predicate transformers given in [JaG85].)

The algebraic structure related to an execution relation structure in [ReB95] is, therefore, a *predicate transformer algebra* $(\mathcal{B}, B_{\perp}, \mathcal{F})$ where $\mathcal{B}$ is a Boolean

algebra, $B_\perp$ is a special ultrafilter in $\mathcal{B}$, and $\mathcal{F}$ is a collection of 1-preserving, strict join homomorphisms $f : B \to B$. Then:

1.2.3 Theorem *Every predicate transformer algebra can be represented as a subalgebra of the predicate transformer algebra of some execution relational structure.* □

On the other hand:

1.2.4 Theorem *Every execution relational structure can be represented as a substructure of the execution relational structure of some predicate transformer algebra.* □

That is, the techniques of Jónsson/Tarski already suffice for a reasonable translation between a basic relational model of programs and predicate transformer semantics. (Note: As pointed out earlier, the purpose of this thesis is to go beyond this work, and give a more sophisticated and more realistic version of such translations.)

Dijkstra [Dij75, Dij76] used weakest precondition predicate transformers to give the semantics of programs written in a small nondeterministic programming language which has been called Dijkstra's *guarded command language*. In Chapter 2, I consider some programs written in this programming language. So I define its syntax here, as in [Gri81], by appealing to the reader's knowledge of mathematics and programming and by effecting some restrictions.

There are two special atomic programs, *skip* and *abort*. The former has the effect of doing nothing, while the latter always fails to reach a final state. Next, there is a whole class of atomic programs called *assignment statements*. These are of the form '$z := e$', where z is a program variable and e is some expression (for example, arithmetical). This statement is intuitively understood as 'z becomes e' or 'assign to z the value of e'. From the atomic programs, compound programs can be constructed in one of three ways. First, any two programs α and β may be *composed* into another program $\alpha;\beta$, intuitively understood as '*do* α, *then do* β. Second, for any predicates B_1 and B_2 and programs α_1 and α_2, there is an *alternative command* IF of the form:

$$\text{if } B_1 \to \alpha_1 \,[\!]\, B_2 \to \alpha_2 \text{ fi}$$

intuitively understood as 'select some true B_i and execute the corresponding α_i'. If both B_1 and B_2 evaluate to false, or if at least one of B_1 or B_2 is undefined, then the program will abort ([Dij76], p 26). Third, there is an *iterative command* DO of the form:

$$\text{do } B_1 \to \alpha_1 \,[\!]\, B_2 \to \alpha_2 \text{ od}$$

intuitively read as 'Repeat the following until no longer possible: select some true B_i and execute the corresponding α_i'. In the case where both B_1 and B_2

initially evaluate to false, the iterative command simply skips ([Dij76], p 39). Note that each B_i ensures that the corresponding α_i is only executed under certain constraints; it is therefore called a *guard*. Each '$B_i \rightarrow \alpha_i$' is called a *guarded command*. For this reason this programming language is sometimes referred to as a *guarded command language*. (Here a command *is* a program.) Nondeterminacy can be introduced when the two guards in an IF or DO construct are not mutually exclusive.

1.3 Domains and the Smyth powerdomain

In this Section I introduce the technicalities of domains and outline two constructions of the Smyth powerdomain: an order-theoretic construction called *completion by ideals* and a topological construction due to Smyth [Smy83]. (For the order-theoretic and topological notions used but not defined in this Section refer to Appendices 1 and 2. For further details refer to [Mai87, GuS90].)

To define domains, we begin with a poset $\mathcal{D} = (D, \sqsubseteq)$. A subset $X \subseteq D$ is called *directed* if for any $x, y \in X$ there is some $z \in X$ with $x \sqsubseteq z$ and $y \sqsubseteq z$. That is, any two elements of a directed set X have an upper bound in X. $\mathcal{D}$ is called a *complete partial order*, or *cpo*, if every directed subset $X \subseteq D$ has a least upper bound, $\bigsqcup X$. If in addition $\mathcal{D}$ has a least element $\bot$ ('bottom') then $\mathcal{D}$ is called a *directed cpo* or *dcpo*. An element $u \in D$ is called *compact* (or *isolated* or *finite*) if it has the property that whenever $u \sqsubseteq \bigsqcup X$, for any directed set X, then there is some $x \in X$ such that $u \sqsubseteq x$. A cpo $\mathcal{D} = (D, \sqsubseteq)$ is said to be *algebraic* if the compact elements form a *basis* for $\mathcal{D}$ (that is, if for any element $u \in D$, the set $X = \{x \mid x \sqsubseteq u \text{ and } x \text{ is compact}\}$ is directed and $u = \bigsqcup X$). If in addition the set of compact elements is countable, then $\mathcal{D}$ is said to be *countably algebraic*, or *ω-algebraic* for short.

1.3.1 Definition [GuS90] A *domain* is an ω-algebraic dcpo.

A standard example of a domain is one constructed from any pre-ordered set $\mathcal{A} = (A, \leq)$ with a bottom element $\bot$, by a process called *completion by ideals*. This is as follows. A subset $I \subseteq A$ in a pre-order $\mathcal{A} = (A, \leq)$ is called an *ideal* if it is a directed set which is downclosed, that is, if $y \in I$ and $x \leq y$ then also $x \in I$. Let $\mathcal{I}(A, \leq)$ be the set of ideals contained in A. Then as shown in (e.g.) [GuS90], the set of ideals $(\mathcal{I}(A, \leq), \subseteq)$ ordered by inclusion is a domain. The compact elements of this domain are exactly the *principal* ideals, that is, downclosures of singleton sets $\{x\}, x \in A$.

Intuitively, we may think of the elements of a domain $\mathcal{D} = (D, \sqsubseteq)$ as states or units of information about variables. Then the ordering is information-theoretic in the sense that going up in the ordering corresponds to increasing information: $x \sqsubseteq y$ if the information encoded by x *approximates* the information encoded

by y, in the sense that at least the information encoded by x is encoded by y. A directed set has the property that any two units of information in that set can be subsumed under another more comprehensive unit of information, and that there is some limit to this process which may be unattainable in a finite amount of time. The compact elements represent information attainable in a finite amount of time. Algebraicity reflects the idea that all information can be constructed from information attainable in a finite amount of time.

The motivation for introducing powerdomains as a next step is to obtain a computable analogue of the powerset construction in order to capture nondeterminism denotationally ([Gun90], p 306). The three standard powerdomains are called, respectively, the *Hoare* (or *lower* or *angelic*) powerdomain, the *Smyth* (or *upper* or *demonic*) powerdomain [Smy78], and the *Plotkin* (or *convex*) powerdomain [Plo76]. Each of these can be constructed in a number of ways [Mai87, GuS90]. I outline the construction using completion by ideals of the Smyth powerdomain and present Smyth's [Smy83] topological hyperspace characterisation thereof.

The completion by ideals construction of the Smyth powerdomain is as follows. Let $\mathcal{D} = (D, \sqsubseteq)$ be a domain, and let D^0 be the set of all compact elements in D. Let $\mathcal{M}(D)$ be the set of all *finite nonempty* subsets of D^0. Define on $\mathcal{M}(D)$ the *Smyth* ordering $\sqsubseteq_1^+$ by:

$$X \sqsubseteq_1^+ Y \text{ iff } (\forall y \in Y)(\exists x \in X)[x \sqsubseteq y]$$

for every $X, Y \in \mathcal{M}(D)$. (This notation is that of [Bri93] where $\sqsubseteq_1^+$ is called the *upper power order* to indicate the idea that the ordering $\sqsubseteq$ on elements of D is lifted to an ordering on subsets of D.) Intuitively, this can be read as saying 'X approximates Y' in the sense that every element of Y is approximated by some element of X. The Smyth ordering can be expressed using upper sets as follows ([Vic89], Proposition 11.1.1):

1.3.2 Theorem $X \sqsubseteq_1^+ Y$ iff $\uparrow Y \subseteq \uparrow X$ *for every* $X, Y \in \mathcal{M}(D)$. □

With this characterisation of the Smyth ordering, it is easily shown that:

1.3.3 Theorem *The Smyth ordering on* $\mathcal{M}(D)$ *is a preorder.* □

The completion by ideals of the preordered set $(\mathcal{M}(D), \sqsubseteq_1^+)$ is the Smyth powerdomain. That is,

1.3.4 Definition For every domain $\mathcal{D} = (D, \sqsubseteq)$, $(\mathcal{I}(\mathcal{M}(D), \sqsubseteq_1^+), \subseteq)$ is the *Smyth powerdomain.*

If the elements of a domain $(D, \sqsubseteq)$ are states, then the elements of its Smyth powerdomain $(\mathcal{I}(\mathcal{M}(D), \sqsubseteq_1^+), \subseteq)$ are sets of sets of states. However only sets of states are needed for describing the final states of nondeterministic programs.

Therefore this domain structure is one level too sophisticated for dealing with nondeterminism. Smyth's [Smy83] topological characterisation is more compatible with our intuitive reason for requiring powerdomains. This characterisation is based on an idea due to Vietoris [Vie21] that from a given topological structure over elements of a set, a topological structure can be defined over certain sets of elements of that set. This topological structure on sets is called a *hyperspace* [Smy83] or *power space* [Smy83, Bri93] of the original space.

The topological construction of the Smyth powerdomain is as follows. Let $\mathcal{D} = (D, \sqsubseteq)$ be a domain. The *Scott topology* on D, denoted by ΣD, has as basis the sets $\uparrow\{a\}$ for compact elements $a \in D$ ([Smy83], p 663). Let $\mathcal{P}_{\uparrow cl}(D)$ denote the collection of all upclosed subsets of D, and $Cl(D)$ the collection of all non-empty closed subsets of D. Topologise this set with a topology τ_u having as basic elements the sets $\{C \in Cl(D) \mid C \subseteq O\}$ for $O \in \Sigma D$. In this way the *upper power space* [Smy83, Bri93] of the domain $\mathcal{D} = (D, \sqsubseteq)$ can be defined. That is,

1.3.5 Definition For every domain $\mathcal{D} = (D, \sqsubseteq)$, $(\mathcal{P}_{\uparrow cl}(D), \tau_u)$ is the *upper power space.*

The following Lemma shows that the specialisation order of the upper power space is the superset order on sets in $\mathcal{P}_{\uparrow cl}(D)$,

1.3.6 Lemma *For every domain* $\mathcal{D} = (D, \sqsubseteq)$, $\forall A, B \in \mathcal{P}_{\uparrow cl}(D)$, $A \leq_{\tau_u} B$ iff $A \supseteq B$. □

With some technical adjustments for non-emptiness, Smyth ([Smy83] Theorem 3) shows that:

1.3.7 Theorem *For every domain* $\mathcal{D} = (D, \sqsubseteq)$ *endowed with the Scott topology, the upper power space in its specialisation order is isomorphic to the upper powerdomain of* $\mathcal{D}$. □

Elements of the Smyth powerdomain are of the form $\uparrow X$ ($X \subseteq D$). Intuitively, we may think of a set $\uparrow X$ as describing the output of a program from a given initial state provided we are willing to accept every state in $\uparrow X$ as a possible final state for the program. The compact elements are upclosures of singleton sets and hence describe the output of deterministic programs. Therefore, mappings from the domain $\mathcal{D}$ to its Smyth powerdomain describe the input-output behaviour of programs. In the Smyth ordering, the fewer elements a set contains, the better it is in the ordering and all sets containing the bottom element $\perp$ are identified. In terms of the output of programs, this means that the output of a program is better than that of another program (from a given state) if it has fewer nondeterministic choices (from that state). There is no distinction between the output, from a given state, of programs which may not terminate from that state.

A technique compatible with Smyth's topological construction of the Smyth powerdomain will be used in Chapter 4 to contruct a domain from an ordered topological space.

1.4 Scott's information systems

In this Section I introduce Scott's [Sco82] information systems, and outline the connection between domains and information systems.

An information system is a structure of the form $(H, \mathrm{Con}, \vdash)$ where H is a collection of propositions that can be made about states, Con is a family of consistent finite sets of propositions and $\vdash$ is an entailment relation.

Two notions of information system are given in [Sco82]: one with an entailment relation $\vdash \subseteq \mathrm{Con} \times H$, and the other with $\vdash \subseteq \mathrm{Con} \times \mathrm{Con}$. Since the first is a special case of the second. I will use the second notion of information system.

1.4.1 Definition A *Scott information system* is a structure $(H, \Delta, \mathrm{Con}, \vdash)$ where H is a set of propositions, Δ is a distinguished member of H, Con is a set of finite subsets of H and $\vdash$ is a binary relation between members of Con such that for all $U_1, U_2, V_1, V_2, W \in \mathrm{Con}$,

(a) $\emptyset \vdash \{\Delta\}$

(b) $U_1 \vdash V_1$ implies $U_1 \cup V_1 \in \mathrm{Con}$

(c) $U_1 \vdash U_1$ (reflexivity of $\vdash$)

(d) $U_1 \vdash W$ and $W \vdash V_1$ imply $U_1 \vdash V_1$ (transitivity of $\vdash$)

(e) $U_1 \subseteq U_2$ and $U_1 \vdash V_1$ and $V_2 \subseteq V_1$ implies $U_2 \vdash V_2$

(f) $U_1 \vdash (V_1 \cup V_2)$ implies $U_1 \vdash V_1$ and $U_1 \vdash V_2$.

Examples and properties of information systems can be found in (e.g.) [Sco82, DaP90]. An example of a Scott information system is provided in Chapter 3.

In order for an information system to generate a domain, we need a way of constructing the elements of this domain from the propositions of the information system. This is provided by the notion of *element* of an information system. Generalising Scott's ([Sco82], Definition 3.1) definition of an element generated by an information system (with $\vdash \subseteq \mathrm{Con} \times H$), I define a notion of an element generated by a Scott information system as follows:

1.4.2 Definition Let $(H, \Delta, \mathrm{Con}, \vdash)$ be a Scott information system. A subset $X \subseteq H$ is called an *element generated by* H if

(a) every finite subset of X is in Con, and

(b) $U \subseteq X$ and $U \vdash V$ implies $X \cap V \neq \emptyset$ for all $U, V \in \mathrm{Con}$.

Let $\mathcal{D}H$ denote the set of all elements generated by a Scott information system $(H, \Delta, \mathrm{Con}, \vdash)$. The poset $(\mathcal{D}H, \subseteq)$ of elements partially ordered by (set) inclusion is called the *information domain associated with* $(H, \Delta, \mathrm{Con}, \vdash)$ and $(H, \Delta, \mathrm{Con}, \vdash)$ is called the *representing information system of* $(\mathcal{D}H, \subseteq)$. The compact elements of this domain are the upclosures (under $\vdash$) of sets in Con — that is, sets $U^{\vdash} = \{x \mid U \vdash \{x\}\}$ for $U \in \mathrm{Con}$.

A Scott information system can be regarded as a Gentzen-style deduction system, where multiple conclusions are deduced from multiple premises [Zha94]. The entailment relation $U \vdash V$ can be interpreted as indicating that the finite conjunction of propositions in U entails the finite disjunction of propositions in V. Then an element X is a set of propositions which is deductively-closed and consistent. It is deductively-closed in the sense that no finite conjunction of propositions chosen from X can generate a set of conclusions which contain no propositions within X; it is consistent in the sense that $A \nvdash \emptyset$ for every finite $A \subseteq X$.

There is an alternative characterisation of elements in $\mathcal{D}H$, which as pointed out by Scott [Sco82], makes the notion of information domain compatible with the notion of domain arising in domain theory. For any element $X \in \mathcal{D}(H)$,

$$X = \bigcup \{U^{\vdash} \mid U \in \mathrm{Con} \text{ and } U \subseteq X\}.$$

Intuitively, this says that any element of the domain $(\mathcal{D}H, \subseteq)$ can be thought of as the limit of its finite approximations. The precise connection between domains in domain theory and information domains associated with information systems is given as follows in (e.g.) ([DaP90], p68):

1.4.3 Theorem *Every domain is order-isomorphic to the information domain associated with its information system of compact elements.* □

For a logical presentation of program semantics based on information systems, a representation of programs as some kind of mappings or relations between information systems is needed. In Scott [Sco82] these are given as structure-preserving mappings between information systems called *approximable mappings*. The word 'mapping' is used to indicate that these relations are counterparts of multifunctions (used in domain theory), and the reason for using 'approximable' is explained in detail in [Sco82].

1.4.4 Definition Let $(H, \Delta, \mathrm{Con}, \vdash)$ be a Scott information system. A *Scott approximable mapping* $f : H \to H$ is a binary relation $f \subseteq \mathrm{Con} \times \mathrm{Con}$ such that for any $U_1, U_2, V_1, V_2 \in \mathrm{Con}$:

(a) $\emptyset f \emptyset$

(b) $U_1 f V_1$ and $U_1 f V_2$ imply $U_1 f (V_1 \cup V_2)$

(c) $U_2 \vdash U_1$, $U_1 f V_1$ and $V_1 \vdash V_2$ imply $U_2 f V_2$.

(This is exactly Scott's definition ([Sco82] Definition 5.1.)

Intuitively, $U_1 f V_1$ can be read as saying 'given consistent information U_1 the program f produces consistent information V_1 as output, and the amount of information produced by the program depends on the amount of information provided as input'. These conditions are interpreted in [Sco82] as follows. Condition (a) means that no information about the output can be obtained from no information abou the input. Condition (b) means that the output corresponding to a fixed input is cumulative. Condition (c) describes how f interacts with the entailment relation.

The results of this Section provide the necessary background for Chapter 3 where generalisations of the notions of Scott information system and Scott information-respecting relation are considered as a logical presentation of program semantics.

1.5 Priestley spaces and Priestley duality

A *Priestley space* is a certain kind of partially ordered topological space. These spaces, also called *ordered Stone spaces* [Pri70, DaP90, Smy92], are used to represent bounded distributive lattices essentially in the way that a Boolean algebra can be represented as a field of sets. The correspondence between Priestley spaces and bounded distributive lattices, established by Priestley [Pri70, Pri72, Pri84] has been referred to as *Priestley duality* in (e.g.) [CLP91].

As explained in [DaP90], the representation results of Stone and Birkhoff lead Priestley [Pri70] to use ordered topological spaces. The standard Stone [Sto36] representation results state that any Boolean algebra is isomorphic to an algebra of sets and, more generally, that a distributive lattice is lattice isomorphic to a lattice of sets. Stone [Sto37] and Birkhoff [Bir33] made these characterisations more explicit by considering prime ideals. (For a definition of prime ideals see Appendix 1.) In the case of a finite distributive lattice L, Birkhoff *ordered* the set of prime ideals $\mathcal{I}L$ with set inclusion, and showed that L is lattice isomorphic to the lattice of lower sets of $(\mathcal{I}L, \subseteq)$. In the case of a Boolean algebra L, Stone *topologised* the set of prime ideals $\mathcal{I}L$ with a so-called *Boolean space topology* $\Omega_{\mathcal{I}L}$ (having as basis the collection $\{X_a \mid a \in B\}$ where $X_a = \{X \in \mathcal{I}B \mid a \in X\}$

for $a \in B$), and showed that B is isomorphic to the algebra of clopen sets of the *Boolean space* $(\mathcal{I}L, \Omega_{\mathcal{I}L})$. Combining these two ideas, Priestley [Pri70] presented a representation result for bounded distributive lattice which emphasises both the order and the topology of the set of prime ideals of a bounded distributive lattice.

Before defining a Priestley space, I provide some facts some about ordered topological spaces that are directly relevant to Priestley spaces and Priestley duality. (For definitions about ordered sets and topology not given here refer to Appendices 1 and 2.) An ordered topological space consists of a non-empty set X, a topology τ and a partial order $\leq$. In general, the order is not the specialisation order of τ. Any topological space (X, τ) can be regarded as an ordered topological space under the discrete (partial) ordering ($x \leq y$ iff $x = y$); and any poset $(X, \leq)$ can be regarded as an ordered topological space under the discrete topology ($\tau = \mathcal{P}(X)$). Let $(X, \leq, \tau)$ be an ordered topological space. A subset $A \subseteq X$ is a *clopen upper set* if it is both a clopen set and an upper set. Dually, a subset $A \subseteq X$ is a *clopen lower set* if it is both a clopen set and a lower set. Clopen upper sets and clopen lower sets are related by set-theoretic complement (since upper and lower sets are set-theoretic complements and the complement of a clopen set is a clopen set). Any ordered topological space has at least two clopen upper and clopen lower sets — namely the empty set and the full space X. A sufficient condition for the existence of other clopen upper and clopen lower sets is a notion of *disconnectedness* for ordered topological spaces. In particular, an ordered topological space $(X, \leq, \tau)$ is *totally order disconnected* with respect to $\leq$ if for any two distinct elements $x, y \in X$ with $x \not\leq y$, there exists a clopen upper set containing y and not x and a clopen lower set containing x and not y. (This implies the Hausdorff separation property given in Appendix 2.)

A *Priestley space* is defined as follows ([Gol89, DaP90, CLP91]):

1.5.1 Definition A *Priestley space* is an ordered topological space $(X, \leq, \tau)$ where:

(a) $(X, \leq)$ is a poset, and

(b) $(X, \leq, \tau)$ is a compact, totally order-disconnected space.

A Boolean space can be regarded as a Priestley space under the discrete order; the lattice of clopen upper sets of a distributive lattice with the discrete topology is a Priestley space. Domains can also be topologised to form Priestley spaces (see (e.g.) [Pri94a, Pri94b] and Chapter 2 of this thesis.)

A *Priestley space morphism* is an order-preserving homeomorphism between Priestley spaces. Priestley spaces $(X, \leq_X, \tau_X)$ and $(Y, \leq_Y, \tau_Y)$ are 'essentially the same' if there exists a map ψ from X onto Y which is simultaneously an order-isomorphism and a homeomorphism. I shall call such a map an *order-homeomorphism* and say that $(X, \leq_X, \tau_X)$ and $(Y, \leq_Y, \tau_Y)$ are *order-homeomorphic*.

The next theorem collects together some properties of Priestley spaces proved in (e.g.) [Gol89, DaP90].

1.5.2 Theorem *Let $(X, \leq, \tau)$ be a Priestley space. Then*

(a) *the clopen upper sets and their complements form a subbasis for the topology τ.*

(b) *τ has a basis of clopen sets.*

(c) *$x \leq y$ in X iff $y \in U$ implies $x \in U$ for every clopen upper set U.*

(d) *for any closed upper set C in X and $x \in C$, there exists a clopen upper set U such that $C \subseteq U$ and $x \notin U$.* □

The clopen upper subsets of a Priestley space form a base for a topology on X called the *upper topology* [Gol89]. Similarly, their complements form a base for the *lower topology* on X. Then by Theorem **1.5.2**, the topology of a Priestley space is the join of these two weak topologies, that is the topology having as subbasic elements the basic elements of the upper and lower topologies. This alternative characterisation of a Priestley space is due to Goldblatt [Gol89].

I now outline the relationship between Priestley spaces and bounded distributive lattices. Ideals, specifically prime ideals, are usually used in formulations of Priestley duality (see (e.g.) [Pri70, Pri72, DaP90]). Here, as in (e.g.) [Gol89], I use prime filters since they have more natural interpretations in the context of program semantics. Given the relationship between prime filters and prime ideals of a lattice, the proofs of theorems about prime filters stated here invoke the same arguments as those for theorems about prime ideals appearing in the literature, and are therefore not given.

I start by showing that a Priestley space can be obtained from a bounded distributive lattice. Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. Define $\mathcal{F}L$ to be the set of all prime filters of L, partially ordered by inclusion. For each $a \in L$, define

$$\sigma_L(a) = \{F \in \mathcal{F}L \mid a \in F\}.$$

Let $\mathcal{S} = \{\sigma_L(a) \mid a \in L\} \bigcup \{\mathcal{F}L - \sigma_L(a) \mid a \in L\}$. The elements $\sigma_L(a)$ for $a \in L$ are upper sets and the elements $\mathcal{F}L - \sigma_L(a)$ for $a \in L$ are lower sets. (Note that $\mathcal{S}$ is not closed under finite intersections.) Let

$$\mathcal{B} = \{\sigma_L(a) \bigcap (\mathcal{F}L - \sigma_L(b)) \mid a, b \in L\}.$$

Then $\mathcal{B}$ contains $\mathcal{S}$ since L is bounded, and $\mathcal{B}$ is closed under finite intersections. Define a topology $\Omega_{\mathcal{F}L}$ having $\mathcal{B}$ as basis (or $\mathcal{S}$ as subbasis), that is, $U \in \Omega_{\mathcal{F}L}$ iff U is a union of members of $\mathcal{B}$. Then $(\mathcal{F}L, \subseteq, \Omega_{\mathcal{F}L})$ is the *prime filter space* associated with $(L, \vee, \wedge, 0, 1)$ and by ([DaP90] Theorem 10.14):

1.5.3 Theorem *The prime filter space $(\mathcal{F}L, \subseteq, \Omega_{\mathcal{F}L})$ of any bounded distributive lattice $(L, \vee, \wedge, 0, 1)$ is a Priestley space.* □

(A similar construction will be used in Chapter 3 to construct a Priestley space from a (Priestley) information system.)

Let $(X, \leq, \tau)$ be a Priestley space. Define $\mathcal{U}(X)$ to be the set of all clopen upper subsets of X. Then $\mathcal{U}(X)$ is closed under finite unions and finite intersections; and hence (as shown in ([Gol89], p 12) ([DaP90], p 199)),

1.5.4 Lemma *The lattice of clopen upper sets $(\mathcal{U}(X), \subseteq)$ of any Priestley space $(X, \leq, \tau)$ is a bounded distributive lattice.* □

One part of Priestley duality is given in Theorem **1.5.5** ([DaP90], Theorem 10.20), and a complementary part in Theorem **1.5.6** ([DaP90], Theorem 10.19).

1.5.5 Theorem (Representation Theorem for bounded distributive lattice) *Every bounded distributive lattice is lattice isomorphic to the lattice of clopen upper subsets of some Priestley space.* □

1.5.6 Theorem (Representation Theorem for Priestley spaces) *Every Priestley space is order-homeomorphic to the prime filter space of some bounded distributive lattice.* □

The final part of Priestley duality involves establishing a bijective correspondence between bounded distributive lattice homomorphisms and Priestley space morphisms. The crucial point for obtaining this bijective correspondence is that, for any lattice homomorphism the preimage of a prime filter is a prime filter. Also for any Priestley space morphism the preimage of a clopen upper set is a clopen upper set. With these bijective correspondences a full categorical duality is obtained between bounded distributive lattices and Priestley spaces. The meaning of Priestley duality is that algebraic notions on bounded distributive lattices can be translated into topological notions on Priestley spaces, and *vice versa*. For example, as shown in [Pri72, Pri84, Pri85]:

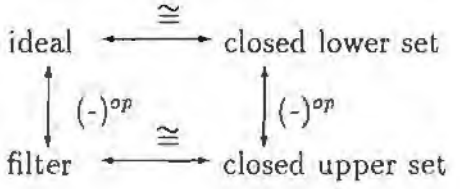
1.5.7 Theorem

(a) *The ideal lattice of a bounded distributive lattice is lattice isomorphic to the lattice of closed lower sets of the dual Priestley space.*

(b) *The filter lattice of a bounded distributive lattice is lattice isomorphic to the lattice of closed upper sets of the dual Priestley space.*

(c) *The principal filter lattice of a bounded distributive lattice is lattice isomorphic to the lattice of clopen upper sets of the dual Priestley space.* □

These lattice isomorphisms can be pictured as follows:



where $(-)^{op}$ reverses the relevant order on the bounded distributive lattice and on the Priestley space, and $\cong$ denotes a lattice isomorphism. The correspondence between filters and closed upper sets will be used in the context of denotational semantics in Chapter 4.

As a consequence of Priestley duality we have, that any Priestley space is a cpo ([DaP90], Exercise 10.9(iv)), but it is not the case that every Priestley space is a domain. In Chapter 4 I show how a domain can be constructed from a Priestley space.

1.6 Relational Priestley spaces and Priestley duality

A *relational Priestley space* is a Priestley space endowed with a collection of certain binary relations — that is, a special relational structure. Jónsson/Tarski [JoT51, JoT52] first put forward the idea of associating a relational structure with a Boolean algebra with operators. Since a Priestley space is associated with a bounded distributive lattice, the Jónsson/Tarski result suggests that relational Priestley spaces can be associated with bounded distributive lattices with unary operators. In fact, this has been done in [Gol89] for bounded distributive lattices *with* meet and/or join homomorphisms, and for bounded distributive lattices *and* join homomorphisms in [CLP91]. (The difference between 'bounded distributive lattices *with* operators', and 'bounded distributive lattices *and* operators' is that the former structure allows only operators over the *same* distributive lattice; while the latter allows operators over possibly *different* distributive lattices. In other words bounded distributive lattices *with* operators are special kinds of bounded distributive lattices *and* operators.) There is a difference in the translations used by Goldblatt and Cignoli: Cignoli uses the Jónsson/Tarski translation given in Section 1.2 to define a join homomorphism from a binary relation, while Goldblatt uses the translations yielding respectively the conjugate and residual of the Cignoli join homomorphism. The Goldblatt translations yield dual homorphisms.

In this Section I superimpose Priestley's representation theorem for bounded distributive lattices on Jónsson/Tarski's representation result for Boolean alge-

bras with operators (outlined in Section 1.2) to obtain a representation theorem for bounded distributive lattice with *join* homomorphisms. (That is, I use the Jónsson/Tarski translations between join homomorphisms and relations, and Priestley's translations between Priestley spaces and bounded distributive lattices). These translations make this result a special case of Cignoli's representation theorem. I then point out how, by reversing orders, a representation theorem for bounded distributive lattice with meet homomorphisms can be obtained. (This representation result will be used in Chapter 2 to translate between an algebraic and a relational presentation of program semantics based on Priestley spaces.)

I now present, as a special case of Cignoli *et al* [CLP91] results, a representation theorem for bounded distributive lattices *with* join homomorphisms, and its complementary part for Priestley spaces with certain relations. The relations over Priestley spaces corresponding to join homomorphisms are defined (in [CLP91] Definition 1.2) as follows:

1.6.1 Definition Let $(X, \leq, \tau)$ be a Priestley space. Then a relation $R \subseteq X \times X$ is called a *Priestley relation* if

(a) images of points are closed lower sets [i.e. $\forall x \in X$, the set $R(x) = \{y \in X \mid (x, y) \in R\}$ is a closed lower set]; and

(b) existential inverse images of clopen upper sets are clopen upper sets [i.e. for every clopen upper set $V \subseteq X$, $R{:}\exists V$ is a clopen upper set in X, where $R{:}\exists V := \{x \in X \mid R(x) \bigcap V \neq \emptyset\}$].

Denote the class of Priestley spaces with Priestley relations by Pspace+R. Denote the class of bounded distributive lattices with join homomorphisms by Dlat+J.

Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. Then given a join homomorphism $j : L \to L$, a relation $\mathcal{F}j \subseteq \mathcal{F}L \times \mathcal{F}L$ is defined, in [CLP91], by:

$$\mathcal{F}j = \{(P, Q) \in \mathcal{F}L \times \mathcal{F}L \mid Q \subseteq j^{-1}(P)\}.$$

With this translation Cignoli *et al* ([CLP91], Ex 1.3 (iv)) show that:

1.6.2 Theorem *For every bounded distributive lattice with join homomorphism $((L, \vee, \wedge, 0, 1), \mathcal{J})$, the prime filter space $(\mathcal{F}L, \subseteq, \Omega_{\mathcal{F}L})$ endowed with relations $\mathcal{F}j$ one for each $j \in \mathcal{J}$ is a Priestley space with Priestley relations.* □

Let $(X, \leq, \tau)$ be a Priestley space. Then given a Priestley relation $R \subseteq X \times X$, a mapping $\mathcal{U}R : \mathcal{U}(X) \to \mathcal{U}(X)$ is defined in [CLP91] by:

$$\mathcal{U}R(V) = \{x \in X \mid R(x) \cap V \neq \emptyset\}$$

for any clopen upper set $V \in \mathcal{U}(X)$. With this translation Cignoli *et al* ([CLP91], Lemma 1.5(i)) show that:

1.6.3 Theorem *For every Priestley space with Priestley relations $((X, \leq, \tau), \mathcal{R})$, the lattice of clopen upper sets $(\mathcal{U}X, \subseteq)$ with mappings $\mathcal{U}R$ one for each $R \in \mathcal{R}$ is a bounded distributive lattice with join homomorphisms.* □

The Cignoli *et al* extension of Priestley's representation theorem (here Theorem **1.5.5**) for bounded distributive lattices with join homomorphisms is given in Theorem **1.6.4**. (The proof may be found in ([CLP91], Lemma 1.5 (iv)).)

1.6.4 Theorem *Every bounded distributive lattice with join homomorphisms $((L, \vee, \wedge, 0, 1), \mathcal{J})$ is lattice isomorphic to the lattice of clopen upper sets with join homomorphisms of some Priestley space with Priestley relations.* □

(Here the isomorphism is a lattice isomorphism ψ over $(L, \vee, \wedge, 0, 1)$ which preserves the join homomorphisms in the sense that for every $j \in \mathcal{J}$, $\psi \circ j = \mathcal{U}\mathcal{F}j \circ \psi$.)

The Cignoli *et al* extension of Priestley's representation theorem (here Theorem **1.5.6**) for Priestley spaces is given in Theorem **1.6.5**. (The proof may be found in ([CLP91], Lemma 1.5 (v)).)

1.6.5 Theorem *Every Priestley space with Priestley relations $((X, \leq, \tau), \mathcal{R})$ is order-isomorphic to the prime filter space with Priestley relations of some bounded distributive lattice with join homomorphisms.* □

(Here the order-isomorphism is an order-homeomorphism ϵ over $(X, \leq, \tau)$ which is structure-preserving in the sense that for every $R \in \mathcal{R}$ and $x, y \in X$, $(x, y) \in R$ iff $(\epsilon(x), \epsilon(y)) \in \mathcal{F}\mathcal{U}R$.)

I now show how to obtain a representation theorem for bounded distributive lattices with meet homomorphisms from that of Cignoli *et al* for bounded distributive lattices with join homomorphisms. Meet homomorphisms over a bounded distributive lattice are in a sense 'opposite' to join homomorphisms, so the corresponding relations over the dual Priestley space are 'opposite' to Priestley relations. Therefore I define a notion of *opposite Priestley relation* by interchanging 'upper' and 'lower' in the definition of a Priestley relation. That is,

1.6.6 Definition Given a Priestley space $(X, \leq, \tau)$, a relation $R \subseteq X \times X$ is an *opposite Priestley relation* if

(a) images of points are closed upper sets [i.e. $\forall x \in X$, the set $R(x) = \{y \in X \mid (x, y) \in R\}$ is a closed upper set]; and

(b) existential inverse images of clopen lower sets are clopen lower sets [i.e. for every clopen lower set $V \subseteq X$, $R :_\exists V$ is a clopen lower set in X where $R :_\exists V = \{x \in X \mid R(x) \cap V \neq \emptyset\}$].

A more useful form of (b) uses the universal inverse image operation defined by:

$$R{:}_\forall V := (R{:}_\exists(V'))' = \{x \in X \mid R(x) \subseteq V\}.$$

for any clopen upper set $V \in \mathcal{U}X$. Then,

(b)′ universal inverse images of clopen upper sets are clopen upper sets [i.e. for every clopen upper set $V \subseteq X$, $R{:}_\forall V$ is a clopen upper set in X].

Denote the class of Priestley spaces with opposite Priestley relations by Pspace+opR. Denote the class of bounded distributive lattices with meet homomorphisms by Dlat+M. Priestley/Cignoli's duality can now be restated as a duality between Dlat+M and Pspace+opR.

Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. Then given a meet homomorphism $g : L \to L$, define a relation $\mathcal{F}g \subseteq \mathcal{F}L \times \mathcal{F}L$ by:

$$\mathcal{F}g = \{(P, Q) \in \mathcal{F}M \times \mathcal{F}L \mid g^{-1}(P) \subseteq Q\}.$$

This relation can be shown to be an opposite Priestley relation by invoking Theorem **1.6.2** as follows: take a bounded distributive lattice with meet homomorphisms, reverse the order on the lattice to obtain a bounded distributive lattice with join homomorphisms, apply Theorem **1.6.2** to obtain a Priestley space with Priestley relations, and then reverse the order on the Priestley space to obtain a Priestley space with opposite Priestley relations.

1.6.7 Theorem *For every bounded distributive lattice with meet homomorphisms $((L, \vee, \wedge, 0, 1), \mathcal{M})$, the prime filter space $(\mathcal{F}L, \subseteq, \Omega_{\mathcal{F}L})$ with relations $\mathcal{F}g$, one for each $g \in \mathcal{M}$, is a Priestley space with opposite Priestley relations.* □

Let $(X, \leq, \tau)$ be a Priestley space. Then given an opposite Priestley relation $R \subseteq X \times X$, define a mapping $\mathcal{U}R : \mathcal{U}(X) \to \mathcal{U}(X)$ by:

$$\mathcal{U}R(V) := R{:}_\forall V = \{x \in X \mid R(x) \subseteq V\}$$

for any clopen upper set $V \in \mathcal{U}(X)$. This mapping can be shown to be a meet homomorphism by invoking Theorem **1.6.3** as follows: take a Priestley space with opposite Priestley relations, reverse the order on the Priestley space to obtain a Priestley space with Priestley relations apply Theorem **1.6.3** to obtain a bounded distributive lattice with join homomorphisms, and then reverse the order on the lattice to obtain a bounded distributive lattice with meet homomorphisms.

1.6.8 Theorem *For every Priestley space with opposite Priestley relations $((X, \leq, \tau), \mathcal{R})$, the lattice of clopen upper sets $(\mathcal{U}X, \subseteq)$ with mappings $\mathcal{U}R$, one for each $R \in \mathcal{R}$, is a bounded distributive lattice with meet homomorphisms.* □

The representation theorem, in Theorem **1. 6. 9**, for bounded distributive lattices with meet homomorphisms can be proved by taking a bounded distributive lattice with meet homomorphisms, reversing the order on the lattice to obtain a bounded distributive lattice with join homomorphisms, applying Theorem **1. 6. 4** to obtain the isomorphic bounded distributive lattice with join homomorphisms, and finally reversing the order on the lattice to obtain the required bounded distributive lattice with meet homomorphisms.

1. 6. 9 Theorem *Every bounded distributive lattice with meet homomorphisms* $((L, \vee, \wedge, 0, 1), \mathcal{M})$ *is lattice isomorphic to the lattice of clopen upper sets with meet homomorphisms of some Priestley space with opposite Priestley relations.* □

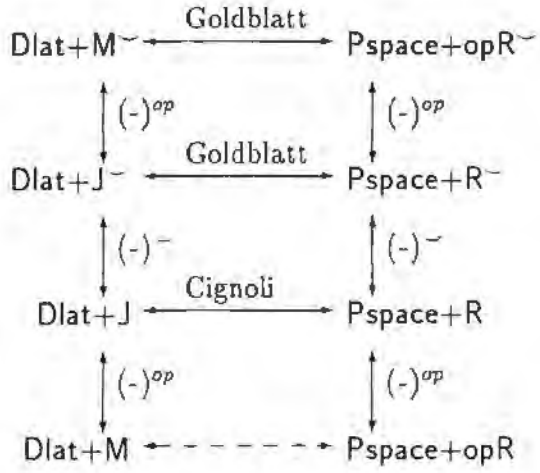
(Here the isomorphism is a lattice isomorphism w over $(L, \vee, \wedge, 0, 1)$ which preserves the meet homomorphisms in the sense that for every $g \in \mathcal{M}$, $\psi \circ g = \mathcal{UF}g \circ \psi$.)

The complementary representation theorem for Priestley spaces with opposite Priestley relations is given in Theorem **1. 6. 10**. This can proved by taking a Priestley space with opposite Priestley relations, reversing the order to obtain a Priestley space with Priestley relations, applying Theorem **1. 6. 5** to obtain the homeomorphic Priestley space with Priestley relations, and finally reversing the order on the Priestley space to obtain the required Priestley space with opposite Priestley relations.

1. 6. 10 Theorem *Every Priestley space with opposite Priestley relations* $((X, \leq, \tau), \mathcal{R})$ *is order-isomorphic to the prime filter space with opposite Priestley relations of some bounded distributive lattice with meet homomorphisms.* □

(Here the order-isomorphism is an order-homeomorphism ϵ over $(X, \leq, \tau)$ which is structure-preserving in the sense that for every $R \in \mathcal{R}$ and $x, y \in X$, $(x, y) \in R$ iff $(\epsilon(x), \epsilon(y)) \in \mathcal{FU}R$.)

The relationship between the results of this Section and those of Goldblatt [Gol89] can be summarised as follows:



The $(-)^{op}$ reverses the relevant order on distributive lattices and on Priestley spaces, but leaves the homomorphisms and relations unchanged. The $(-)^{\sim}$ reverses the direction of the homomorphisms (i.e. $f : X \to Y$ becomes $f^{\sim} : Y \to X$) and the relations (i.e. $R \subseteq X \times Y$ becomes $R^{\sim} \subseteq Y \times X$), but leaves the orders unchanged.

The results of this Section will be used in Chapter 2 to define relational and an algebraic presentation of program semantics based on Priestley spaces, and to translate between them.

Chapter 2

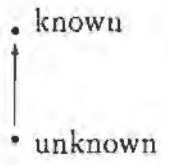
Relational Semantics and Predicate Transformer Semantics

In this Chapter I put forward the idea that the notion of a relational Priestley space (from Section 1.3) provides a suitable relational model of program semantics. I exhibit an example of a Priestley space with programs, namely the universal domain T^{ω} [Plo78] topologised with the Scott topology and its dual. Using this example I advance three arguments in favour of using a Priestley space as a state space. First, a Priestley space provides a setting in which the logical and information-theoretic aspects of a state space can be appreciated. Second, natural topological characterisations of computational concepts such as observability, nondeterminism, termination, and predicate transformers can be obtained. Third, considering Dijkstra's [Dij76] ideas in the context of a Priestley space gives rise to natural relational counterparts of healthiness conditions imposed on predicate transformers.

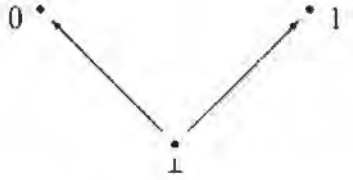
2.1 A domain of states

Let $Var = \{v_1, v_2, v_3, \ldots\}$ be a denumerable set of program variables. For the purposes of this Chapter I assume that all the variables are of type `Bool`. I do not assume that the value of every variable is known. (For example, in a distributed system it would be unrealistic to assume that the value of every variable is known at every point in time.) If the value of a variable is *known*, then the variable

denotes either 0 or 1; if the value of a variable is *unknown*, then the variable still denotes either 0 or 1 but it is not known which. I introduce a special symbol '$\perp$' which means 'it is unknown whether the value is 0 or 1' as the denotation of a variable with unknown value. The philosophical motivation for this idea is that I want a theory of *information* (an epistemological theory) rather than a theory of *existence* (an ontological theory). That is, I assume that at any point in time some information is known, but do not assume everything is always known. So I distinguish between 'the known' and 'the unknown' assuming 'the known' is better than 'the unknown'. Then the basic ordering of information is:



The actual information is given by the values of the (known) variables. Here I do not assume any value is better than another; hence the ordering on denotations of variables is that of the three-valued truth value cpo T (that is, $0 \leq_T 0$, $1 \leq_T 1$, $\perp \leq_T \perp, 0, 1$).



In this context, I represent a state as an infinite ordered vector in T^ω (the Cartesian product of ω copies of T). That is, a state s is a vector $(s(v_1), s(v_2), \ldots)$ where $s(v_i)$ is the value of variable v_i in state s and $s(v_i) \in T$. I assume the values of variables in these vectors are ordered in the sense that the value of variable v_i occurs in position i of the vector. Let State be the set of all states. Then State $= T^\omega$. I call a state *atomic* if all except one of its values are $\perp$; *finite* if all except a finite number of its values are $\perp$; and *maximal* if none of its values is $\perp$. Let Atom denote the set of all atomic states, and State^o the set of all finite states. Then Atom $\subseteq$ State^o. I assume $(\perp, \perp, \ldots) \in$ State^o but $(\perp, \perp, \ldots) \notin$ Atom. (The reason for this assumption will become apparent in the proof of Theorem **2.2.4**.)

The ordering on State is the componentwise ordering $\leq$ inherited from the ordering $\leq_T$ on T . That is,

2.1.1 For $x, y \in \mathsf{State}$,

$$x \leq y \text{ iff } \forall i > 0,\ x(v_i) \leq_{\mathsf{T}} y(v_i)$$
$$\text{iff } \forall i > 0,\ (x(v_i) = \bot \text{ or } x(v_i) = y(v_i)).$$

An interpretation of this ordering can be obtained by thinking of a state (in State) as an encoding of information. Let s be a state. Then the *information content* of s, denoted I_s, is given by the set $\{v_i \mid s(v_i) \neq \bot\}$ of variables known in state s. The variables in I_s could have value 1 or value 0 (and it is known which). Accordingly, the *positive information content* of s, denoted P_s, is the set $\{v_i \mid s(v_i) = 1\} \subseteq I_s$ of variables with value 1 in state s; and the *negative information content* of s, denoted N_s, is the set $\{v_i \mid s(v_i) = 0\} \subseteq I_s$ of variables with value 0 in state s. Note that $P_s \cup N_s = I_s$. Now states can be ordered with respect to their positive and negative information contents as follows:

2.1.2 For $x, y \in \mathsf{State}$,

$$x \sqsubseteq y \text{ iff } P_x \subseteq P_y \text{ and } N_x \subseteq N_y.$$

The ordering '$x \sqsubseteq y$' can be read as 'x is an *approximation* of y' in the sense that 'every variable whose value is known in x is known and has the same value in y'. Simply put: at least as much is known about y as about x. Note that $x \sqsubseteq y$ implies $I_x \subseteq I_y$, but not conversely. (For example, if $x = (0, 1, 1, \bot, \ldots)$ and $y = (0, 1, 0, 1, \bot, \ldots)$ then $I_x \subseteq I_y$ but $P_x \not\subseteq P_y$ so $x \not\sqsubseteq y$.)

2.1.3 Theorem *The orderings* $\sqsubseteq$ *and* $\leq$ *over* State *coincide.*

Proof: Take any $x, y \in \mathsf{State}$. Suppose $x \sqsubseteq y$. Then by **2.1.2** $P_x \subseteq P_y$ and $N_x \subseteq N_y$; so $\forall i > 0,\ x(v_i) = 0$ implies $y(v_i) = 0$; $x(v_i) = 1$ implies $y(v_i) = 1$ and $x(v_i) = \bot$ implies $y(v_i) \in \{\bot, 0, 1\}$. Therefore, for every $i > 0,\ x(v_i) \leq_{\mathsf{T}} y(v_i)$; so $x \leq y$.

For the converse, suppose $x \leq y$. Then for every $i > 0, x(v_i) \leq_{\mathsf{T}} y(v_i)$. Now $\forall i \geq 0,\ x(v_i) = 1$ implies $y(v_i) = 1$ since $1 \leq_{\mathsf{T}} 1$; so $P_x \subseteq P_y$. Also for every $i > 0, x(v_i) = 0$ implies $y(v_i) = 0$ since $0 \leq_{\mathsf{T}} 0$; so $N_x \subseteq N_y$. Hence $x \sqsubseteq y$.

Therefore, for all $x, y \in \mathsf{State}$, $x \sqsubseteq y$ iff $x \leq y$. □

The state space is (State, $\sqsubseteq$). I now examine its structure. There are many maximal elements, so (State, $\sqsubseteq$) is not a lattice, but rather:

2.1.4 Lemma *The ordered set* (State, $\sqsubseteq$) *is a cpo.*

Proof: This follows from the fact that T is a cpo, that State $= T^{\omega}$, and that the Cartesian product of cpo's is a cpo ([Plo78], p 210). □

The finite states are the *compact elements* of the cpo (State, $\sqsubseteq$). This fits the intuition (explained in Section 1.3) that compact elements of a domain may be thought of as information attainable in a finite amount of time.

2.1.5 Lemma *The compact elements of* (State, $\sqsubseteq$) *are exactly the finite states.*

Proof: Let $x \in \mathsf{State}^o$ be a finite state and D be a directed set such that $x \sqsubseteq \bigsqcup D$. Then for every $v \in P_x$ there is some $d \in D$ with $d(v) = 1$ (since $\bigsqcup D(v) = 1$ and $D \neq \emptyset$), and for every $v \in N_x$ there is some $e \in D$ with $e(v) = 0$. For each $v \in P_x$, select some $d \in D$ with $d(v) = 1$ and let P be the set of these d's. Similarly, for each $v \in N_x$, select some $d \in D$ with $d(v) = 0$ and let N be the set of these d's. The sets P and N are both finite sets (since P_x and N_x are respectively finite) and subsets of D. Since (State, $\sqsubseteq$) is a cpo, P has an upper bound say p, in D and N has an upper bound, say q, in D. Since D is directed, and $p, q \in D$ there is some $u \in D$ such that $p \sqsubseteq u$ and $q \sqsubseteq u$. Also $x \sqsubseteq u$, so x is compact.

Now to check that no other elements are compact, take any $x \in \mathsf{State}$ such that $\{v \mid x(v) \neq \bot\}$ is infinite. Then either one or both of P_x and N_x are infinite. Suppose P_x is infinite. Then P_x is countable, so its elements can be enumerated as, say, $P_x = \{v_{i_1}, v_{i_2}, \ldots, v_{i_n}, \ldots\}$. Let $X = \{x_1, x_2, \ldots, x_n, \ldots\}$ where $x_k \in \mathsf{State}$ with $P_{x_k} = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ and $N_{x_k} = N_x$. Now X is a directed set with $\bigsqcup X = x$. Hence $x \sqsubseteq \bigsqcup X$ and $w \sqsubseteq x$ for every $w \in X$; so x cannot be compact. □

The finite states form a countable basis for (State, $\sqsubseteq$), which means that (State, $\sqsubseteq$) is an ω-algebraic cpo and hence a domain. (It is called the *universal domain* [Plo78].)

2.1.6 Theorem *The cpo* (State, $\sqsubseteq$) *is a domain.*

Proof: First, for algebraicity I show that State^o is a basis for (State, $\sqsubseteq$). Consider any state $x \in \mathsf{State}$. Define $X = \{b \in \mathsf{State}^0 \mid b \sqsubseteq x\}$. The set X is directed. For take any $b_1, b_2 \in X$. Let $b \in \mathsf{State}$ be the state with $P_b = P_{b_1} \cup P_{b_2}$ and $N_b = N_{b_1} \cup N_{b_2}$. The state b is compact (since P_b and N_b are finite), and since $b_1 \sqsubseteq x$ and $b_2 \sqsubseteq x$, $b \sqsubseteq x$. Thus $b \in X$ and since $b_1 \sqsubseteq b$ and $b_2 \sqsubseteq b$, it follows that X is directed. Since (State, $\sqsubseteq$) is a cpo, X has a least upper bound, and $\bigsqcup X \sqsubseteq x$. Let $Y = \{y \in \mathsf{State} \mid y \sqsubseteq x\}$. Then $\bigsqcup Y = x$, so $x \sqsubseteq \bigsqcup X$ (since $X \subseteq Y$) and hence $x = \bigsqcup X$. Thus State^o is a basis and hence (State, $\sqsubseteq$) is algebraic.

Second, for ω-algebraicity I show that the basis State^o is countable. A one-one enumeration $b_0, b_1, \ldots$ of the compact elements is given (in [Plo78] p 213) by taking b_n to be the compact element x such that

$$n = \sum_{i \in I_P} 2 \cdot 3^i + \sum_{i \in I_N} 3^i$$

where $I_P = \{i \geq 0 \mid v_i \in P_x\}$ and $I_N = \{i \geq 0 \mid v_i \in N_x\}$.
Therefore, (State, $\sqsubseteq$) is an ω-algebraic cpo, and hence a domain. □

What role do the atomic states play? There are enough atomic states to distinguish between distinct states. The following sequence of lemmas shows how a state can be constructed abstractly from its atomic approximations.

A (finite) state x is called a (*finite*) *approximation* of a state y iff $x \sqsubseteq y$. The set of all states of which x is a (finite) approximation, denoted $\uparrow\{x\}$, is the set $\{y \in \mathsf{State} \mid x \sqsubseteq y\}$; this is an upper set (with respect to $\sqsubseteq$). The set-theoretical complement of $\uparrow\{x\}$, denoted $\overline{\uparrow\{x\}}$, is a lower set (with respect to $\sqsubseteq$). For any set $X \subseteq \mathsf{State}$, $\uparrow X$ is the set of states of which some $x \in X$ is an approximation (that is, $\uparrow X = \bigcup_{x \in X} \uparrow\{x\}$).

Any two atomic states (with distinct information contents) determine a state in State^o.

2.1.7 Theorem *For distinct $a, b \in \mathsf{Atom}$, if $I_a \neq I_b$ then there is some $x \in \mathsf{State}^o$ such that $\uparrow\{a\} \cap \uparrow\{b\} = \uparrow\{x\}$. If $I_a = I_b$ then $\uparrow\{a\} \cap \uparrow\{b\} = \emptyset$.*
Proof: Take any $a, b \in \mathsf{Atom}$. First, suppose $I_a \neq I_b$. Let $x \in \mathsf{State}^o$ with $P_x = P_a \cup P_b$ and $N_x = N_a \cup N_b$. Then $x \in \mathsf{State}^o$ since I_a and I_b are singletons. Now $y \in \uparrow\{a\} \cap \uparrow\{b\}$ iff $a \sqsubseteq y$ and $b \sqsubseteq y$ iff $P_a \cup P_b \subseteq P_y$ and $N_a \cup N_b \subseteq N_y$ iff $y \in \uparrow\{x\}$. Second, suppose $I_a = I_b$. Then either $P_a = N_b = \{v\}$ or $P_b = N_a = \{v\}$ for some $v \in \mathit{Var}$. In either case $\uparrow\{a\} \cap \uparrow\{b\} = \{y \mid a \sqsubseteq y \text{ and } b \sqsubseteq y\} = \{y \mid y(v) = 1 \text{ and } y(v) = 0\} = \emptyset$. □

In general, any set of atomic states (with distinct information contents) determines a state in State. That is, for any finite $A \subseteq \mathsf{Atom}$, if $I_a \neq I_b$ for every pair $a, b \in A$ then there is some $x \in \mathsf{State}^o$ such that $\bigcap_{a \in A} \uparrow\{a\} = \uparrow\{x\}$. Otherwise $\bigcap_{a \in A} \uparrow\{a\} = \emptyset$. For any non-finite $A \subseteq \mathsf{Atom}$, if $I_a \neq I_b$ for every pair $a, b \in A$ then there is some $x \in \mathsf{State} - \mathsf{State}^o$ such that $\bigcap_{a \in A} \uparrow\{a\} = \uparrow\{x\}$. Otherwise $\bigcap_{a \in A} \uparrow\{a\} = \emptyset$.

On the other hand, any (finite) state is (finitarily) determined by its atomic approximations in the sense that: for any $x \in \mathsf{State}$,

$$P_x = \bigcup\{P_a \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\} \quad \text{and} \quad N_x = \bigcup\{N_a \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}.$$

(If x is a finite state then these unions are finite.)

2.1.8 Theorem *For every $x \in \mathsf{State}$, $\uparrow\{x\} = \bigcap\{\uparrow\{a\} \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$.*
Proof: Take any $x \in \mathsf{State}$. Let $A = \{a \in \mathsf{Atom} \mid a \sqsubseteq x\}$. Suppose $y \in \uparrow\{x\}$. Then $x \sqsubseteq y$, but for every $a \in A$, $a \sqsubseteq x$ by transitivity of $\sqsubseteq$. Thus $y \in \bigcap\{\uparrow\{a\} \mid a \in A\}$. Conversely, suppose $y \in \bigcap\{\uparrow\{a\} \mid a \in A\}$. Then for every $a \in A$, $a \sqsubseteq y$ and in particular, $x \sqsubseteq y$ since $x \in A$. Thus $y \in \uparrow\{x\}$. Therefore $\uparrow\{x\} = \bigcap\{\uparrow\{a\} \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$. □

The role of the atomic and finite states of State will be formalised in Section 3.1 by the notion of Scott information system (Definition **1. 4. 1**).

2.2 A Priestley space of states

In this Section I topologise the domain (State, $\sqsubseteq$) to form a Priestley space. I define three topologies on (State, $\sqsubseteq$). Namely:

(a) a topology generated from a subbase of principal upper sets $\uparrow\{a\}$ ($a \in \mathsf{Atom}$);

(b) a topology generated from a subbase of (set-theoretic) complements of principal upper sets $\overline{\uparrow\{a\}}$ ($a \in \mathsf{Atom}$); and

(c) a topology generated from a subbase of the sets $\uparrow\{a\}$ and their complements $\overline{\uparrow\{a\}}$ ($a \in \mathsf{Atom}$).

Then I show that (State, $\sqsubseteq$) with the third topology forms a Priestley space. The first topology is the *Scott topology* on the domain (State, $\sqsubseteq$) ; the second is the *dual of the Scott topology*; and the third topology is the *patch topology* of the Scott topology and its dual. (More general definitions of these topologies can be found in (e.g) [Law87, Smy92].) For any domain with the Scott topology, the Lawson topology coincides with the patch of the Scott topology [GHK80] ([Smy92] Ex 7.7.8(8)). Therefore the construction used here is a special case of the general construction of a Priestley space from a domain using the Lawson topology given in (e.g.) [Pri85, Pri94a, Pri94b].

First, let $\mathcal{S}_u = \{\uparrow\{a\} \mid a \in \mathsf{Atom}\}$ and define $\mathcal{B}_u$ to be the collection of all finite intersections of members of $\mathcal{S}_u$ together with the set $\{(\bot, \bot, \ldots)\}$ — that is, $\mathcal{B}_u = \{\uparrow\{x\} \mid x \in \mathsf{State}^o\} \cup \{(\bot, \bot, \ldots)\}$ (by Lemma **2. 1. 7** and since $\{(\bot, \bot, \ldots)\} \cap \uparrow\{a\} = \emptyset$ for all $a \in \mathsf{Atom}$). Then $\mathcal{B}_u$ is closed under finite intersections, and hence:

2. 2. 1 Lemma *$\mathcal{B}_u$ is a base for a topology on* (State, $\sqsubseteq$) .
Proof: I check the two conditions for a set to be a base for a topology:

(i) For every $x \in \mathsf{State} - \{(\bot, \bot, \ldots)\}$, there is some $a \in \mathsf{Atom}$ such that $a \sqsubseteq x$; so $x \in \bigcup_{a \in \mathsf{Atom}} \uparrow\{a\}$; hence $\mathsf{State} - \{(\bot, \bot, \ldots)\} \subseteq \bigcup_{a \in \mathsf{Atom}} \uparrow\{a\}$. Trivially the reverse inclusion holds. Thus $\mathsf{State} - \{(\bot, \bot, \ldots)\} = \bigcup_{a \in \mathsf{Atom}} \uparrow\{a\}$, and hence $\mathsf{State} = \bigcup_{a \in \mathsf{Atom}} \uparrow\{a\} \cup \{(\bot, \bot, \ldots)\}$, that is, $\mathsf{State} = \bigcup_{b \in \mathcal{B}_u} b$.

(ii) Take any $B_1, B_2 \in \mathcal{B}_u$ with $x \in B_1 \cap B_2$. Without loss of generality I can assume $B_1 = \uparrow\{b_1\}$ and $B_2 = \uparrow\{b_2\}$ for some $b_1, b_2 \in \mathsf{Atom}$ (by

Theorem **2. 1. 7** and since $\uparrow\{a\} \cap \{(\bot, \bot, \ldots)\} = \emptyset$ for all $a \in \mathsf{Atom}$). Then $B_1 \cap B_2 \in \mathcal{B}_u$ (by Theorem **2. 1. 7**). Therefore, there is some $B_3 \in \mathcal{B}_u$, namely $B_1 \cap B_2$ such that $x \in B_3 \subseteq B_1 \cap B_2$.

By (i) and (ii), $\mathcal{B}_u$ is a base. □

The family $\mathcal{B}_u$ is not itself closed under unions, and hence is not itself a topology on $(\mathsf{State}, \sqsubseteq)$. (For example, let $a = (1, \bot, \ldots)$ and $b = (\bot, 1, \ldots)$. Then $\uparrow\{a\} \cup \uparrow\{b\} = \{x \mid x(v_1) = 1 \text{ or } x(v_2) = 1\} \neq \uparrow\{c\}$ for any $c \in \mathsf{State}^o$.) Let τ_u be the collection of all unions of members of $\mathcal{B}_u$ — that is,

$$\tau_u = \{\uparrow X \mid X \subseteq \mathsf{State}^o\} \cup \{\uparrow X \cup \{(\bot, \bot, \ldots)\} \mid X \subseteq \mathsf{State}^o\}$$

since by definition $\uparrow X = \bigcup_{x \in X} \uparrow\{x\}$. Then τ_u is a topology on $(\mathsf{State}, \sqsubseteq)$ which I call the *upper topology* since it is generated from upper sets. Elements of τ_u are open upper sets. By definition, τ_u is the Scott topology on $(\mathsf{State}, \sqsubseteq)$ and its specialisation order is $\sqsubseteq$.

Second, let $\mathcal{S}_l = \{\overline{\uparrow\{a\}} \mid a \in \mathsf{Atom}\}$ and define $\mathcal{B}_l$ to be the collection of all finite intersections of members of $\mathcal{S}_l$ — that is, $\mathcal{B}_l = \{\overline{\uparrow A} \mid \text{ finite } A \subseteq \mathsf{Atom}\}$ since for $A = \{a_1, \ldots a_n\} \subseteq \mathsf{Atom}$, $\overline{\uparrow\{a_1\}} \cap \ldots \cap \overline{\uparrow\{a_n\}} = \overline{\uparrow\{a_1\} \cup \ldots \cup \uparrow\{a_n\}} = \overline{\uparrow A}$. Then $\mathcal{B}_l$ is closed under finite intersections, and hence:

2. 2. 2 **Lemma** *$\mathcal{B}_l$ is a base for a topology on* $(\mathsf{State}, \sqsubseteq)$.

Proof: I check the two conditions for a set to be a base for a topology:

(i) For any $a, b \in \mathsf{Atom}$ with $I_a = I_b$ $(a \neq b)$, $\uparrow\{a\} \cap \uparrow\{b\} = \emptyset$; so $\mathsf{State} = \overline{\emptyset} = \overline{\uparrow\{a\} \cap \uparrow\{b\}} = \overline{\uparrow\{a\}} \cup \overline{\uparrow\{b\}}$. Hence $\mathsf{State} = \bigcup_{b \in \mathcal{B}_l} b$.

(ii) Take any $B_1, B_2 \in \mathcal{B}_l$ with $x \in B_1 \cap B_2$. Without loss of generality I can assume $B_1 = \overline{\uparrow\{b_1\}}$ and $B_2 = \overline{\uparrow\{b_2\}}$ for some $b_1, b_2 \in \mathsf{Atom}$ (since for all finite $A, B \subseteq \mathsf{Atom}$, $\overline{\uparrow A} \cap \overline{\uparrow B} = \overline{\uparrow\{a_1\}} \cap \ldots \cap \overline{\uparrow\{a_n\}}$ where $a_1, \ldots, a_n \in A \cup B$). Then $B_1 \cap B_2 \in \mathcal{B}_l$. Therefore, there is some $B_3 \in \mathcal{B}_l$, namely $B_1 \cap B_2$, such that $x \in B_3 \subseteq B_1 \cap B_2$.

By (i) and (ii), $\mathcal{B}_l$ is a base. □

The family $\mathcal{B}_l$ is not itself closed under unions, and hence is not itself a topology on $(\mathsf{State}, \sqsubseteq)$. (For example, let $a = (1, \bot, \ldots)$ and $b = (0, \bot, \ldots)$. Then $\overline{\uparrow\{a\}} \cup \overline{\uparrow\{b\}} = \{x \mid x(v_1) \neq 1 \text{ or } x(v_1) \neq 0\} \neq \overline{\uparrow A}$ for any finite $A \subseteq \mathsf{Atom}$.) Let τ_l be the collection of all unions of members of $\mathcal{B}_l$. Then τ_l is a topology on $(\mathsf{State}, \sqsubseteq)$ which I call the *lower topology* since it is generated from lower sets. Elements of τ_u are open lower sets.

Third, let $\mathcal{S} = \mathcal{S}_u \cup \mathcal{S}_l$, and define $\mathcal{B}$ to be the collection of finite intersections of sets of the form $\uparrow\{a\} \cap \overline{\uparrow\{b\}}$ $(a, b \in \mathsf{Atom})$ and all finite intersections of sets of the form $\overline{\uparrow\{b\}}$ $(b \in \mathsf{Atom})$. Then

$$\mathcal{B} = \{\overline{\uparrow B} \mid \text{finite } B \subseteq \mathsf{Atom}\} \cup \{\uparrow\{x\} \cap \overline{\uparrow B} \mid x \in \mathsf{State}^o, \text{ finite } B \subseteq \mathsf{Atom}\}$$

since: $(\uparrow\{a_1\} \cap \overline{\uparrow\{b_1\}}) \cap \ldots \cap (\uparrow\{a_n\} \cap \overline{\uparrow\{b_n\}})$
$= (\uparrow\{a_1\} \cap \ldots \cap \uparrow\{a_n\}) \cap (\overline{\uparrow\{b_1\}} \cap \ldots \cap \overline{\uparrow\{b_n\}})$
$= \uparrow\{x\} \cap \overline{(\uparrow\{b_1\} \cup \ldots \cup \uparrow\{b_n\})}$
$= \uparrow\{x\} \cap \overline{\uparrow B}$

where $B = \{b_1, \ldots, b_n\}$ is a finite subset of Atom and $x \in \mathsf{State}^o$. Also $\emptyset \in \mathcal{B}$ (since $\emptyset = \uparrow\{a\} \cap \overline{\uparrow\{a\}}$ for all $a \in$ Atom) but $\mathsf{State} \notin \mathcal{B}$.

2.2.3 Lemma *$\mathcal{B}$ is a base for a topology on* (State, $\sqsubseteq$) .
Proof: I check the two conditions for a set to be a base for a topology:

(i) By Lemma **2.2.2**, $\mathsf{State} = \bigcup_{b \in \mathsf{Atom}} \overline{\uparrow\{b\}}$; so $\mathsf{State} = \bigcup_{b \in \mathcal{B}} b$.

(ii) Take any $B_1, B_2 \in \mathcal{B}$ with $x \in B_1 \cap B_2$. Without loss of generality I need only consider three cases:

(a) Suppose $B_1 = \uparrow\{a\} \cap \overline{\uparrow\{b\}}$ and $B_2 = \overline{\uparrow\{c\}}$ $(a, b, c \in \mathsf{Atom})$. Then by Lemma **2.2.2**, $B_1 \cap B_2 \in \mathcal{B}$.

(b) Suppose $B_1 = \uparrow\{a_1\} \cap \overline{\uparrow\{b_1\}}$ and $B_2 = \uparrow\{a_2\} \cap \overline{\uparrow\{b_2\}}$ $(a_1, b_1, a_2, b_2 \in \mathsf{Atom})$. Then by Lemma **2.2.2**, $B_1 \cap B_2 \in \mathcal{B}$.

(c) Suppose $B_1 = \overline{\uparrow\{b_1\}}$ and $B_2 = \overline{\uparrow\{b_2\}}$ $(b_1, b_2 \in \mathsf{Atom})$. Then by Lemmas **2.2.2** and **2.2.1**, $B_1 \cap B_2 \in \mathcal{B}$.

In all three cases, there is some $B_3 \in \mathcal{B}$ such that $x \in B_3 \subseteq B_1 \cap B_2$.

By (i) and (ii), $\mathcal{B}$ is a base. □

The family $\mathcal{B}$ is not itself closed under unions (since $\mathcal{B}_u$ and $\mathcal{B}_l$ are not), and hence is not itself a topology in (State, $\sqsubseteq$) . Let τ_{State} be the collection of all unions of members of $\mathcal{B}$. Then $\emptyset \in \tau_{\mathsf{State}}$ (since $\emptyset = \uparrow\{a\} \cap \overline{\uparrow\{a\}}$ for any $a \in$ Atom) and $\mathsf{State} \in \tau_{\mathsf{State}}$ (since $\mathsf{State} = \overline{\uparrow\{a\}} \cup \overline{\uparrow\{b\}}$ where $a, b \in$ Atom with $I_a = I_b$ $(a \neq b)$). By definition, τ_{State} is closed under finite intersections and arbitrary unions, and therefore τ_{State} is a topology on (State, $\sqsubseteq$) . This is the patch topology [Smy92] of the upper and lower topologies.

For each $a \in \mathsf{State}^o$, $\uparrow\{a\}$ and $\overline{\uparrow\{a\}}$ are respectively open upper and open lower sets in (State, $\sqsubseteq, \tau_{\mathsf{State}}$) . These sets are also set-theoretic complements (in State) of each other, so $\uparrow\{a\}$ is a clopen upper set and $\overline{\uparrow\{a\}}$ is a clopen lower set. Since $\mathcal{B}$ contains all finite intersections of members of $\mathcal{S}$, and a finite intersection of clopen sets is a clopen set, the sets in $\mathcal{B}$ are clopen sets. Therefore (State, $\sqsubseteq, \tau_{\mathsf{State}}$) has a subbase of clopen upper sets and clopen lower sets, and a base of clopen sets.

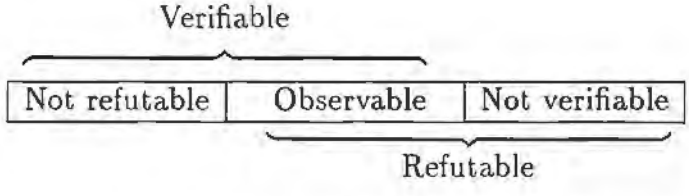
2.2.4 Theorem *The space* (State, $\sqsubseteq, \tau_{\mathsf{State}}$) *is a Priestley space.*
Proof: By Lemma **2.1.4**, (State, $\sqsubseteq$) is a partially ordered set. To prove compactness I use the Alexander subbasis theorem. That is, I need to show that every subbasic open cover of State has a finite subcover. Suppose (State, $\sqsubseteq, \tau_{\mathsf{State}}$) is not compact. Let $\mathcal{U} = \{\uparrow\{a\} \mid a \in A\} \cup \{\overline{\uparrow\{b\}} \mid$

$b \in B\}$ $(A, B \subseteq \mathsf{Atom})$ be a subbasic open cover of State with no finite subcover. Let $X_B = \uparrow B$ $(= \bigcup_{b\in B} \uparrow\{b\})$. Then $X_B \cap \bigcap_{b\in B} \uparrow\{b\} \neq \emptyset$ (since $\bigcap_{b\in B} \uparrow\{b\} \subseteq \bigcup_{b\in B} \uparrow\{b\}$) Thus $X_B \not\subseteq \bigcup_{b\in B} \uparrow\{b\}$. Also $X_B \not\subseteq \bigcup_{a\in A} \uparrow\{a\}$. This follows from the fact that $X_B \cap A = \emptyset$. To show the latter suppose $a \in X_B$ for some $a \in A$. Then $b \sqsubseteq a$ for some $b \in B$, so $a = b$ since $a, b \in \mathsf{Atom}$ and $(\bot, \bot, \ldots) \notin \mathsf{Atom}$. Then $\uparrow\{a\} \cup \overline{\uparrow\{b\}} = \mathsf{State}$, that is, $\{\uparrow\{a\}, \overline{\uparrow\{b\}}\}$ is a finite subcover. But $\mathcal{U}$ has no finite subcover; so I obtain a contradiction to the assumption that $X_B \cap A \neq \emptyset$. Thus $X_B \cap A = \emptyset$ which implies $B \cap A = \emptyset$ (since $A, B \subseteq \mathsf{Atom}$) and hence $X_B \not\subseteq \bigcup_{a\in A} \uparrow\{a\}$, and so I obtain a contradiction to the assumption that $\mathcal{U}$ is an open cover of State. Hence $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ is compact.
For total order-disconnectedness, take any $x, y \in \mathsf{State}$ with $x \not\sqsubseteq y$. Then $P_x \not\subseteq P_y$ or $N_x \not\subseteq N_y$. I f $P_x \not\subseteq P_y$ then there is some $v \in P_x - P_y$, so there is some $a \in \mathsf{Atom}$ with $P_a \subseteq P_x - P_y$. For such an atomic state, $x \in \uparrow\{a\}$ and $y \in \overline{\uparrow\{a\}}$. Similarly, if $N_x \not\subseteq N_y$ then there is some $v \in N_x - N_y$, so there is some $b \in \mathsf{Atom}$ with $N_b \subseteq N_x - N_y$. For such an atomic state, $x \in \uparrow\{b\}$ and $y \in \overline{\uparrow\{b\}}$. Thus, for any $x, y \in \mathsf{State}$ with $x \not\sqsubseteq y$ there is some clopen upper set $\uparrow\{a\}$ and some clopen lower set $\overline{\uparrow\{a\}}$ $(a \in \mathsf{Atom})$ such that $x \in \uparrow\{a\}$ and $y \in \overline{\uparrow\{a\}}$. Therefore $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ is totally order-disconnected. This completes the proof that $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ is a Priestley space. □

This completes our example of a Priestley space that can be viewed as a state space. The next step is to consider whether every Priestley space can be viewed as a state space.

2.3 Open sets as verifiable properties

In this Section I introduce notions of *verifiable properties*, *refutable properties* and *observable properties* of states. These will stand to each other in the following relationship:



I link up the notions of verifiability, refutability and observability with topological notions of open, closed and clopen sets — specifically from a Priestley space.

The idea that 'open sets are ... properties' was first put forward by Smyth [Smy83], but there appears to be no generally accepted standard way of doing so, nor any accepted terminology. For example, Smyth [Smy83] speaks of computable property and semi-decidable property, Vickers [Vic89] speaks of affirmative assertion, Abramsky [Abr91] and Smyth [Smy92] speak of finitely observable property, and Alpern and Schneider [AlS85] and Kwiatkowska [Kwi91] speak of (finitary) liveness properties. An account of the variations is given in [Smy92]. At the end of this section I compare the notions introduced here with those of Smyth and in Section 2.4 a comparison will be made with the Alpern, Schneider and Kwiatkowska notions.

For an informal introduction to the notions of verifiability, refutability and observability consider the state space $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ defined in Sections 2.1 and 2.2. (Cf also the example of Smyth [Smy92].) A property of a state in State is taken to be a description of the values of program variables and the relationships between them. It will be convenient to identify a property with the set of states in which it is true — that is, properties are subsets of State. This allows us to think of the classical logical connectives as the classical set-theoretic operations. The behaviour of a program can be determined by the properties detected by an observer of the input and output states. An observer, being human, can only make judgements in a finite amount of time and based on a finite amount of information. In other words, given a property and a state belonging to it, an observer can only establish this fact by inspecting some finite number of variables. If the property in question describes specific variables or refers to a specific range of variables the observer will know in advance something more, namely which variables to consider. Similarly for a state not belonging to a given property.

With these ideas in mind I say a property of states is:

- *verifiable* if whenever it is present this fact can be established by inspection of some finite number of variables;
- *refutable* if whenever it is absent this fact can be established by inspection of some finite number of variables; and
- *observable* if it is both verifiable and refutable.

It should be noted that observability is a stronger notion than verifiability and than refutability in the sense that an observer knows in advance exactly which variables or what range of variables must be inspected to establish whether the state under consideration has the property in question.

There are some properties for which it is not always the case that their presence can be established by inspection of some finite number of variables. For example, that infinitely many variables have value 0 or infinitely many variables are known. Such properties are simply *not verifiable*. Likewise, there are some

properties for which it is not always the case that their absence can be established by inspection of some finite number of variables. For example, that two successive variables have value 0. Such properties are simply *not refutable*. Then any property which is either not verifiable or not refutable is *not observable*.

For examples of these notions I consider properties of states in State. I present these properties in a table to reflect the relationships between them.

Verifiable not refutable	Observable	Refutable not verifiable
some variable v has value 1	(fixed) variable v has value 1	every variable v has value 1
some variable v is known	(fixed) variable v is known	every variable v is known
at least three distinct variables have value 1	at least the first three variables have value 1	at most three distinct variables have value 1
some variables v_i and v_{i+1} have different values	variables v_i and v_{i+1} have different values	all variables v_i and v_{i+1} have different values

It is an elementary exercise to check that the properties are indeed examples of the notions as indicated. For example, suppose a state x has the property 'every variable v has value 1'. This property is refutable since to establish its absence from a state, at least one variable with value not equal to one must be found. Once this variable has been found only finitely many variables will have been inspected; so the property is refutable. It is not verifiable since to establish its presence in a state denumerably many variables must be inspected. Suppose a state x has the property 'at least three distinct variables have value 1'. Once three distinct variables have been found with value 1, only finitely many variables will have been inspected; so the property is verifiable. It is not refutable since to establish its absence from a state it must be checked that at most two variables have value 1, and hence the value of every variable would need to be inspected.

The form of these examples suggests the existence of some basic observable properties from which other properties can be formed using (logical) conjunctions (that is, universal quantification) and disjunctions (that is, existential quantification).

For conjunctions, suppose P is an infinite conjunction of observable properties. If a state x has property P then every conjunct in P is true in x. So establishing the presence of P in x would involve establishing the presence of every single conjunct in x. For each conjunct we need only check a finite number of variables, but repeating this for each conjunct in P would involve inspecting an infinite number of variables. In practice this can never be done. Hence property P is not verifiable. (Note that if P is a finite conjunction of observable properties, then the presence can be established and hence P is verifiable.) If a state x does not have the property P then at least one conjunct in P does not hold in x. Once this conjunct has been found, we will have considered a finite number of conjuncts in P, and for each of these conjuncts we will have inspected a finite

number of variables. Therefore the absence of P from x can be established by inspecting a finite number of variables, and hence P is a refutable property. Thus:

- *Any finite conjunction of observable properties is an observable property.*
- *An infinite conjunction of observable properties is a refutable but not observable property (i.e. it is not verifiable).*

Similar arguments for disjunctions of observable properties, establish that:

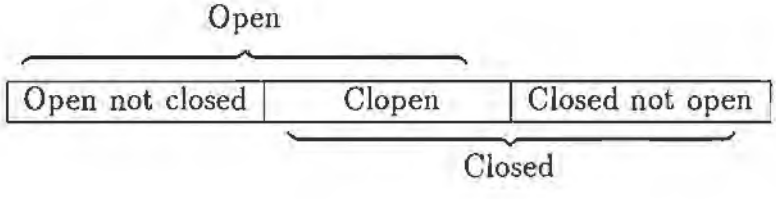
- *Any finite disjunction of observable properties is an observable property.*
- *An infinite disjunction of observable properties is a verifiable but not observable property (i.e. it is not refutable).*

This means that the collection of verifiable properties is closed under finite conjunctions and arbitrary disjunctions, while the collection of refutable properties is closed under finite disjunctions and arbitrary conjunctions.

The relationship between verifiable and refutable properties can be described in terms of logical negation. In particular,

- *The negation of a verifiable property is a refutable property, and vice versa.*
- *The negation of a verifiable but not observable property is a refutable but not observable property, and vice versa.*
- *The negation of an observable property is an observable property.*

Compare the above closure conditions with those of open, closed and clopen subsets of a topological space. Then identifying properties of states with sets of states from a Priestley space, we have that the trichotomy of properties illustrated above is mirrored exactly by and can be viewed as an interpretation of the trichotomy:



The observable properties constitute a base for the topology of verifiable properties. Therefore to establish that the collection of verifiable properties is the whole topology, it suffices to characterise the observable properties, as done for example in the case study $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$.

In conclusion, I compare these interpretations of open, closed and clopen sets with those given in Smyth [Smy92]. Smyth introduces the notion of *finitely observable* property ([Smy92], p 713) to describe open sets.

On my understanding, a property is *Smyth finitely observable* ([Smy92], p 642) if whenever the property is present in a state this fact can be established by considering some finite approximation of the state under consideration. If it is known in advance, where the finite approximation can be found, then the property is called *Smyth boundedly observable* ([Smy92], p 713). Comparing these notions with the notions of verifiability, refutability and observability, we see that the only difference lies in the formulation, not in the intuition. Specifically, for any state, any finite approximation of it describes some finite information about it in terms of a finite number of variables. Therefore, the phrase 'finite approximation' is another way of saying a 'finite number of variables', and hence:

- *A property is verifiable iff it is Smyth finitely observable.*
- *A property is observable iff it is Smyth boundedly observable.*

My interpretation of the topology of a Priestley space coincides with Smyth's interpretation of a topological space with a clopen base since every Smyth finitely observable property is a disjunction of boundedly observable properties ([Smy92], p 713).

2.4 Upper sets as positive properties

In this Section I interpret upper and lower sets as *positive* and *negative* properties of states. I introduce these interpretations by considering the distinguishing features of upper and lower sets of the Priestley space $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ and of Priestley spaces, in general. Combining these interpretations with those of Section 3, I show that the upper topology of a Priestley space presents all positive properties about states and the lower topology presents all negative properties. Based on these interpretations of the subsets of a Priestley space, I put forward the idea that a Priestley space can be viewed as a state space.

For an informal introduction to the notions of positive and negative properties, consider the state space $(\mathsf{State}, \sqsubseteq)$. A state $x \in \mathsf{State}$ encodes a certain amount of information about program variables, namely the values of variables known in x. This information is present in those states which encode at least the

information encoded by x, but it is not present in those states which encode less information than x and/or different information from x. Therefore, by Definition **2.1.2** of the ordering $\sqsubseteq$, the upper set $\uparrow\{x\}$ is the set of states in which the information encoded by x is definitely present; while the lower set $\overline{\uparrow\{x\}}$ is the set of states in which the information encoded by x is definitely not present. This idea can be extended to sets of states. For any set $X \subseteq \mathsf{State}$, $\uparrow X = \bigcup_{x \in X} \uparrow\{x\}$. Therefore, $\uparrow X$ is the set of all states in which the information encoded by some state in X is definitely present. Dually, the lower set $\overline{\uparrow X}$ is the set of all states in which the information encoded by every state in X is definitely not present.

In general, the distinguishing feature of upper sets (as properties of states) is that they are *information-preserving*, because if a state has such a property, then so does any state with more information. In addition there is the idea that a certain amount of information is definitely present. I call a property with these features, a *positive property*. On the other hand, lower sets have the distinguishing feature that they preserve *lack of information*, because if a state has such a property, then so does any state with less information. In addition there is the idea that a certain amount of information is definitely not present. I call a property with these features, a *negative property*.

Compare this explanation with that of Smyth ([Smy92] p 733). According to Smyth, a positive property has the feature that, once it is true in a state then it is true in all states with more information. On the other hand, a negative property is such that once it is false in a state then it is false in all states with less information. This is another way of saying that upper sets can be interpreted as positive properties, and lower sets as negative properties.

Note that I am using 'positive' in two different ways in this thesis: 'positive' information content, as in a state (Section 2.1), and 'positive' property, as a set of states (here). In Section 2.1 I viewed the set $\{v \mid s(v) = 1\}$ as conveying positive information about the state s and the set $\{v \mid s(v) = 0\}$ as conveying negative information. Here I equate information with positive and negative information, and contrast the presence of information with the lack of information. I leave open the question of whether these notions have a connection with the notion of *positive formula* in propositional logic. (Recall (from (e.g.) [Kei77]) that a propositional formula is *positive* if it can be constructed from unnegated propositional variables using only conjunction and disjunction.) The representation, in Section 3.1, of states in State as propositional *theories* suggests that the use of 'positive' in Section 2.1 is consistent with the classical one.

Combine the interpretations of upper and lower sets with those of open, closed and clopen sets from Section 2.3. Then the upper topology of a Priestley space has a base of clopen upper sets of the Priestley space (Section 1.2) and therefore presents positive properties about states. In particular, open, closed and clopen upper sets can be interpreted as verifiable, refutable and observable positive properties. Dually, the lower topology of a Priestley space has a base of

complements of clopen upper sets of the Priestely space (Section 1.2) and therefore presents negative properties about states, in particular, the negations of the properties presented by the upper topology. That is, open, closed and clopen lower sets can be interpreted as verifiable, refutable and observable negative properties.

Compare the notions of verifiable positive property and refutable positive property with the notions of (finitary) liveness property (in [AlS85, Kwi91]) and safety property (in [AlS85, Kwi91, Smy92]). On my understanding, a *safety property* describes a finite observable property that must not occur, and a *finitary liveness property* describes a finite observable property that must occur. The phrase 'must occur' is another way of saying 'is definitely present', so

- *A property is a verifiable positive property iff it is a finitary liveness property.*

Since the negation of a verifiable (finitely observable property) is a refutable property,

- *A property is a refutable positive property iff it is a safety property.*

There is no comparable notion here for a liveness property. Note further that in ([AlS85], p 181) ([Kwi91], p 168) and ([Smy92], p 653) a safety property is defined to be a (Scott) closed set, and in ([Kwi91], p 168) a finitary liveness property is a (Scott) open set and a liveness property is a (Scott) G_δ-set (that is, a countable intersection of open sets). Therefore in the case of a Priestley space (such as $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$), where the upper topology is the Scott topology (that is, where the open upper sets are Scott open sets and the closed upper sets are the Scott closed sets) our interpretation coincides with that of [AlS85, Kwi91].

Examples of positive and negative properties of states in $(\mathsf{State}, \sqsubseteq)$ are listed in the following table.

Positive	Negative
at least variable v has value 1	at least variable v does not have value 1
at least variable v is known	at least variable v is not known
at least three distinct variables all have value 1	at most two distinct variables have value 1
at least variables v_i and v_{i+1} for some i have value 1	no two successive variables v_i and v_{i+1} both have value 1
every variable is known	at least one variable is unknown
all variables v_i have value 1	at least some variable v_i does not have value 1

It can easily be checked that these properties are indeed positive or negative. For example, the property 'at least variable v does not have value 1' is negative because it can be represented as the complement of the upclosure of the atomic

state in which variable v has value 1. The property 'every variable is known' is positive because it can be represented as the union of the upclosures of the maximal states in State. This is simply the set of maximal states, and hence an upper set. (The definition of a maximal state is in Section 2.1.)

The results of Sections 2.3 and 2.4 can be summarised as follows. A Priestley space $(X, \leq, \tau)$ can be viewed as a state space consisting of a collection X of states together with an order $\leq$ placing states in a hierarchy of increasing information and a collection τ of verifiable properties of states built up from basic observable properties. (In general, there is no special state of nontermination. As will be discussed in Section 2.5, nontermination can be captured using closed upper subsets of a Priestley space.) Taking the state space to be a Priestley space allows distinct incomparable states to be distinguished by a certain amount of information which is definitely present in the one and definitely not present in the other. (This is the total order-disconnectedness condition.) Compactness says that only a finite amount of information is required to make such distinctions. Taking the state space to be a Priestley space $(X, \leq, \tau)$ provides a distinction between two kinds of properties: a *topological kind* (that is, verifiable, refutable and observable properties) and an *order-theoretic kind* (that is, positive and negative properties). Properties of the topological kind arise from the topology τ, and are formed from finite observations using logical conjunctions, disjunctions and negation. This suggests that these properties form a *logic of finite observations* and that the topology τ is a model for this logic. (Abramsky [Abr91] coined the phrase 'logic of finite observations'; the idea that a topology is a model for a 'logic of affirmative (or finitely observable) assertions' is due to Vickers [Vic89] and has also been investigated by Smyth [Smy92].) Properties of the order-theoretic kind arise from the order $\leq$ and hence deal with *information*. Therefore, a Priestley space provides a good model for separating properties dealing with the idea of finite observation from properties dealing with information, that is, for separating the logical and the information-theoretic aspects of a state space. In the terminology of Vickers [Vic89], a Priestley space separates 'a theory of information' from 'a logic of finite observations'.

2.5 Properties describing sets of final states of programs

Having established that a Priestley space can be viewed as a state space, the next step is to find a suitable relational representation of programs over Priestley spaces. In this Section I show how the topological characterisation of the set of final states of a program depends on its nature: its nondeterministic and its terminating behaviour. I relate these topological characterisations to those of the

properties of states (defined in Sections 2.3 and 2.4) and thereby determine the kinds of properties that can be used to describe the output of programs. These properties will be used in Section 2.6 to describe the input-output behaviour of programs in terms of mappings between properties of states. These mappings, when viewed as power operations of binary relations (in the sense of Section 1.2), give rise to a natural relational representation of programs.

I require a slightly extended view of initial and final states of programs than that given in Section 1.1. A *final state* for a program α is taken to be any state in which *at least* the variables manipulated in the program are known with values as produced by the program; the states with more information than produced by the program can be viewed as final states for the program when executed as part of a larger program — the other values being the values of the global variables of the larger program, not changed in α. An *initial state* for a program α is taken to be any state in which *at least* the variables used but not initialised in the program are known; the states with more information than required by the program can be viewed as initial states for the program when executed as part of a larger program — the other values being the values of the global variables of the larger program not used in α. I refer to a final state t produced by a program from a given initial state s as an *actual final state* if t does not contain more information than can be produced by the program from the information provided by s. Recall (from Chapter 1) that a program is nondeterministic from a given initial state if more than one final state is possible from that state. Therefore, the set of final states of a program from an initial state is identified with the upper set generated by the set of actual final states of the program from that initial state. I will denote the set of final states of a program α from a state s by $\alpha(s)$.

In line with the approach of the previous sections I start with an informal introduction to the topological characterisations of sets of final states in $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ of programs. Then $\alpha(s) = \uparrow T$ where $T \subseteq \mathsf{State}$ is the set of actual final states of α from s. In particular, if α is finitely nondeterministic from s then T is a finite subset of State; if α is unboundedly nondeterministic from s then T is an infinite subset of State; if α is terminating from s then $T \subseteq \mathsf{State}^0$; if α is nonterminating from s then $T \not\subseteq \mathsf{State}^0$; and if α aborts from s then $\alpha(s) = \mathsf{State}$ (since $(\bot, \bot, \ldots) \in \alpha(s)$).

I now determine the sets of final states in $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ for programs written in Dijkstra's guarded command language. (For the syntax of this language refer to Section 1.2.)

2.5.1 Example Consider a finitely nondeterministic, terminating program α,

$$\alpha : \text{if } v_1 = v_2 \rightarrow \text{ skip } [] \ (v_1 \neq 0 \wedge v_2 \neq 0) \rightarrow v_1 = 0 \text{ fi}$$

The program α always terminates cleanly from states in which (at

least) both v_1 and v_2 are defined and (at least) one of the guards is true — namely from the states $s_1 = (0, 0, \perp, \ldots), s_2 = (1, 1, \perp, \ldots)$, and all their extensions. From s_1 the final state is $(0, 0, \perp, \ldots)$ so $\alpha(s_1) = \uparrow\{(0, 0, \perp, \ldots)\}$, and from s_2 two final states are possible, namely $(1, 1, \perp, \ldots)$ and $(0, 1, \perp, \ldots)$, so $\alpha(s_2) = \uparrow\{(1, 1, \perp, \ldots), (0, 1, \perp, \ldots)\}$. By definition of the IF command, α aborts from states in which both the guards are false — that is, from the states $s_3 = (0, 1, \perp, \ldots), s_4 = (1, 0, \perp, \ldots)$, and all their extensions $\alpha(s_3) = \uparrow\{(\perp, \perp, \ldots)\}$ and $\alpha(s_4) = \uparrow\{(\perp, \perp, \ldots)\}$. Now consider states in which the values of v_1 and/or v_2 are unknown — namely, the states $s_5 = (\perp, 0, \ldots), s_6 = (\perp, 1, \ldots), s_7 = (0, \perp, \ldots), s_8 = (1, \perp, \ldots)$ and all their extensions. From these states, the final state of α is $(\perp, \perp, \ldots)$, so $\alpha(s_i) = \uparrow\{(\perp, \perp, \ldots)\} = \mathsf{State}$ (for $i = 5, 6, 7, 8$). Similarly, for the sets of final states of α executed from extensions of the initial states $s_1, \ldots, s_8$. Then $\alpha(s_i)$ is a clopen upper set for each $i = 1, 2, \ldots, 8$.

So a possible topological characterisation of the set of final states of a program α is:

$$\forall s \in \mathsf{State} \;\; \alpha(s) \text{ is a clopen upper set.}$$

At the end of this section, I shall show that this characterisation suffices for finitely nondeterministic, terminating programs. However, as the next example shows, it does not cover programs which can produce a set of incomparable final states or a directed set of final states from at least one initial state.

2.5.2 Example Consider the unboundedly nondeterministic program μ which assigns to some variable v_i the value 1 and to all v_j with $j < i$ the value 0.

μ : go-on := true; $i := 1$;
do go-on $\rightarrow i := i + 1$ [] go-on $\rightarrow$ false od
$v_i := 1; i := i - 1$
do $i > 0 \rightarrow v_i := 0; i := i - 1$ od

The program μ always terminates from any state. The set $\mu(s)$ of final states of μ from any state s is

$$\uparrow(1, \perp, \ldots) \cup \uparrow(0, 1, \perp, \ldots) \cup \uparrow(0, 0, 1, \perp, \ldots) \cup \ldots \cup \uparrow(0, \ldots, 0, 1, \perp, \ldots) \cup \ldots.$$

The states in any two of the clopen upper sets in this union are incomparable, so this union cannot be simplified any further, and hence $\mu(s)$ is an open upper set.

A weaker characterisation of the set of final states of a program α is:

$$\forall s \in \mathsf{State} \;\; \alpha(s) \text{ is an open upper set.}$$

At the end of this section, I shall show that this characterisation can indeed be used for unboundedly nondeterministic programs. However the next example shows that it does not cover nonterminating programs.

2.5.3 Example Consider the deterministic program β

$$\beta : i := 1; \text{do } i > 0 \to (v_i := 1; v_{i+1} := 0; i := i + 2) \text{ od}$$

Any state is a possible initial state for β, but β is guaranteed never to terminate. If given enough time β would terminate with an infinite alternating sequence of 0's and 1's. If we interrupt execution of β from state $(\perp, \perp, \ldots)$ after (say) n iterations of the loop we will obtain an approximation to this sequence — a sequence

$$t_n = (1, 0, 1, 0, \ldots, 1, 0, \perp, \ldots)$$

where the first $n + 1$ entries are arranged in an alternating sequence of 0's and 1's, and $v_{n+2}, v_{n+3}, \ldots$ are all unknown. The set $\uparrow\{t_n\}$ then includes the approximations obtained when β is started from any other state and is interrupted after n iterations of the loop. But $\bigcap_{n>0} \uparrow\{t_n\}$ is the state representing the required infinite sequence of 0's and 1's. That is, for every state s, $\beta(s) = \bigcap_{n>0} \uparrow\{t_n\}$ which is a closed upper set.

An alternative characterisation of the set of final states of a program α is:

$$\forall s \in \mathsf{State} \;\; \alpha(s) \text{ is a closed upper set.}$$

These examples suggest that the topological structure of the set of final states of a program from a state $s \in \mathsf{State}$ depends not only on the nondeterminism of the program but also on its terminating behaviour.

In general, suppose the state space is a Priestley space $(X, \leq, \tau)$. Take a program α and a state s describing the least amount of information required by the program.

Suppose α is finitely nondeterministic and terminating from s. Then there are finitely many final states of α from s, and in each of these the values of finitely many program variables are known. Checking that a state t is a final state of α from s, involves finding a state in $\alpha(s)$ in which the values of variables are the same as in t. Once this state has been found, a finite number of states, and hence a finite number of variables, will have been inspected. Checking that a state is not a final state of α from s involves finding a state in $\alpha(s)$ which differs from t in the value of at least one variable. Hence the output of α from s can be described by an observable property.

Suppose α is an unboundedly nondeterministic and terminating program from s. Then there are infinitely many final states of α from s, and in each the value

of finitely many program variables are known. Checking that a state t is one of these final states involves finding the state in $\alpha(s)$ in which the values of variables are the same as in t. Once this state has been found, finitely many states, and hence finitely many variables, will have been inspected. However checking that a state t is not one of the final states of α from s would involve finding for each state in $\alpha(s)$ a variable with a value different from that in t. In practice this can never be done. Hence the output of an unboundedly nondeterministic and terminating program can be described by a verifiable property.

Suppose α does not terminate from s. Then either α aborts or loops or if it were given infinite time it would reach a state describing an infinite amount of information about program variables. In the first and second cases there are no final states of α from s, so the 'output' of α can be described by an observable property. In the third case checking that a state is not the 'final' state of the program involves finding at least one variable with a different value from that in the 'final' state. Once this variable has been found finitely many variables will have been inspected. However checking that a state is the 'final' state of α from s would require checking that the value of every single variable is the same as in the 'final' state. In practice this can never be done. So the output of a program which may possibly not terminate from a state can be described by a refutable property.

Taking the extended view of final and initial states described at the start of this Section, we have that for a program α and a state s,

- if α is finitely nondeterministic and terminating from s then the output of α from s can be described by an observable positive property;
- if α is unboundedly nondeterministic and terminating from s then the output of α from s can be described by a verifiable positive property; and
- if α is nonterminating from s then the output of α from s can be described by a refutable positive property;

and in terms of open, closed and clopen subsets of the Priestley space $(X, \leq, \tau)$,

- if α is finitely nondeterministic and terminating from s then $\alpha(s)$ is a clopen upper set;
- if α is unboundedly nondeterministic and terminating from s then $\alpha(s)$ is an open upper set;
- if α is nonterminating from s then $\alpha(s)$ is a closed upper set; and
- if α aborts from s then $\alpha(s)$ is the clopen upper set X (if X has a bottom state) or $\emptyset$.

Note that these characterisations are relative to a state s. I call a program *finitely nondeterministic* if it is finitely nondeterministic from every state, *terminating* if it is terminating from every state, *unboundedly nondeterministic* if it is unboundedly nondeterministic from some state, and nonterminating if it is nonterminating from some state. Then the output of finitely nondeterministic, terminating programs can be described by observable positive properties, the output of unboundedly nondeterministic, terminating programs can be described by verifiable positive properties, and the output of nonterminating programs by refutable positive properties. Note that I am assuming that the behaviour of a program is known in advance since determining this is an undecidable question.

2.6 Programs as mappings between observable positive properties

In this Section I introduce three notions of predicate transformers: *verifiable*, *refutable* and *observable predicate transformers*. I link up the representations of programs given here with Dijkstra's predicate transformers [Dij75, Dij76] outlined in Section 1.2. Applying the view of Section 1.2, that predicate transformers are power operations of programs-as-relations, I obtain a natural relational representation of programs over Priestley spaces. Then invoking the techniques of Priestley duality (of Sections 1.5 and 1.6) I derive a corresponding algebraic presentation of program semantics.

Suppose the state space is a Priestley space $(X, \leq, \tau)$. Let $\mathcal{U}X$ denote the clopen upper subsets of $(X, \leq, \tau)$, $\mathcal{O}X$ the open upper subsets, and $\mathcal{C}X$ the closed upper subsets. I define three notions of predicate transformer as follows:

2.6.1 Definition For a program α,

(a) a *verifiable predicate transformer* is a mapping $vp(\alpha, -) : \mathcal{O}X \to \mathcal{O}X$ between open upper sets;

(b) a *refutable predicate transformer* is a mapping $rp(\alpha, -) : \mathcal{C}X \to \mathcal{C}X$ between closed upper sets; and

(ca) an *observable predicate transformer* is a mapping $op(\alpha, -) : \mathcal{U}X \to \mathcal{U}X$ between clopen upper sets.

Properties of these predicate transformers arise from the closure conditions of the sets on which they are defined, and from the terminating and nondeterministic behaviour captured by these sets (viewed as properties of states). I use the terminology (for predicate transformers) introduced in Section 1.2 to describe these properties.

By definition, a verifiable predicate transformer maps open sets to open sets. (Recall open sets are closed under finite intersections.) Therefore its value on a finite intersection of sets can be obtained by taking the intersection of the images. This means that $vp(\alpha, -)$ is *conjunctive*. Recall (from Sections 2.3 and 2.4) that open upper sets can be interpreted as verifiable positive properties. Based on the terminating and nondeterministic behaviour of programs captured by these properties, the preconditions $vp(\alpha, Q)$ allow for the possibility of unbounded nondeterminism but do not allow nontermination. The termination requirement means that $vp(\alpha, -)$ is *strict*. Hence this predicate transformer is useful for the semantic characterisation of all terminating programs including unboundedly nondeterministic ones, and is therefore consistent (in the sense of Section 1.2) with Dijkstra and Scholten's [DiS90] weakest precondition predicate transformer.

A refutable predicate transformer maps closed sets to closed sets. (Recall closed sets are closed under arbitrary intersections.) Therefore its value on an arbitrary intersection of sets can be obtained by taking the intersection of the images. This means that $rp(\alpha, -)$ is *completely conjunctive*. Recall closed upper sets can be interpreted as refutable positive properties, and such properties can be used to describe nonterminating behaviour of programs. Therefore the preconditions $rp(\alpha, Q)$ allow for the possibility of nontermination. Hence for a program α, the predicate transformer $rp(\alpha, -)$ maps a given postcondition Q describing the desired output of α to the set of all states from which whenever α terminates it does so in a state in which Q is true. Therefore, refutable predicate transformers are consistent with Dijkstra's [Dij76, DiS90] weakest liberal precondition predicate transformers.

Observable predicate transformers map open sets (which are also closed) to open sets (which are also closed), and are therefore *conjunctive*. Clopen upper sets can be interpreted as observable positive properties, and such properties can be used to describe the finitely nondeterministic and terminating behaviour of programs. This means that $op(\alpha, -)$ is *continuous* (in the sense of Section 1.2) and *strict*. Hence for a program α, the predicate transformer $op(\alpha, -)$ maps a given postcondition Q about the desired output of α to the set of all states from which α is guaranteed to terminate in a state in which Q is true. In other words, observable predicate transformers are consistent with Dijkstra's [Dij75, Dij76] weakest precondition predicate transformers.

I now formalise the relationships between Dijkstra's predicate transformers and observable, verifiable and refutable predicate transformers. Recall (from Section 1.2) that Dijkstra's weakest precondition predicate transformers are mappings between sets of states, where the state space is a set $\mathcal{S}$ with the discrete order $\leq_d$ and the discrete topology τ_d. But $(\mathcal{S}, \leq_d, \tau_d)$ is not a Priestley space since it is not compact. A one-point compactification of $\mathcal{S}$ yields a Priestley space $(U, \leq_d, \tau_\perp)$, where $U = \mathcal{S} \cup \{\perp\}$. Let $\mathcal{P}(U)_\perp$ denote the subsets of U containing $\perp$, and $\mathcal{P}_f(\mathcal{S})$ denote the finite subsets of $\mathcal{S}$. Let WP_1 denote the set of Dijkstra's [Dij75, Dij76] weakest precondition mappings $wp(\alpha, -) : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$, and

OPT the set of observable predicate transformers $f : \mathcal{U}U \to \mathcal{U}U$. The clopen upper subsets of the Priestley space $(U, \leq_d, \tau_\perp)$ are the finite subsets of $\mathcal{S}$, the set U and the cofinite subsets of U containing $\perp$. Then:

2.6.2 Theorem WP_1 *is order-isomorphic to* OPT.
Proof: Define a mapping $\mu : \mathsf{WP}_1 \to \mathsf{OPT}$ by:

$$\mu(wp(\alpha, -))(Q) = \begin{cases} wp(\alpha, Q) & \text{if } Q \in \mathcal{P}_f(\mathcal{S}) \\ U & \text{otherwise} \end{cases}$$

for $Q \in \mathcal{U}U$. Define a mapping $\eta : \mathsf{OPT} \to \mathsf{WP}_1$ by $\eta(f)(Q) = f(Q)$ for $Q \in \mathcal{P}_f(\mathcal{S})$. These mappings are well defined. Then for $Q \in \mathcal{P}_f(\mathcal{S})$, $\mu \circ \eta(f)(Q) = \mu(f(Q)) = f(Q)$ and $\eta \circ \mu(wp(\alpha, -))(Q) = \eta(wp(\alpha, Q)) = wp(\alpha, Q)$. Therefore WP_1 and OPT are order-isomorphic. □

Let WP_2 denote the set of Dijkstra and Scholten's [DiS90] weakest precondition mappings $wp(\alpha, -) : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$, and VPT the set of verifiable predicate transformers $f : \mathcal{O}U \to \mathcal{O}U$. The open upper subsets of the Priestley space $(U, \leq_d, \tau_d)$ are the subsets of $\mathcal{S}$, the set U and the cofinite subsets of U containing $\perp$. Then the mappings in Theorem **2.6.2** define an order-isomorphism between WP_2 and VPT if the restriction to finite subsets of $\mathcal{S}$ is removed in the definition of η. Thus:

2.6.3 Theorem WP_2 *is order-isomorphic to* VPT. □

Let WLP denote the set of weakest liberal precondition mappings $wlp(\alpha, -) : \mathcal{P}(U)_\perp \to \mathcal{P}(U)_\perp$, and RPT the set of refutable predicate transformers $f : \mathcal{C}U \to \mathcal{C}U$. The closed upper sets of $(U, \leq_d, \tau_\perp)$ are subsets of U containing $\perp$, the finite subsets of $\mathcal{S}$ and the set U. Then:

2.6.4 Theorem WLP *is order-isomorphic to* RPT.
Proof: Define a mapping $\mu : \mathsf{WLP} \to \mathsf{RPT}$ by:

$$\mu(wlp(\alpha, -))(Q) = \begin{cases} wlp(\alpha, Q \cup \{\perp\}) & \text{if } Q \in \mathcal{P}_f(\mathcal{S}) \\ wlp(\alpha, Q) & \text{if } Q \in \mathcal{P}(U)_\perp \end{cases}$$

Define a mapping $\eta : \mathsf{RPT} \to \mathsf{WLP}$ by $\eta(f)(Q) = f(Q)$ for $Q \in \mathcal{P}(U)_\perp$. These mappings are well defined. Then for $Q \in \mathcal{P}(U)_\perp$, $\mu \circ \eta(f)(Q) = \mu(f(Q)) = f(Q)$ and $\eta \circ \mu(wlp(\alpha, -))(Q) = \eta(wlp(\alpha, Q)) = wlp(\alpha, Q)$. Therefore WLP and RPT are order-isomorphic. □

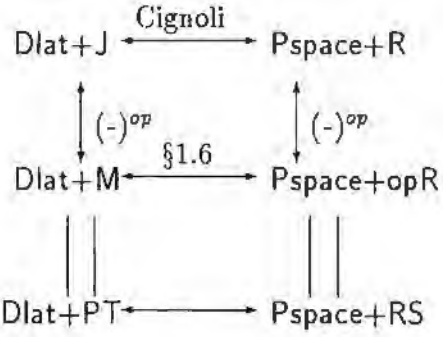
To introduce the relational and algebraic presentations of program semantics, I consider observable predicate transformers for two reasons. First, observable predicate transformers can be viewed as power operations of programs-as-opposite-Priestley-relations (Theorem **1.6.8**). Second, clopen upper sets form

a bounded distributive lattice of observable positive properties of states (Theorem **1.5.4**) and observable predicate transformers are meet homomorphisms over this lattice.

Therefore, a natural *relational presentation of program semantics based on Priestley spaces* is obtained by taking the state space to be a Priestley space $(X, \leq, \tau)$ and representing programs as opposite Priestley relations between states in X. The condition (in Definition **1.6.6**(a)) that $R(x)$ is a closed upper set for $x \in X$ means that the output of programs can be described in terms of refutable positive properties. This relational representation of programs therefore captures the input-output behaviour of finitely nondeterministic programs including those for which nontermination from a state is possible. The condition (in Definition **1.6.6**(b)) that R gives rise to a mapping between clopen upper sets means that observable predicate transformers are power operations of programs-as-opposite-Priestley-relations and that properties of opposite Priestley relations are relational counterparts of the properties of observable predicate transformers. A Priestley space with programs-as-opposite-Priestley-relations I call a *program Priestley space*. Denote the class of all program Priestley spaces by Pspace+RS.

The corresponding *algebraic presentation of program semantics* is given in terms of a bounded distributive lattice $(D, \vee, \wedge, 0, 1)$ endowed with a collection $\mathcal{M}$ of meet homomorphisms preserving 1 $f : D \to D$. I call such mappings *Priestley predicate transformers*, and a bounded distributive lattice with Priestley predicate transformers a *Priestley predicate transformer lattice*. Denote the class of all Priestley predicate transformer lattices by Dlat+PT.

The duality in Section 1.6 can now be restated as a duality between Dlat+PT and Pspace+RS. More precisely,



That is,

2.6.5 Theorem *Every Priestley predicate transformer lattice can be represented as the Priestley predicate transformer lattice of some program Priestley space.* □

On the other hand,

2.6.6 Theorem *Every program Priestley space can be represented as the program Priestley space of some Priestley predicate transformer lattice.* □

Therefore, applying the techniques of Priestley duality, I have established that the relational and algebraic presentations of program semantics based on Priestley spaces mutually characterise each other, and hence have verified the arrow between relational semantics and predicate transformer semantics in the program semantics triangle (on p 6).

Chapter 3

Information Systems

The purpose of this Chapter is to provide a logical presentation of program semantics related to the relational presentation based on Priestley spaces, and related to the algebraic presentation based on bounded distributive lattices. That is, it deals with the top left-hand corner of the program semantics triangle (in Sections 3.1, 3.2, 3.3), the arrow between information systems and relational semantics (in Section 3.4), and the arrow between information systems and predicate transformer semantics (in Section 3.5). Priestley duality, in particular the part showing how a Priestley space can be recovered from its clopen upper sets, lies at the heart of this Chapter.

3.1 Finite states viewed as an information system

In this Section I show how the finite states of the Priestley space $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ can be viewed as a Scott information system (Definition **1. 4. 1**) from which the domain $(\mathsf{State}, \sqsubseteq)$ can be constructed. For this I found it useful to represent a state in State as a propositional *theory* describing the values of variables known in the state, rather than as an infinite three-valued vector (as in Section 2.1). (A similar representation is used in [BrG95] for finite and infinite Boolean valuations.)

Consider a propositional language $\mathcal{L}$ built up as a free algebra from the denumerably infinite set *Var* of program variables using the classical logical connectives $\wedge$ ('and'), $\vee$ ('or') and $\neg$ ('not'). I refer to a variable with or without a negation sign as a *literal*. By a *theory* I mean any set of propositional formulae, not

necessarily closed under logical implication and not necessarily consistent. For any theories T_1 and T_2 I define:

3.1.1 $T_1 \wedge T_2 = T_1 \cup T_2$
$T_1 \Rightarrow T_2$ iff $T_1 \wedge T_2 = T_1$ iff $T_2 \subseteq T_1$.

I now represent any state $s \in \mathsf{State}$ as a propositional theory in the following way: by asserting the variables with value 1, by asserting the negations of variables with value 0, and by not saying anything about variables with value $\perp$. The resulting theories are necessarily consistent but are not closed under logical implication (since they only contain literals). More precisely, I represent the positive information content of s as the theory:

$$s_p^* = \{v_p \mid s(v_p) = 1\}$$

and the negative information content of s as the theory:

$$s_n^* = \{\neg v_n \mid s(v_n) = 0\}.$$

Then the state s can be represented as the theory:

$$s^* = s_p^* \wedge s_n^* = \{v_p \mid s(v_p) = 1\} \cup \{\neg v_n \mid s(v_n) = 0\}.$$

For finite states $s \in \mathsf{State}^0$, these theories are finite and can hence be expressed as a finite conjunction of literals:

$$s^* = v_{p_1} \wedge \ldots \wedge v_{p_a} \wedge \neg v_{n_1} \wedge \ldots \wedge \neg v_{n_b}$$

where $P_s = \{v_{p_1}, \ldots v_{p_a}\}$ and $N_s = \{v_{n_1}, \ldots v_{n_b}\}$. For atomic states $a \in \mathsf{Atom}$, these theories are singletons and can hence be expressed as a single literal:

$$a^* = v_p \text{ if } P_a = \{v_p\} \text{ and } a^* = v_n \text{ if } N_a = \{v_n\}.$$

To represent the bottom state I introduce a special proposition, denoted Δ. The ordering $\sqsubseteq$ between states can be represented as logical implication between theories as follows:

$$x \sqsubseteq y \text{ iff } y^* \Rightarrow x^* \text{ iff } x^* \subseteq y^*.$$

That is, the theory representing a state implies the theories representing its finite approximations.

It turns out that with these representations, atomic and finite states can be viewed as a Scott information system. Namely: $(\mathsf{Atom}, (\perp, \perp, \ldots), \mathsf{State}^0, \sqsubseteq)$ with $\sqsubseteq$ restricted to finite states.

3.1.2 Theorem $(\mathsf{Atom}, (\perp, \perp, \ldots), \mathsf{State}^0, \sqsubseteq)$ *is a Scott information system.*
Proof: It suffices to check that the propositional representations of states satisfy the six conditions in Definition **1.4.1**. Consider the structure $(\mathcal{D}, \Delta, \mathrm{Con}, \vdash)$ where $\mathcal{D}$ is the set of literals representing atomic

states, Δ is the least informative proposition representing the state $(\perp, \perp, \ldots)$, Con is the set of finite consistent sets of literals representing finite states, and $\vdash$ is logical implication between theories representing the ordering $\sqsubseteq$.
Then for any $u_1^*, u_2^*, v_1^*, v_2^*, w^* \in$ Con:

(a) $\emptyset \vdash \{\Delta\}$ since there is no finite state a such that $a \sqsubseteq (\perp, \perp, \ldots)$.

(b) If $u^* \vdash v^*$ then $u^* \cup v^* = u^*$ so $u^* \cup v^* \in$ Con.

(c) By reflexivity of $\sqsubseteq$, $u^* \vdash u^*$.

(d) By transitivity of $\sqsubseteq$, $u^* \vdash v^*$ and $v^* \vdash w^*$ imply $u^* \vdash w^*$.

(e) Suppose $u_2^* \supseteq u_1^*$ and $u_1^* \vdash v_1^*$ and $v_1^* \supseteq v_2^*$. Then

$$\begin{aligned} u_2^* \cup v_2^* &= (u_1^* \cup u_2^*) \cup (v_1^* \cap v_2^*) \\ &= (u_1^* \cup u_2^* \cup v_1^*) \cap (u_1^* \cup u_2^* \cup v_2^*) \\ &= (u_1^* \cup u_2^*) \cap (u_2^* \cup v_2^*) \\ &= (u_2^* \cap (u_2^* \cup v_2^*) \\ &= u_2^* \end{aligned}$$

So $u_2^* \vdash v_2^*$.

(f) Suppose $u_1^* \vdash v_1^*$ and $u_1^* \vdash v_2^*$. Then $u_1^* \cup (v_1^* \cup v_2^*) = (u_1^* \cup v_1^*) \cup v_2^* = u_1^* \cup v_2^* = u_1^*$. Thus $u_1^* \vdash (v_1^* \cup v_2^*)$.

Thus $(\mathcal{D}, \Delta, \text{Con}, \vdash)$ is a Scott information system, and hence $(\mathsf{Atom}, (\perp, \perp, \ldots), \mathsf{State}^0, \sqsubseteq)$ can be regarded as a Scott information system. □

I now show how the domain $(\mathsf{State}, \sqsubseteq)$ can be constructed from the information system $(\mathsf{Atom}, (\perp, \perp, \ldots), \mathsf{State}^0, \sqsubseteq)$ invoking again the representations of states as propositional theories. By Definition **1. 4. 2**, an element of the Scott information system $(\mathcal{D}, \Delta, \text{Con}, \vdash)$ is of the form

$$x^* = \bigcup \{\uparrow_{\vdash} u^* \mid u^* \in \text{Con and } u^* \subseteq x\}$$

where $\uparrow_{\vdash} u^* = \{d \in \mathcal{D} \mid u^* \vdash \{d\}\}$.

3.1.3 Theorem $(\mathsf{State}, \sqsubseteq)$ *is the information domain determined by* $(\mathsf{Atom}, (\perp, \perp, \ldots), \mathsf{State}^0, \sqsubseteq)$.
Proof: For this I show that states in State correspond to elements of the information domain determined by the information system $(\mathcal{D}, \Delta, \text{Con}, \vdash)$. Take any state $x \in \mathsf{State}$. Then $P_x = \bigcup \{P_a \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$ and $N_x = \bigcup \{N_a \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$. That is, in terms of propositional theories, $x_p^* = \bigcup \{a_p^* \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$ and $x_n^* = \bigcup \{a_n^* \mid a \in \mathsf{Atom} \text{ and } a \sqsubseteq x\}$. Thus

$$x^* = \bigcup \{a^* \mid a^* \in \mathcal{D} \text{ and } a \sqsubseteq x\},$$

and hence an element of the information domain determined by $(\mathcal{D}, \Delta, \mathrm{Con}, \vdash)$.
On the other hand, take any element x of the information domain determined by $(\mathcal{D}, \Delta, \mathrm{Con}, \vdash)$. Then

$$x = \bigcup\{\uparrow_{\vdash} u^* \mid u^* \in \mathrm{Con} \text{ and } u^* \subseteq x\}$$

where $\uparrow_{\vdash} u^* = \{d \in \mathcal{D} \mid u^* \vdash \{d\}\}$. But $\uparrow_{\vdash} u^* = u^*$ since $u^* \vdash \{d\}$ implies $\{d\} \subseteq u^*$. So $x = \bigcup\{u^* \mid u^* \in \mathrm{Con} \text{ and } u^* \subseteq x\}$. Then the theory x represents the information content of a state $s \in \mathsf{State}$. □

Therefore, if we assume the existence of a collection of finite facts about states in State and structure them abstractly as a Scott information system, then any state in State can be constructed abstractly from the finite facts true in it and any consistent collection of finite facts determines a state in State. The topology τ_{State} was constructed in Chapter 2 from finite states, and hence can be constructed from the information system $(\mathsf{Atom}, (\bot, \bot, \ldots), \mathsf{State}^0, \sqsubseteq)$. This presentation of the Priestley space (State, $\sqsubseteq$) is not surprising since the conditions characterising Scott information systems are derived from the feature that elements of a domain can be constructed from its compact elements. However not every Priestley space is a domain, and so not every Priestley space has so-called compact elements to work with. By Priestley duality (Theorem **1.5.6**), the points of a Priestley space can be constructed from its clopen upper sets. Therefore the next step in finding a suitable logical presentation of Priestley spaces is to determine whether the clopen upper subsets of a Priestley space can be structured to form some kind of information system.

3.2 Observable positive properties viewed as an information system

A generalisation of the notion of Scott information system is proposed in [EdS93] as a logical presentation of Boolean spaces (the Stone duals of Boolean algebras). Recall that Boolean algebras are bounded distributive lattices (in which every element has a complement), and Boolean spaces are Priestley spaces (in which the order is the specialisation order of the topology). In this Section, to obtain a logical presentation of Priestley spaces, I generalise the Edalaat/Smyth notion of information system in the same way that bounded distributive lattices generalise Boolean algebras. Using the Priestley space (State, $\sqsubseteq, \tau_{\mathsf{State}}$) I argue that this is a reasonable notion of information system. It turns out that Priestley spaces can be constructed from Priestley information systems using a Priestley-type duality.

I define a notion of *Priestley information system* consisting of a collection of propositions endowed with two binary operations and an entailment relation which is a preorder. The conditions imposed are syntactic counterparts of the conditions on a bounded distributive lattice.

3.2.1 Definition A *Priestley information system* is a structure $(H, \vdash, \wedge, \vee, \bot, \top)$ where H is a countable set of propositions, $\vdash$ is a preorder over H, $\wedge$ and $\vee$ are binary operations over H, $\bot$ is a distinguished proposition (the least informative proposition), and $\top$ is a distinguished proposition (the most informative proposition), such that for $x, y, z \in H$:

(a) $x \vdash \top$; $\quad \bot \vdash x$;

(b) $x \vdash x \vee y$; $\quad y \vdash x \vee y$; $\quad x \vdash z$ and $y \vdash z$ imply $x \vee y \vdash z$

(c) $x \wedge y \vdash x$; $\quad x \wedge y \vdash y$; $\quad x \vdash y$ and $x \vdash z$ imply $x \vdash y \wedge z$.

Priestley information systems generalise Scott information systems in that the existence of a base set H from which consistent sets are constructed is not assumed, and axioms such as **1. 4. 1**(c) derived from domains are dropped. Another generalisation of Scott information systems is considered in [DrG90] as a logical presentation of Priestley spaces. However, the absence of binary operations $\vee$ and $\wedge$ on this information system make it difficult to capture a notion of nondeterminism for programs. Priestley information systems generalise the Edalaat/Smyth [EdS93] notion of information system in that the unary operation corresponding to Boolean complementation and the associated axioms are dropped.

For an example of a Priestley information system, I consider the Priestley space $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$. By Priestley duality, the clopen upper subsets of $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ form a bounded distributive lattice $(\mathcal{U}\mathsf{State}, \subseteq)$ under set inclusion, and hence form a Priestley information system $(\mathcal{U}\mathsf{State}, \subseteq, \cap, \cup, \emptyset, \mathsf{State})$. To gain an intuitive understanding of the notion of Priestley information system, I give an order-theoretic characterisation of the clopen upper subsets of $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ and an interpretation of its Priestley information system.

3.2.2 Theorem *Any clopen upper subset of* $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ *is of the form* $\uparrow X$ *(finite* $X \subseteq \mathsf{State}^o$*).*

Proof: For any $x \in \mathsf{State}^o$, $\uparrow\{x\} = \bigcap\{\uparrow\{a\} \mid a \sqsubseteq x, a \in \mathsf{Atom}\}$ is a finite intersection of clopen upper sets, and hence a clopen upper set. For any finite $X \subseteq \mathsf{State}^o$, $\uparrow\{X\}$ is the finite union of clopen upper sets $\uparrow\{x\}$ $(x \in X)$ and hence a clopen upper set. To show that every clopen upper set is of this form, take any clopen upper set $U \subseteq \mathsf{State}$. Then U, being an open upper set, is open in the upper topology. Since $\mathcal{B}_u$ is a base for τ_u, $U = \bigcup_{x \in X_u} \uparrow\{x\}$ where $X_u \subseteq \mathsf{State}^o$. But U is closed and hence compact, so $U = \bigcup_{x \in X} \uparrow\{x\} = \uparrow X$ for some finite

$X \subseteq X_u \subseteq \mathsf{State}^o$. In general, this union cannot be simplified further. □

Let $\mathcal{P}_f(\mathsf{State}^0)$ denote the set of finite sets of finite states. Define an entailment relation on $\{\uparrow X \mid X \in \mathcal{P}_f(\mathsf{State}^0)\}$ by

$$\uparrow X \vdash \uparrow Y \text{ iff } \uparrow X \subseteq \uparrow Y, \quad \text{for } X, Y \in \mathcal{P}_f(\mathsf{State}^0).$$

Note that:

3.2.3 Theorem For $X, Y \in \mathcal{P}_f(\mathsf{State}^0)$,

$$\uparrow X \vdash \uparrow Y \quad \textit{iff} \quad (\forall x \in X)(\exists y \in Y)[y \sqsubseteq x].$$

Proof: Assume $\uparrow X \vdash \uparrow Y$. Take any $x \in X$. Then $x \in \uparrow X$, so $x \in \uparrow Y$ and hence there is some $y \in Y$ such that $y \sqsubseteq x$. On the other hand, assume for every $x \in X$ there is some $y \in Y$ such that $y \sqsubseteq x$. Take $z \in \uparrow X$. Then there is some $x \in X$ such that $x \sqsubseteq z$. Since $x \in X$ there is some $y \in Y$ such that $y \sqsubseteq x \sqsubseteq z$. Hence $z \in \uparrow Y$, so $\uparrow X \vdash \uparrow Y$. □

This means that the entailment relation $\vdash$ is the converse of the Smyth (power) ordering $\sqsubseteq_1^+$ (defined in Section 1.3) of the ordering $\sqsubseteq$. Using the properties of set-inclusion, it follows that:

3.2.4 Theorem $(\{\uparrow X \mid X \in \mathcal{P}_f(\mathsf{State}^0)\}, \vdash, \cap, \cup, \emptyset, \mathsf{State})$ *is a Priestley information system.* □

Recall from Chapter 2 that the clopen upper sets of $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ can be interpreted as observable positive properties. Intuitively, then the Priestley information system $(\{\uparrow X \mid X \in \mathcal{P}_f(\mathsf{State}^0)\}, \vdash, \cap, \cup, \emptyset, \mathsf{State})$ consists of observable positive properties about states, some of which may be more informative than others. Since $(\mathsf{State}, \sqsubseteq, \tau_{\mathsf{State}})$ is a Priestley space, there are enough of these properties to distinguish between distinct states. The entailment relation (between propositions) captures the dependencies between properties: using the characterisation of $\vdash$ in Theorem **3.2.3**, an entailment $\uparrow X \vdash \uparrow Y$ can be read as 'X informationally entails Y' in the sense that for every element in X there is an element in Y with less information (with respect to $\sqsubseteq$). That $\vdash$ is a preorder not a partial order means that although two observable positive properties may be the same they may be generated from different finite sets of finite states.

With these ideas in mind, we may intuitively think of the propositions of a Priestley information system as observable positive properties about states — that is, information about states that can be comprehended in a finite amount of time. Then the entailment relation captures information entailment in the sense that $x \vdash y$ means that property y describes at most the information described by

property x. This entailment relation is a preorder not a partial order which simply means that properties satisfied by the same states are not identified. That is, as in any logical context we are concerned not only with 'what properties we can express' but also with 'how we can say them'. Therefore Priestley information systems provide a reasonable logical presentation of Priestley spaces and capture a notion of information about states.

By Priestley duality, a Priestley space can be constructed from a bounded distributive lattice using prime filters. Therefore, to define a Priestley space from a Priestley information system, I introduce a notion of *state* of a Priestley information system as a syntactic version of the notion of prime filter of a bounded distributive lattice.

3.2.5 Definition Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. Then a non-empty subset $S \subseteq H$ is called a *state* iff for $x, y \in H$,

(a) $x \in S$ and $x \vdash y$ then $y \in S$ (upclosed with respect to $\vdash$);

(b) $x, y \in S$ implies $(\exists z \in S)[z \vdash x, y]$ (directed with respect to $\vdash$);

(c) $x \vee y \in S$ implies $(\exists z \in S)[z \vdash x \text{ or } z \vdash y]$ (primeness).

Compare this with the notion of *ideal* of an ordered set (defined in (e.g.) ([DaP90] Ex 3.19)). An ideal I of an ordered set $(X, <)$ is a directed downclosed set (with respect to $<$), that is, $x \in I$ and $y < x$ imply $y \in I$, and $x, y \in I$ implies $x, y < z$ for some $z \in I$. Then the conditions **3.2.5**(a) and (b) are 'opposite' to these conditions.

Let $\mathcal{S}H$ denote the set of all states of a Priestley information system $(H, \vdash, \wedge, \vee, \bot, \top)$. States can be partially ordered by (set) inclusion with '$S \subseteq T$' intuitively read as 'every observable positive property true of S is also true of T'. The poset $(\mathcal{S}H, \subseteq)$ can be topologised in the same way as the set of prime filters of a bounded distributive lattice is topologised in Section 1.5. For each $x \in H$, define

$$\mu_H(x) = \{S \in \mathcal{S}H \mid x \in S\}.$$

Let $\mathcal{S} = \{\mu_H(x) \mid x \in H\} \cup \{\mathcal{S}H - \mu_H(x) \mid x \in H\}$. Topologise $(\mathcal{S}H, \subseteq)$ with a topology $\Omega_{\mathcal{S}H}$ having $\mathcal{S}$ as subbasis. Then $(\mathcal{S}H, \subseteq, \Omega_{\mathcal{S}H})$ is the *information space associated with* $(H, \vdash, \wedge, \vee, \bot, \top)$.

3.2.6 Theorem *Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. Then the information space $(\mathcal{S}H, \subseteq, \Omega_{\mathcal{S}H})$ is a Priestley space.*

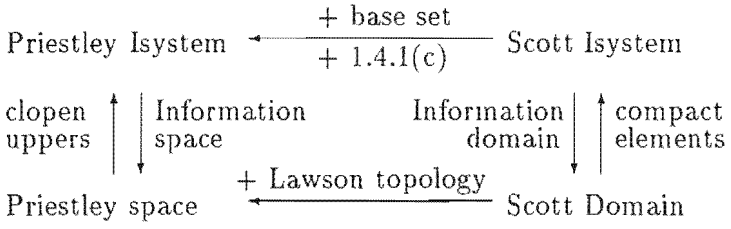
Proof: The topology $\Omega_{\mathcal{S}H}$ is the subspace topology on $\mathcal{S}H$ of the topology τ_H on $\mathcal{P}H$ which has a subbasis consisting of sets of the form

$$U_x = \{S \in H \mid x \in S\} \text{ and } \overline{U_x} = \{S \in H \mid x \notin S\}$$

for $x \in H$. This topology τ_H is compact ([DaP90] Ex 10.20). Now to show compactness of $(\mathcal{S}H, \Omega_{\mathcal{S}H})$ I show that $\mathcal{S}H$ is a closed in $\mathcal{P}(H)$.

Take $Y \in \mathcal{P}(H) - \mathcal{S}H$. Then there are some $A, B \subseteq H$ such that $A \subseteq Y$ and $A \vdash B$ and $B \subseteq \overline{Y}$. Let $C = A \cup B$ and $\mathcal{U} = \{V \mid V \cap C = Y \cap C\}$. then $\mathcal{U}$ is τ_H-clopen and $Y \in \mathcal{U} \notin \mathcal{S}H$. Therefore $\mathcal{S}H$ is a closed subset of $\mathcal{P}H$, and hence $(\mathcal{S}H, \Omega_{\mathcal{S}H})$ is a compact space. For total order-disconnectedness, suppose $X \not\subseteq Y$ for some $X, Y \in \mathcal{S}H$. Take $x \in H$ such that $x \in X - Y$. Let $\mathcal{U} = \{V \subseteq H \mid x \in V\}$. Then $\mathcal{U} \cap \mathcal{S}H$ is a clopen upper subset of $(\mathcal{S}H, \subseteq, \Omega_{\mathcal{S}H})$ with $X \in \mathcal{U}$ and $Y \notin \mathcal{U}$. Therefore $(\mathcal{S}H, \subseteq \Omega_{\mathcal{S}H})$ is a Priestley space. □

Therefore, the construction of a Priestley space from a Priestley information system is similar to that (given in Section 1.4) of a Scott domain from a Scott information system. The exact relationship can be summarised as follows:



The next step is to find a representation of programs over Priestley information systems related to the relational and algebraic representations given in §2.6.

3.3 Programs as relations between observable positive properties

In this Section I introduce a notion of *Priestley information-respecting relation* over Priestley information systems as a generalisation of the notion of Scott approximable mapping ([Sco82] Definition 5.1) over Scott information systems. I compare this representation with Hoare's [Hoa69] representation of programs as *Hoare triples* ([JaG85]).

3.3.1 Definition Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. A relation $R \subseteq H \times H$ is called a *Priestley information-respecting relation* if for all $u, v, u', v' \in H$:

(a) $\bot R \bot$ and $\top R \top$

(b) if $u' \vdash u$ and uRv then $u'Rv$

(c) uRv and $v \vdash v'$ then uRv'

(d) uRv and $u'Rv$ then $(u \vee u')Rv$

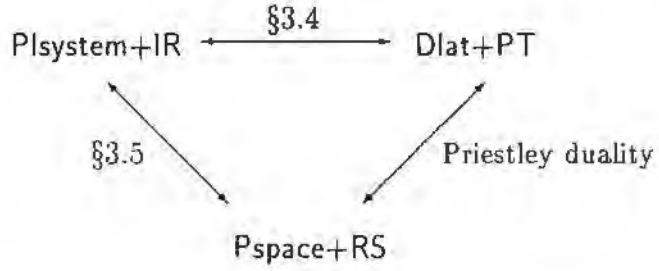
(e) uRv and uRv' then $uR(v \wedge v')$.

Since a Priestley information system is a collection of observable positive properties, Priestley information-respecting relations are binary relations between observable positive properties. Intuitively, the relationship uRv can be read as 'if (at least) property u holds of the input to R then (at least) property v will hold of the output'. Condition (a) can be read as saying 'if no information is provided about the input to R, no information can be provided about the output'. Condition (b) can be read as saying 'for a given property about the output to R the property about the corresponding input can be strengthened to a more precise description of the input to R'. On the other hand, condition (c) can be read as saying 'for a given property about the input to R, the property about the corresponding output can be weakened to include a description of all possible outputs'. Condition (d) captures the idea that different inputs may yield the same output; while condition (e) captures the idea that from a fixed input different consistent outputs may be possible (that is, nondeterminism).

Compare this representation to Hoare's [Hoa69] approach to program semantics (usually called *Hoare logic*). In Hoare logic the input-output behaviour of a program α is described by triples of the form $P\{\alpha\}Q$, where P is a precondition and Q is a postcondition for α. The idea is that if the property P holds when the program α begins, then the property Q holds if and when α terminates. There is an established connection between Hoare triples and precondition predicate transformers. Namely, for any program α, $P\{\alpha\}Q$ iff $P \subseteq f_\alpha(Q)$ where f_α is a precondition predicate transformer for program α ([Gri81]). The collection of precondition-postconditions pairs (P, Q) related by α in this way can be viewed as a binary relation between preconditions and postconditions of α which can be used to represent the program. Then reformulating the axioms and deduction rules constituting the calculus for Hoare logic I obtain the conditions on Priestley information-respecting relations. For example, Hoare's ([Hoa69] p 578) *rules of consequence* written as:
if $\vdash Q \supset P$ and $\vdash P\{\alpha\}R$ then $\vdash Q\{\alpha\}R$
if $\vdash P\{\alpha\}R$ and $\vdash R \supset S$ then $\vdash P\{\alpha\}S$,
where P, Q are preconditions and R, S are postconditions for α become respectively conditions **3. 3. 1**(b) and (c). (These are conditions have also been called *the law of precondition strengthening* and the *law of postcondition weakening* (in (e.g.) [Par81]).) Therefore, Priestley information-respecting relations are relational counterparts of Hoare triples, and hence Priestley information systems with Priestley information-respecting relations provide a reasonable logical presentation of program semantics.

Having dealt with the top left-hand corner of the program semantics triangle the next step is to relate this presentation to the algebraic and the relational presentations of Section 2.6.

That is, I need to establish the arrows labelled §3.4 and §3.5 in the following diagram:



where PIsystem+IR denotes the class of all Priestley information systems with Priestley information-respecting relations. For this I need to assume that the set $\{(u, v) \mid uRv\}$ is finite for every Priestley information-respecting relation R. This assumption is reasonable since as pointed out by Parikh ([Par83], p 112) an observer can only know a finite (possibly arbitrarily large) number of precondition-postcondition pairs for a given program.

3.4 Information systems and predicate transformer semantics

In this Section I deal with the arrow between information systems and predicate transformer semantics in the program semantics triangle. I show that a bounded distributive lattice is a Priestley information system in which the entailment relation is a partial order instead of a preorder. The translation from meet homomorphisms to Priestley information-respecting relations is based on the connection between Hoare triples and precondition predicate transformers. To obtain a bounded distributive lattice with meet homomorphisms from an information system with Priestley information-respecting relations, I apply the Lindenbaum/Tarski construction (described in Section 1.1).

I start by showing that a Priestley information system with Priestley information-respecting relations can be defined from a bounded distributive lattice with meet homomorphisms. Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. The underlying order $\leq$ of L (given by $x \leq y$ iff $x \wedge y = x$ for $x, y \in L$) is a partial order. Hence the structure $(L, \leq, \wedge, \vee, 0, 1)$ is a Priestley information system in which the entailment relation is a partial order. Thus:

3.4.1 Theorem *Any bounded distributive lattice is a Priestley information system.* □

Given a meet homomorphism $f : L \to L$, define a relation $\mathcal{I}f \subseteq L \times L$ by:

$$(u, v) \in \mathcal{I}f \text{ iff } u \leq f(v) \text{ for } u, v \in L.$$

(Note that this is the relationship between Hoare triples and precondition predicate transformers on p 61.)

3.4.2 Theorem *$\mathcal{I}f \subseteq L \times L$ is a Priestley information-respecting relation.*
Proof: I need to check that $\mathcal{I}f$ satisfies the conditions in Definition **3.3.1**.

(a) $f(1) = 1$ since f is a meet homomorphism, so $(1, 1) \in \mathcal{I}f$.
$0 \leq f(u)$, $\forall u \in L$, so $(0, 0) \in \mathcal{I}f$.

(b) Suppose $u' \vdash_L u$ and $(u, v) \in \mathcal{I}f$. Then $u' \leq u$ and $u \leq f(v)$, so $u' \leq f(v)$; hence $(u', v) \in \mathcal{I}f$.

(c) Suppose $(u, v) \in \mathcal{I}f$ and $v \vdash_L v'$. Then $u \leq f(v)$ and $v \leq v'$, so $u \leq f(v) \leq f(v')$ since f is monotone; hence $(u, v') \in \mathcal{I}f$.

(d) Suppose $(u, v) \in \mathcal{I}f$ and $(u', v) \in \mathcal{I}f$. Then $u \leq f(v)$ and $u' \leq f(v)$, so $u \vee u' \leq f(v)$ (by definition of least upper bound). Thus $(u \vee u', v) \in \mathcal{I}f$.

(e) Suppose $(u, v) \in \mathcal{I}f$ and $(u, v') \in \mathcal{I}f$. Then $u \leq f(v)$ and $u \leq f(v')$; so $u \leq f(v) \wedge f(v')$ (by definition of greatest lower bound). But f preserves finite meets so $u \leq f(v \wedge v')$ and hence $(u, v \wedge v') \in \mathcal{I}f$. □

Combining Theorems **3.4.1** and **3.4.2**, it follows that:

3.4.3 Theorem *For every bounded distributive lattice with meet homomorphisms $((L, \vee, \wedge, 0, 1), \mathcal{M})$, the structure $((L, \leq, \vee, \wedge, 0, 1), \{\mathcal{I}f \mid f \in \mathcal{M})$ is a Priestley information system with Priestley information-respecting relations.*

I now show that a bounded distributive lattice with meet homomorphisms can be defined from a Priestley information system with Priestley information-respecting relations. Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. Define an equivalence relation $\equiv$ (which identifies equivalent propositions) by:

$$x \equiv y \text{ iff } x \vdash y \text{ and } y \vdash x.$$

(It can easily be checked that $\equiv$ is an equivalence relation.) Let $[x] = \{y \in H \mid x \equiv y\}$, $H_{\equiv} = \{[x] \mid x \in H\}$, $[x] \vdash_{\equiv} [y]$ iff $x \vdash y$ (or $x = y$). It can easily be checked that $\vdash_{\equiv}$ is well defined, that is, $[x] \vdash_{\equiv} [y], [x] = [x_1]$ and $[y] = [y_1]$ together imply $[x_1] \vdash_{\equiv} [y_1]$. The least and greatest elements are respectively $\bot_{\equiv} = [\bot]$ and $\top_{\equiv} = [\top]$. Also, $\vdash_{\equiv}$ is a partial ordering. The next step is to define join and meet on $H_{\equiv}$. Let $[x] \wedge_{\equiv} [y] = [x \wedge y]$ and $[x] \vee_{\equiv} [y] = [x \vee y]$. Since $(H_{\equiv}, \vdash_{\equiv})$ is the Lindenbaum algebra of $(H, \vdash, \vee, \wedge, \bot, \top)$ it follows that:

3.4.4 Theorem *$(H_{\equiv}, \vdash_{\equiv})$ is a bounded distributive lattice (with $[\top]$ as top and $[\bot]$ as bottom).* □

Given a Priestley information-respecting relation $R \subseteq H \times H$, define a mapping $R_{\equiv} : H_{\equiv} \to H_{\equiv}$ by:

$$R_{\equiv}([y]) = \bigvee_{\equiv} \{[x] \mid xRy\} \text{ for } [y] \in H_{\equiv}.$$

Recall that $\{(x,y) \mid xRy\}$ is finite, and hence the join exists in the lattice $H_{\equiv}$ (by [DaP90] Lemma 2.11). Then:

3.4.5 Theorem *$R_{\equiv} : H_{\equiv} \to H_{\equiv}$ is a meet homomorphism.*
Proof: (I will drop the subscripts $_{\equiv}$ on $\bigvee$ in this proof.) Take any $[x], [y] \in H_{\equiv}$. Let $F_y = \{[x] \mid (x,y) \in R\}$ for $y \in H$. Then $F_{y \wedge z} = F_y \cap F_z$ where $\subseteq$ holds by **3.2.1**(c) and **3.3.1**(c), and $\supseteq$ holds by **3.3.1**(e). Thus $\bigvee F_{y \wedge z} = \bigvee(F_y \cap F_z)$. By **3.2.1**(c), $\bigvee(F_y \cup F_z) \vdash \bigvee F_y \cup \bigvee F_z$. For the other direction, let $F_y = F_{y_1} \cap F_{y \wedge z}$ and $F_z = F_{z_1} \cap F_{y \wedge z}$ for $y_1, z_1 \in H$. Then $\bigvee F_y \wedge \bigvee F_z \vdash \bigvee F_{y \wedge z} \vee (\bigvee F_{y_1} \wedge \bigvee F_{z_1}) = \bigvee F_{y \wedge z} \vee \{\bot\} = \bigvee F_{y \wedge z}$. Therefore $R_{\equiv}([y] \wedge [z]) = \bigvee(F_y \cap F_z) = \bigvee F_y \wedge \bigvee F_z = R_{\equiv}([y]) \wedge R_{\equiv}([z])$. $R_{\equiv}$ preserves the top $[\top]$ since $R_{\equiv}([\top]) = \bigvee\{[x] \mid (x, \top) \in R\} = \{[\top]\}$ by **3.3.1**(a) and **3.2.1**(a). □

Combining Theorems **3.4.4** and **3.4.5**, it follows that:

3.4.6 Theorem *For every Priestley information system with Priestley information-respecting relations $((H, \vdash, \wedge, \vee, \bot, \top), \mathcal{I})$, the Lindenbaum/Tarski algebra $((H_{\equiv}, \vdash_{\equiv}), \{R_{\equiv} \mid R \in \mathcal{I}\})$ is a bounded distributive lattice with meet homomorphisms.* □

Priestley information systems $(H, \vdash, \vee, \wedge, \bot, \top)$ in which the preorder $\vdash$ is a partial order are bounded distributive lattices. Hence:

3.4.7 Theorem *Every bounded distributive lattice with meet homomorphisms $((L, \vee, \wedge, 0, 1), \mathcal{M})$ is lattice isomorphic to $((L, \vee, \wedge, 0, 1), \{(\mathcal{I}f)_{\equiv} \mid f \in \mathcal{M}\})$ under id_L.*
(Here the isomorphism is a lattice isomorphism ψ over $(L, \vee, \wedge, 0, 1)$ which preserves the meet homomorphisms in the sense that for every $f \in \mathcal{M}$, $\psi \circ f = (\mathcal{I}f)_{\equiv} \circ \psi$.)
Proof: Let $((L, \vee, \wedge, 0, 1), \mathcal{M})$ be a bounded distributive lattice with meet homomorphisms. Since every bounded distributive lattice is a Priestley information system in which the entailment relation is a partial order, and *vice versa*, it suffices to show that: $f = (\mathcal{I}f)_{\equiv}$. Take any $y \in L$. Then $(\mathcal{I}f)_{\equiv}(y) = \bigvee\{x \mid (x,y) \in \mathcal{I}f\} = \bigvee\{x \mid x \leq f(y)\} = f(y)$. □

For Priestley information systems with Priestley information-respecting relations I have:

3.4.8 Theorem *Every Priestley information system with Priestley information-respecting relations* $((H, \vdash, \wedge, \vee, \bot, \top), \mathcal{H})$ *is order-isomorphic to* $((H_{\equiv}, \vdash_{\equiv}, \wedge_{\equiv}, \vee_{\equiv}, \top_{\equiv}, \bot_{\equiv}), \{\mathcal{I}(R_{\equiv}) \mid R \in \mathcal{H}\})$.
(Here the order-isomorphism is an order-embedding of H onto $H_{\equiv}$ which is structure-preserving in the sense that for every $R \in \mathcal{H}$ and $x, y \in H$, $(x, y) \in R$ iff $(\psi(x), \psi(y)) \in (\mathcal{I}R)_{\equiv}$.)
Proof: Let $((H, \vdash, \wedge, \vee, \bot, \top), \mathcal{H})$ be a Priestley information system with Priestley information-respecting relations. Then $(H_{\equiv}, \vdash_{\equiv})$ is a bounded distributive lattice and hence a Priestley information system. Define a mapping $\psi : (H, \vdash, \wedge, \vee, \bot, \top) \to (H_{\equiv}, \vdash_{\equiv}, \wedge_{\equiv}, \vee_{\equiv}, \bot_{\equiv}, \top_{\equiv})$ by: $\psi(x) = [x]$. Then ψ is well defined. Since $x \vdash y$ (or $x = y$) iff $[x] \vdash [y]$ (or $[x] = [y]$), ψ is an order-embedding. Also for every $[y] \in H_{\equiv}$, $\psi(y) = [y]$ so ψ is surjective. Therefore ψ is an order-isomorphism.
It remains to show that ψ is structure-preserving, that is, $(x, y) \in R$ iff $([x], [y]) \in (\mathcal{I}R)_{\equiv}$. Suppose $(x, y) \in R$. Then $[x] \vdash \bigvee_{\equiv}\{[z] \mid (z, y) \in R\}$, so $[x] \vdash R_{\equiv}([y])$, and hence $([x], [y]) \in \mathcal{I}(R_{\equiv})$. On the other hand, suppose $([x], [y]) \in \mathcal{I}(R_{\equiv})$ Then $[x] \vdash_{\equiv} R_{\equiv}([y])$, so by **3.2.1** there is some $[x']$ such that $(x', y) \in R$ and $[x] \vdash [x']$ and hence by **3.3.1**(c) $(x, y) \in R$. □

3.5 Information systems and relational semantics

In this Section I deal with the arrow between information systems and relational semantics in the program semantics triangle. I use Priestley duality to relate Priestley spaces and Priestley information systems, and relational counterparts of the translations between opposite Priestley relations and meet homomorphisms to translate between opposite Priestley relations and Priestley information-respecting relations.

I start by showing that a Priestley information system with Priestley information-respecting relations can be defined from a Priestley space with opposite Priestley relations. Let $(X, \leq, \tau)$ be a Priestley space. By Lemma **1.5.4**, the lattice of its clopen upper sets $(\mathcal{U}X, \subseteq)$ is a bounded distributive lattice and by Lemma **3.4.1** $(\mathcal{U}X, \subseteq, \cap, \cup, \emptyset, X)$ is a Priestley information system. Therefore,

3.5.1 Theorem *The lattice of clopen upper sets of a Priestley space is a Priestley information system.* □

Given an opposite Priestley relation $R \subseteq X \times X$, define a relation $\mathcal{H}R \subseteq \mathcal{U}X \times \mathcal{U}X$ by:

$(U,V) \in \mathcal{H}R$ iff $(\forall u, v \in X)[(u \in U \text{ and } (u,v) \in R) \text{ imply } v \in V]$ for $U, V \in \mathcal{U}X$.

Now:

3.5.2 Theorem *$\mathcal{H}R \subseteq \mathcal{U}X \times \mathcal{U}X$ is a Priestley information-respecting relation.*
Proof: I show that for any opposite Priestley relation R, $\mathcal{H}R = \mathcal{IU}R$ where $\mathcal{U}R$ is the meet homomorphism associated with R (defined in Section 1.3), and then invoke Theorem **3.4.2** to deduce that $\mathcal{H}R$ is a Priestley information-respecting relation. Take any $U, V \in \mathcal{H}X$, such that $(U,V) \in \mathcal{H}R$. Then for every $u \in U$, $R(u) \subseteq V$, and so $u \in \mathcal{U}R(V)$. Thus $U \subseteq \mathcal{U}R(V)$, which is exactly the definition of the Priestley information-respecting relation $\mathcal{IU}R$ associated with the meet homomorphism $\mathcal{U}R$. Therefore, $\mathcal{H}R = \mathcal{IU}R$ is a Priestley information-respecting relation. □

Combining Theorems **3.5.1** and **3.5.2**, it follows that:

3.5.3 Theorem *For every Priestley space with opposite Priestley relations $((X, \leq, \tau), \mathcal{R})$, the structure $((\mathcal{U}X, \subseteq, \cap, \cup, X, \emptyset), \{\mathcal{H}R \mid R \in \mathcal{R}\})$ is a Priestley information system with Priestley information-respecting relations.* □

I now show that a Priestley space with opposite Priestley relations can be defined from a Priestley information system with Priestley information-respecting relations. Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. Then by Theorem **3.2.6**, $(\mathcal{S}H, \subseteq, \Omega_{\mathcal{S}H})$ is a Priestley space.

Given a Priestley information-respecting relation $R \subseteq H \times H$, a relation $\mathcal{S}R \subseteq \mathcal{S}H \times \mathcal{S}H$ can be defined by:

$$(U,V) \in \mathcal{S}R \text{ iff } (\forall y \in H)[f_R(y) \in U \Rightarrow y \in V] \text{ for } U, V \in \mathcal{S}H$$

where $f_R(y) = \bigvee\{x \mid xRy\}$ for $y \in H$. (Recall that $\{(x,y) \mid xRy\}$ is finite, so this join exists.) Now:

3.5.4 Lemma *The mapping f_R is a meet homomorphism over H.*
Proof: Note that $f_R(y) = z$ where $[z] = R_{\equiv}([y])$. By Theorem **3.4.5**, $R_{\equiv}$ is a meet homomorphism over $H_{\equiv}$, and hence f_R is a meet homomorphism over H. □

It now follows that:

3.5.5 Theorem *$\mathcal{S}R \subseteq \mathcal{S}H \times \mathcal{S}H$ is an opposite Priestley relation.*
Proof: By definition, for any $U, V \in \mathcal{S}H$, $(U,V) \in \mathcal{S}R$ iff $f_R^{-1}(U) \subseteq V$. Thus by Theorem **1.6.8** $\mathcal{S}R$ is an opposite Priestley relation. □

By Priestley duality, a Priestley space is order-homeomorphic to the Priestley space associated with some Priestley information system, namely, the prime filter space of the bounded distributive lattice of its clopen upper sets. Thus:

3.5.6 Theorem *Every Priestley space with opposite Priestley relations $((X, \leq, \tau), \mathcal{R})$ is order-isomorphic to*
$((\mathcal{SU}X, \subseteq, \Omega_{\mathcal{SU}X}), \{\mathcal{SH}R \mid R \in \mathcal{R}\}$.
(Here the isomorphism is an order-homeomorphism ϵ of $(X, \leq, \tau)$ onto $(\mathcal{SU}X, \subseteq, \Omega_{\mathcal{SU}X})$ which is structure-preserving in the sense that for every $R \in \mathcal{R}$ and $x, y \in X$, $(x,y) \in R$ iff $(\epsilon(x), \epsilon(y)) \in \mathcal{SH}R$.)
Proof: Let $((X, \leq, \tau), \mathcal{R})$ be a Priestley space with opposite Priestley relations. Then $(\mathcal{U}X, \subseteq)$ is a bounded distributive lattice and hence a Priestley information system. The information space $(\mathcal{SU}X, \subseteq, \Omega_{\mathcal{SU}X})$ is then the Priestley space $(\mathcal{FU}X, \subseteq, \Omega_{\mathcal{FU}X})$. By Priestley duality, $(X, \leq, \tau)$ is homeomorphic to $(\mathcal{SU}X, \subseteq, \Omega_{\mathcal{SU}X})$ under the mapping ϵ_X where $\epsilon_X(x) = \{U \in \mathcal{U}X \mid x \in U\}$. It remains to show that ϵ_X is structure-preserving, that is $(x,y) \in R$ iff $(\epsilon_X(x), \epsilon_X(y)) \in \mathcal{SH}R$. First, suppose $(x,y) \notin R$. Then since $R(x)$ is a closed upper set, there is some $V \in \mathcal{U}X$ such that $y \in V$ and $V \cap R(x) = \emptyset$. Then $V \in \epsilon_X(y)$ and $R(x) \not\subseteq V$, and hence $x \notin \mathcal{U}R(V)$. Thus $x \notin f_{\mathcal{H}R}(V)$, so $f_{\mathcal{H}R}(V) \notin \epsilon_X(x)$. Therefore $(\epsilon_X(x), \epsilon_X(y)) \notin \mathcal{SH}R$. Second, suppose $(x,y) \in R$. Take any $V \in \mathcal{U}X$, such that $\mathcal{U}R(V) \in \epsilon_X(x)$. Then $x \in \mathcal{U}R(V)$, that is $R(x) \subseteq V$, so $y \in V$ and hence $V \in \epsilon_X(y)$. Thus $(\epsilon_X(x), \epsilon_X(y)) \in \mathcal{SH}R$. □

For Priestley information systems with Priestley information-respecting relations I have:

3.5.7 Theorem *Every Priestley information system with Priestley information-respecting relations $((H, \vdash, \wedge, \vee, bot, \top), \mathcal{H})$ can be embedded in $((\mathcal{U}SH, \subseteq, \cap, \cup, \emptyset, \mathcal{S}H), \{\mathcal{H}SR \mid R \in \mathcal{H}\})$.*
Proof: Define a mapping $\psi : (H, \vdash, \wedge, \vee, \bot, \top) \to (\mathcal{U}SH, \subseteq, \cap, \cup, \emptyset)$ by $\psi(x) = \mu_H(x) = \{F \in \mathcal{S}H \mid x \in F\}$. Then $(\mathcal{U}SH, \subseteq)$ is a Priestley information system. Every $\mu_H(x)$ is a clopen upper set of the Priestley space $(\mathcal{S}H, \subseteq, \Omega_{\mathcal{SH}})$ but not every member of $\mathcal{U}SH$ is necessarily of the form $\mu_H(x)$ for some $x \in H$. The mapping ψ is an order-embedding. For suppose $x \vdash y$ and $F \in \psi(x)$. Then $x \in F$ and hence $y \in F$ since $F \in \mathcal{S}H$. On the other hand, suppose $x \nvdash y$. Then for some

$F \in \mathcal{US}H$, $x \in F$ and $y \notin F$, so $\psi(x) \not\subseteq \psi(y)$. It remains to show that ψ is structure-preserving. Suppose $(x, y) \in R$. Take any $F, G \in \mathcal{S}R$ such that $F \in \psi(x)$ and $(F, G) \in \mathcal{S}R$. Then $x \in F$ and for every $z \in H$, $f_R(z) \in F$ implies $z \in G$. Now $f_R(y) \in F$ so $y \in G$, and hence $G \in \psi(y)$. Thus $(\psi(x), \psi(y)) \in \mathcal{HS}R$. □

A Priestley information system $(H, \vdash, \wedge, \vee, \bot, \top)$ is a preordered set $(H, \vdash)$ with respect to its entailment relation. Hence the bounded distributive lattice $(\mathcal{US}H, \subseteq)$ is a completion of $(H, \vdash)$ via the order-embedding ψ in Theorem **3.5.7**. (Although this observation is worth mentioning in its own right it is not clear what the usefulness of such a completion would be.) Recall $\mathcal{S} = \{\mu_H(x) \mid x \in H\} \cup \{\mathcal{S}H - \mu_H(x) \mid x \in H\}$ is a subbasis for the topology $\Omega_{\mathcal{S}H}$ on $(\mathcal{S}H, \subseteq)$. So every clopen upper set (that is, member of $\mathcal{US}H$) is necessarily (by definition of a subbasis) expressible as a (possible infinite) join of finite meets of members of $\mathcal{S}$. Since a Priestley space is compact, a finite join suffices. Hence $(H, \vdash)$ join/meet generates $(\mathcal{US}H, \vdash)$ (in the above sense) and $(\mathcal{US}H, \vdash)$ is a join/meet completion of $(H, \vdash)$.

Chapter 4

Denotational Semantics

The purpose of this Chapter is to find an order-theoretical presentation of program semantics to obtain the program semantics triangle forecast in Chapter 1 (p 6). That is, it deals with the centre of the program semantics triangle (in Section 4.1), the arrow between denotational semantics and relational semantics (in Section 4.2), the arrow between denotational semantics and predicate transformer semantics (in Section 4.3) and the arrow between denotational semantics and information systems (in Section 4.4). The bijective correspondence between filters of a bounded distributive lattice and closed upper sets of a Priestley space, lies at the heart of this Chapter.

4.1 Refutable positive properties viewed as a domain

In this Section I construct a domain from a Priestley space, from a bounded distributive lattice and from a Priestley information system, and establish isomorphisms between the domains thus obtained. With the correspondences between Priestley spaces, bounded distributive lattices and Priestley information systems, it suffices to consider only one of these constructions. But since I have not yet fixed a starting point for unifying program semantics I present all three.

Though every Priestley space is a cpo ([DaP90] Ex 10.9(iv)), not every Priestley space is a domain. I now construct a domain from a Priestley space using a technique compatible with Smyth's powerspace construction (outlined in §1.3)

Let $(X, \leq, \tau)$ be a Priestley space. The upper topology τ_u on X has as basis the clopen upper sets of $(X, \leq, \tau)$. Let $\mathcal{C}X$ denote the set of all closed upper subsets of $(X, \leq, \tau)$. Topologise this set with a topology τ_{cu} having a base consisting of sets of the form $B_O = \{A \in \mathcal{C}X \mid A \subseteq O\}$ for $O \in \mathcal{U}X$. (My reason for using clopen upper sets instead of the open upper sets or open sets will become apparent in Theorem **4. 2. 2**). I call the space $(\mathcal{C}X, \tau_{cu})$, the *upper power space* of $(X, \leq, \tau)$. (This is the terminology of [Smy83, Bri93].)

The following Theorem shows that the specialisation order of τ_{cu} is the superset inclusion relation on members of $\mathcal{C}X$. (The definition of the specialisation order of a topology is given in Appendix 2.)

4.1.1 Theorem *For any* $S, T \in \mathcal{C}X$, $S \leq_{\tau_{cu}} T$ *iff* $S \supseteq T$.
Proof: Take any $S, T \in \mathcal{C}X$ with $S \leq_{\tau_{cu}} T$. Then for every $O \in \mathcal{U}X$, $S \in B_O$ implies $T \in B_O$; so for every $O \in \tau_{cu}$, $S \subseteq O$ implies $T \subseteq O$. Hence $T \subseteq \bigcap\{O \mid S \subseteq O\} = S$. For the converse, take any $S, T \in \mathcal{C}X$ with $S \supseteq T$. For any $O \in \mathcal{U}X$ with $S \in B_O$, $S \subseteq O$, and hence $T \subseteq O$. Thus $S \leq_{\tau_{cu}} T$. □

In the following sequence of results I show that $(\mathcal{C}X, \supseteq)$ is an algebraic dcpo, and hence a domain.

4.1.2 Lemma $(\mathcal{C}X, \supseteq)$ *is a dcpo.*
Proof: Since $(\mathcal{C}X, \supseteq)$ is a complete lattice, every subset $\mathcal{C} \subseteq \mathcal{C}X$ has a least upper bound, so in particular every directed set has a least upper bound in $\mathcal{C}X$. Now $X \in \mathcal{C}X$ and for every $A \in \mathcal{C}X$, $X \supseteq A$; so X is the least element of $\mathcal{C}X$. □

4.1.3 Lemma *Every clopen upper subset of* $(X, \leq, \tau)$ *is a compact element of* $(\mathcal{C}_u X, \supseteq)$.
Proof: Let $\mathcal{U} \in \mathcal{U}X$ be a clopen upper set and suppose $U \supseteq \bigcap \mathcal{S}$ for some set $\mathcal{S} \subseteq \mathcal{C}X$ directed with respect to superset inclusion. By compactness there exists a finite $\mathcal{S}' \supseteq \mathcal{S}$ such that $U \supseteq \bigcap \mathcal{S}'$. But $\mathcal{S}$ is directed thus there is some $S \in \mathcal{S}$ such that $S' \supseteq S$ for every $S' \in \mathcal{S}'$. Therefore, $U \supseteq \bigcap \mathcal{S}' \supseteq S$. □

The clopen upper sets form a basis for $(\mathcal{C}X, \supseteq)$ which means:

4.1.4 Theorem $(\mathcal{C}X, \supseteq)$ *is an algebraic dcpo.*
Proof: Take any $C \in \mathcal{C}X$. Let $\mathcal{C} = \{U \in \mathcal{U}X \mid U \supseteq C\}$. Then $\mathcal{C}$ is directed: For take any $U_1, U_2 \in \mathcal{C}$. Then $U_1 \cap U_2 \in \mathcal{U}X$ since $\mathcal{U}X$ is closed under finite intersections, and $U_1 \cap U_2 \supseteq C$ since $U_1 \supseteq C$ and $U_2 \supseteq C$. Thus $U_1 \cap U_2 \in \mathcal{C}$ and $U_1 \supseteq U_1 \cap U_2$ and $U_2 \supseteq U_1 \cap U_2$; so $\mathcal{C}$ is directed with respect to $\supseteq$. Since $(\mathcal{C}_u X, \supseteq)$ is a cpo, $\mathcal{C}$ has a least

upper bound and $\bigcap \mathcal{C} \supseteq C$. For the reverse inclusion, suppose $x \notin C$. Then by Lemma **1.5.2**, there is a clopen upper set U such that $C \subseteq U$ and $x \notin U$. Then $x \notin \bigcap\{U \in \mathcal{U}X \mid U \supseteq C\}$.Thus $\bigcap \mathcal{C} \subseteq C$, and hence $C = \bigcap \mathcal{C}$ as required to complete the proof. □

4.1.5 Corollary *If X is countable then $(\mathcal{C}X, \supseteq)$ is an ω-algebraic dcpo.* □

Therefore, the upper power space $(\mathcal{C}X, \tau_{cu})$ of a Priestley space $(X, \leq, \tau)$ is (in its specialisation order) the domain $(\mathcal{C}X, \supseteq)$. As shown in (e.g.) [Joh82] for every domain $(D, \sqsubseteq)$, the specialisation order of the Scott topology on the domain is the order $\sqsubseteq$. It turns out that the Scott topology and upper topology on $\mathcal{C}X$ coincide.

4.1.6 Theorem *Let $(X, \leq, \tau)$ be a Priestley space. Then the Scott topology of $(\mathcal{C}X, \supseteq)$ coincides with the upper topology on $\mathcal{C}X$.*
Proof: Consider a basic open set U_O ($O \in \mathcal{U}X$) of the upper topology of $\mathcal{C}X$. Then U_O is an upper set (with respect to $\supseteq$) since $A \in U_O$ and $A \supseteq B$ imply $B \subseteq O$. Take any set $S \subseteq \mathcal{C}X$ that is directed with respect to $\supseteq$ and such that $\bigcap S \in U_O$. Then $\bigcap S \subseteq O$ and hence, by compactness of (X, τ) and directedness of S, there is some $A \in S$ such that $A \in U_O$. Therefore, U_O is open in the Scott topology on $(\mathcal{C}X, \supseteq)$. Conversely, take any clopen upper set U of $(X, \leq, \tau)$. Then by Lemma **4.1.3**, U is a compact element of $(\mathcal{C}X, \supseteq)$, and hence $\uparrow U$ is a basic open set of the Scott topology on $(\mathcal{C}X, \supseteq)$. But $\uparrow U = \{A \in \mathcal{C}X \mid U \supseteq A\} = U_U$. Therefore, $\uparrow U$ is a basic open set of the upper topology on $\mathcal{C}X$. □

By Priestley duality (Theorem **1.5.6**), (a homeomorphic copy of) the original space can be recovered from the compact elements of its domain.

For the construction of a domain from a bounded distributive lattice, I use what I call a *filter completion*. This construction is 'opposite' to the ideal completion process of a poset (in Section 1.4) in the sense that filters of a lattice are 'opposite' to its ideals. It is a feasible construction since lattices, unlike domains, have both a top and a bottom element.

Let $(L, \leq)$ be a bounded distributive lattice. Define the *filter completion* of L to be the set $FiltL$ of all filters of L, ordered by subset inclusion. (For the definition of a filter of a lattice see Appendix 1.) In the following sequence of results I show that the filter completion of a bounded distributive lattice is a domain.

4.1.7 Lemma *$(FiltL, \subseteq)$ is a dcpo.*
Proof: In order to show that every directed subset of $(FiltL, \subseteq)$ has a lub, it is enough to show that the union of the filters of each directed

set is a filter of $(L, \leq)$. Let $\mathcal{E} \subseteq FiltL$ be directed with respect to $\subseteq$. Upclosedness of $\bigcup\mathcal{E}$ follows from the upclosedness of the filters in $\mathcal{E}$. Since $\mathcal{E}$ is directed, $\bigcup\mathcal{E}$ is closed under finite meets. Therefore $(FiltL, \subseteq)$ is a cpo. The upclosedness of filters ensures that every filter contains the top element 1 of the lattice. Since $\{1\}$ is upclosed and directed, this set defines the bottom of $(FiltL, \subseteq)$. □

The principal filters of L are the compact elements of $(FiltL, \subseteq)$. (For the definition of a principal filter see Appendix 1.)

4.1.8 Lemma *For any $a \in L$, $\uparrow\{a\}$ is a compact element of $(FiltL, \subseteq)$.*
Proof: Take any $a \in L$ and any set $\mathcal{E} \subseteq FiltL$ directed with respect to $\subseteq$ such that $\uparrow\{a\} \subseteq \bigcup\mathcal{E}$. Then $a \in \bigcup\mathcal{E}$ by definition of $\uparrow\{a\}$ and hence $a \in E$ for some $E \in \mathcal{E}$. By upclosedness of E, $\uparrow\{a\} \subseteq E$, and hence $\uparrow\{a\}$ is a compact element of (Filt $L, \subseteq$). □

4.1.9 Lemma *Let $F \in (FiltL, \subseteq)$. Then $\{\uparrow\{x\} \mid x \in F\}$ is a directed subset of $(FiltL, \subseteq)$.*
Proof: Take any $x_1, x_2 \in F$. Now $x_1 \wedge x_2 \leq x_1$ and $x_1 \wedge x_2 \leq x_2$, so $\uparrow\{x_1\} \subseteq \uparrow\{x_1 \wedge x_2\}$ and $\uparrow\{x_2\} \subseteq \uparrow\{x_1 \wedge x_2\}$. Since F is a filter, $x_1 \wedge x_2 \in F$, and hence $\{\uparrow\{x\} \mid x \in F\}$ is directed with respect to $\subseteq$. □

4.1.10 Lemma *Every compact element of $(FiltL, \subseteq)$ is of the form $\uparrow\{a\}$ for some $a \in L$.*
Proof: Let $E \in Filt(L)$ be a compact element. By definition of $\uparrow\{e\}$, $e \in \uparrow\{e\}$ for every $e \in E$, and thus $E \subseteq \bigcup\{\uparrow\{e\} \mid e \in E\}$. Now by Lemma **4.1.9**, $\{\uparrow\{e\} \mid e \in E\}$ is directed and its union is its least upper bound. By compactness of E, $E \subseteq \uparrow\{e\}$ for some $e \in E$. By the upclosedness of the filter E, $e \in E$ and $e \leq a$ implies $a \in E$ for all $a \in L$. Thus $\uparrow\{e\} \subseteq E$. So $E = \uparrow\{e\}$ for some $e \in L$, as required to complete the proof. □

The principal filters of L form a basis for $(FiltL, \subseteq)$ which means:

4.1.11 Theorem *$(FiltL, \subseteq)$ is an algebraic dcpo.*
Proof: By Lemmas **4.1.8, 4.1.10**, $Filt(L)^\circ = \{\uparrow\{a\} \mid a \in L\}$. Let $F \in Filt(L)$. Then $\{H \mid H \subseteq F$ and H is a compact element$\}$ is directed since $\uparrow\{x\} \subseteq F$ iff $x \in F$ for all $x \in L$ and by Lemma **4.1.9**. Also $F \subseteq \bigcup\{\uparrow\{x\} \mid x \in F\}$ and by upclosedness of F, $\bigcup\{\uparrow\{x\} \mid x \in F\} \subseteq F$. Thus $F = \bigcup\{H \mid H \subseteq F$ and H is a compact element$\}$. □

4.1.12 Corollary *If L is countable then $(FiltL, \subseteq)$ is an ω-algebraic dcpo.* □

Therefore, the filter completion $(FiltL, \subseteq)$ of a bounded distributive lattice is a domain. Since elements of a bounded distributive lattice correspond one-one to principal filters of the lattice, a bounded distributive lattice can be recovered from the compact elements of its filter completion.

For the construction of a domain from a Priestley information system, we use a method similar to that of Scott (given in Section 1.6) for defining an information domain from a Scott information system.

Let $(H, \vdash, \wedge, \vee, \bot, \top)$ be a Priestley information system. Then:

4.1.13 Definition A non-empty subset $S \subseteq H$ is called an *element generated by the information system* if for $x, y \in H$,

(a) $x \in S$ and $x \vdash y$ then $y \in S$ (upclosed with respect to $\vdash$);

(b) $x, y \in S$ implies $(\exists z \in S)[z \vdash x, y]$ (directed with respect to $\vdash$).

Note that I have dropped condition **3.2.5** (c) in the definition of a state of a Priestley information system. (This is the terminology of [Sco82].)

Let $\mathcal{D}H$ denote the set of all elements of the Priestley information system $(H, \vdash, \wedge, \vee, \bot, \top)$. The poset $(\mathcal{D}H, \subseteq)$ I shall call (using the terminology of [Sco82]) the *information domain associated with* $(H, \vdash, \wedge, \vee, \bot, \top)$. In the following sequence of results I show it is indeed a domain.

4.1.14 Lemma $(\mathcal{D}H, \subseteq)$ *is a dcpo.*

Proof: Let $\mathcal{E} \subseteq \mathcal{D}H$ be a directed set with respect to set inclusion. To show that $\mathcal{E}$ has a least upper bound in $\mathcal{D}H$, it suffices to show that the union of elements in $\mathcal{E}$ is an element of $(H, \vdash, \wedge, \vee, \bot, \top)$. $\bigcup \mathcal{E}$ is upclosed (with respect to $\vdash$) since each $E \in \mathcal{E}$ is. For directedness, take any $x, y \in \bigcup \mathcal{E}$. Then $x \in E_1$ and $y \in E_2$ for some $E_1, E_2 \in \mathcal{E}$. Since $\mathcal{E}$ is directed, $E_1, E_2 \subseteq E_3$ for some $E_3 \subseteq \mathcal{E}$. Hence $x, y \in E_3$, and since E_3 is directed $z \vdash x, y$ for some $z \in E_3 \subseteq \bigcup \mathcal{E}$. Therefore $(\mathcal{D}H, \subseteq)$ is a cpo. The upclosedness with respect to $\vdash$ of elements in $\mathcal{E}$ ensures that $\top \in E$ for each $E \in \mathcal{E}$. Since $\{1\}$ is upclosed and directed with respect to $\vdash$, this set defines the bottom of $(\mathcal{D}H, \subseteq)$. □

For any $a \in H$, denote the set $\{y \in H \mid a \vdash y\}$ by $\uparrow_{\vdash}\{a\}$. Then:

4.1.15 Lemma *For any* $a \in H$, $\uparrow_{\vdash}\{a\}$ *is a compact element of* $\mathcal{D}H$.

Proof: First, I show that $\uparrow_{\vdash}\{a\} \in \mathcal{D}H$. Take $x, y \in H$ with $x \in \uparrow_{\vdash}\{a\}$ and $x \vdash y$. Then $a \vdash y$ and $x \vdash y$, so $a \vdash y$ (by transitivity of $\vdash$), hence $y \in \uparrow_{\vdash}\{a\}$. Second, I show that $\uparrow_{\vdash}\{a\}$ is compact. Take any directed set $\mathcal{E} \subseteq \mathcal{D}H$ with $\uparrow_{\vdash}\{a\} \subseteq \bigcup \mathcal{E}$. Then $a \in \bigcup \mathcal{E}$ and hence $a \in E$ for some $E \in \mathcal{E}$. Since $E \in \mathcal{D}H$, $\uparrow_{\vdash}\{a\} \subseteq E$, and hence $\uparrow_{\vdash}\{a\}$ is a compact element of $(\mathcal{D}H, \subseteq)$. □

4.1.16 Lemma *Let $F \in \mathcal{D}H - \{\emptyset\}$. Then $\{\uparrow_{\vdash}\{a\} \mid a \in F\}$ is directed with respect to $\subseteq$.*

Proof: Take $a_1, a_2 \in F$. Then by **4.1.13**(b), $z \vdash a_1$ and $z \vdash a_2$ for some $z \in F$. Then for every $y \in H$, $a_1 \vdash y$ implies $z \vdash y$ and $a_2 \vdash y$ implies $z \vdash y$. Thus $\uparrow_{\vdash}\{a_1\} \subseteq \uparrow_{\vdash}\{z\}$ and $\uparrow_{\vdash}\{a_2\} \subseteq \uparrow_{\vdash}\{z\}$. So since $z \in F$, $\{\uparrow_{\vdash}\{a\} \mid a \in F\}$ is directed with respect to $\subseteq$. □

4.1.17 Lemma *Every compact element of $(\mathcal{D}H, \subseteq)$ is of the form $\uparrow_{\vdash}\{a\}$ for some $a \in H$.*

Proof: Let $E \in \mathcal{D}H$ be a compact element of $\mathcal{D}H$. Since $\vdash$ is a preorder, $e \vdash e$ for all $e \in E$, so $e \in \uparrow_{\vdash}\{e\}$ and hence $E \subseteq \bigcup\{\uparrow_{\vdash}\{e\} \mid e \in E\}$. By Lemma **4.1.16**, $\{\uparrow_{\vdash}\{e\} \mid e \in E\}$ is directed and its union is its least upper bound. By compactness of E, $E \subseteq \uparrow_{\vdash}\{e\}$ for some $e \in E$. By **4.1.13**(a), $\uparrow_{\vdash}\{e\} \subseteq E$. So $E = \uparrow_{\vdash}\{e\}$ for some $e \in H$ as required to complete the proof. □

4.1.18 Theorem *$(\mathcal{D}H, \subseteq)$ is an algebraic dcpo.*

Proof: By Lemmas **4.1.16** and **4.1.17**, $\{\uparrow_{\vdash}\{a\} \mid a \in H\}$ is the set of compact elements of $(\mathcal{D}H, \subseteq)$. Let $F \in \mathcal{D}H$. Then $\{H \mid H \subseteq F \text{ and } H \text{ is compact}\}$ is directed since $\uparrow_{\vdash}\{x\} \subseteq F$ iff $x \in F$ for all $x \in H$. Also $F \subseteq \bigcup\{\uparrow_{\vdash}\{x\} \mid x \in F\}$ and by **1.4.2** (a), $\bigcup\{\uparrow_{\vdash}\{x\} \mid x \in F\} \subseteq F$. Thus $F = \bigcup\{H \mid H \subseteq F \text{ and } H \text{ is compact}\}$. □

Therefore, the information domain associated with a Priestley information system is a domain.

Recall (Theorem **3.4.1**) that a bounded distributive lattice is a Priestley information system. As a consequence of Theorem **4.1.19**, the filter completion of a bounded distributive lattice is its information domain.

4.1.19 Theorem *Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice, and let S be a non-empty subset of L. Then S is an element of L as a Priestley information system iff S is a prime filter of L as bounded distributive lattice.*

Proof: Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. Then $(L, \vdash, \vee, \wedge, 0, 1)$ with $\vdash$ as $\leq$, is a Priestley information system. Let S be a state of L as a Priestley information system. Then:

(a) S is an upper set with respect to $\vdash$.

(b) Suppose $x, y \in S$. Then for some $z \in S$, $z \vdash x, y$ (by **3.2.5**(b)); so $z \vdash x \wedge y$ (by **3.2.1**(c)), and hence by **3.2.5**(a) $x \wedge y \in S$. So S is closed under finite meets.

So S is a prime filter of L as a bounded distributive lattice. Conversely, suppose S is a prime filter of L as a bounded distributive lattice. Then:

(a) $x \in S$ and $x \vdash y$ implies $y \in S$ since S is an upper set with respect to $\vdash$.

(b) Suppose $x, y \in S$. Then $x \wedge y \in S$ (since S is closed under finite meets). But $x \wedge y \vdash x, y$ by **3.2.1**(c), so there is some $z \in S$ (namely $x \wedge y$) with $z \vdash x, y$.

So S is an element of L as a Priestley information system. □

Having completed three different constructions of domains, I now consider how they are related. The following theorem establishes an isomorphism between the domain of a Priestley space and the domain of its lattice of clopen upper sets. (It is one of the consequences of Priestley duality (Theorem **1.5.7**.).)

4.1.20 Theorem *Let $(X, \leq, \tau)$ be a Priestley space. Then $(CX, \supseteq)$ is isomorphic to $(Filt\mathcal{U}X, \subseteq)$.*

Proof: A filter $\mathcal{F}$ of $(\mathcal{U}X, \subseteq)$ is determined by its members, which are clopen upper subsets of X. Define

$$\Phi(\mathcal{F}) = \bigcap\{A \in \mathcal{U}X \mid A \in \mathcal{F}\}.$$

for $\mathcal{F} \in Filt\mathcal{U}X$. Then $\Phi(\mathcal{F})$, being an intersection of clopen upper sets, is a closed upper set. Conversely, for every closed upper set $W \in CX$, define

$$\Psi(W) = \{U \in \mathcal{U}X \mid U \supseteq W\}.$$

Then it is easily checked that $\Psi(W)$ is a filter of $(\mathcal{U}X, \subseteq)$. The mappings $\Phi : (Filt\mathcal{U}X, \subseteq) \to (CX, \supseteq)$ and $\Psi : (CX, \supseteq) \to (Filt\mathcal{U}X, \subseteq)$ are well-defined and order-preserving. I now show that they are inverses of each other: $\Phi \circ \Psi(W) = W$ for all $W \in CX$ and $\Psi \circ \Phi(\mathcal{F}) = \mathcal{F}$ for all $\mathcal{F} \in Filt\mathcal{U}X$. By definition, for every $W \in CX$ $\Phi \circ \Psi(W) \subseteq W$, and for every $\mathcal{F} \in Filt\mathcal{U}X$ $\Psi \circ \Phi(\mathcal{F}) \supseteq \mathcal{F}$. Conversely, take any $a \in W$. Choose any $A \in \epsilon_X(a)$ with $W \subseteq A$. Then $A \in \Psi(W)$ and $a \in A$. Hence $a \in \bigcap\{A \in \mathcal{U}X \mid A \supseteq W\}$; so $a \in \Phi \circ \Psi(W)$. Now take any $A \in \Psi \circ \Phi(\mathcal{F})$. Then $A \in \mathcal{U}X$ and $\bigcap\{B \mid B \in \mathcal{F}\} \subseteq A$. By compactness, the intersection of finitely many B's in $\mathcal{F}$ is contained in A; hence $A \in \mathcal{F}$ since $\mathcal{F}$ is a filter. Therefore, $(CX, \supseteq)$ is isomorphic to $(Filt\mathcal{U}X, \subseteq)$. □

As a corollary to Theorems **3.4.1** and **4.1.20**, there is a bijective correspondence between the domain of a Priestley space and the domain of its Priestley information system of clopen upper sets.

4.1.21 Corolloary *Let $(X, \leq, \tau)$ be a Priestley space. Then $(CX, \supseteq)$ is isomorphic to $(\mathcal{D}\mathcal{U}X, \subseteq)$.* □

Similar order-isomorphisms can be established when starting with a bounded distributive lattice or with a Priestley information system.

The results of this Section allow us to start at any one of the corners of the program semantics triangle and then to construct a domain using one of constructions. Moreover, this domain is related to the domain of the other two presentations derived from the chosen presentation. In this Chapter I shall take the relational presentation of program semantics based on Priestley spaces (bottom corner of the program semantics triangle) as the basic presentation. Using the interpretations in Chapter 2 of a Priestley space, the elements of this domain are refutable positive properties and the ordering means one property is 'better' than another if fewer variables have to be inspected to establish its absence from a state. The next step is to obtain a representation of programs based on this domain related to the relational, algebraic and logical representations of programs introduced in Chapters 2 and 3.

4.2 Programs as mappings to refutable positive properties

In this Section I introduce a representation of programs as *Priestley state transformers* mapping states to refutable positive properties of states related to the relational representation of programs as opposite Priestley relations. The results of this Section then unify the order-theoretic and relational representations of programs based on Priestley spaces.

As in previous chapters the state space is a Priestley space $(X, \leq, \tau)$. Due to the possible nontermination and nondeterminism of programs, viewing a program as a function from the state space into itself (mapping initial states to final states) is too simplistic; instead here I think of a program as a function from states to sets of states mapping any initial state to the set of all its possible final states from that state. In Section 2.5 it was discovered that the set of final states of a program from a given initial state is not just a set of states but it has a topological structure which captures the nondeterministic and terminating behaviour of the program from the initial state. This suggests a representation of programs as functions from the state space to certain subsets of the state space, rather than to all subsets. I consider closed upper sets of a Priestley space for three reasons. First, as explained in Section 2.5, closed upper sets capture the idea of bounded nondeterminism and possible nontermination from an initial state. Second, as explained in Section 4.1, closed upper sets with respect to superset inclusion form a domain — the domain of the Priestley space. Third, closed upper sets provide a connection between the representation proposed here and the relational representation of programs as opposite Priestley relations in

Section 2.6.

With these ideas, I define a notion of *Priestley state transformer* based on the domain of the Priestley space $(X, \leq, \tau)$.

4.2.1 Definition A mapping $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ is called a *Priestley state transformer* if it is continuous with respect to the upper topology on $\mathcal{C}X$.

(For a definition of continuity with respect to a topology see Appendix 2.) Let Pdomain+ST denote the collection of Priestley domains with Priestley state transformers.

As explained in Sections 2.3 and 2.4, closed upper sets can be interpreted as refutable positive properties. Therefore Priestley state transformers are mappings from states to refutable positive properties, and hence describe the input-output behaviour of boundedly nondeterministic (possibly nonterminating) programs.

The correspondence between Priestley state transformers and opposite Priestley relations is almost trivial, given that binary relations $R \subseteq X \times X$, multifunctions $R : X \to X$ with $R(x) \subseteq X$ (for $x \in X$) and single-valued functions $R : X \to \mathcal{P}(X)$ are in a sense that same thing. The details are as follows.

Every opposite Priestley relation $R \subseteq (X, \leq, \tau) \times (X, \leq, \tau)$ is a multifunction $R : (X, \leq, \tau) \to (X, \leq, \tau)$ which assigns to each $x \in X$ the closed upper set $R(x) = \{y \mid (x, y) \in R\}$. By condition **1.6.6** (b) on opposite Priestley relations, this multifunction is continuous in the sense that $\mathcal{U}R(V) \in \mathcal{U}X$ whenever $V \in \mathcal{U}X$, where $\mathcal{U}R(V) = \{x \mid R(x) \subseteq V\}$. To such a multifunction there corresponds a single-valued function $F_R : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ defined by:

$$F_R(x) = R(x) \text{ for } x \in X.$$

Then:

4.2.2 Theorem *For every opposite Priestley relation R over $(X, \leq, \tau)$, the function $F_R : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ is a Priestley state transformer.*
Proof: By condition **1.6.6** (a) on opposite Priestley relations, F_R is well defined. Moreover F_R is continuous with respect to the upper topology because for every basic open set B_O, $F_R^{-1}(B_O) = \{x \mid F_R(x) \in B_O\} = \{x \mid F_R(x) \subseteq O\} = \mathcal{U}R(O)$. But O is a clopen upper set, and hence by **1.6.6**(b) so is $\mathcal{U}R(O)$. □

To every Priestley state transformer $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ there corresponds a binary relation $R_F \subseteq (X, \leq, \tau) \times (X, \leq, \tau)$ defined by:

$$(x, y) \in R_F \text{ iff } y \in F(x) \text{ for } x, y \in X.$$

Then:

4.2.3 Theorem *For every Priestley state transformer* $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$, *the relation* $R_F \subseteq (X, \leq, \tau) \times (X, \leq, \tau)$ *is an opposite Priestley relation.*
Proof: By definition, $R_F(x)$ is a closed upper set for every $x \in X$. Take any clopen upper set V. Then $(R_F) :_\forall (V) = \{x \mid R_F(x) \subseteq V\} = \{x \mid F(x) \subseteq V\} = F^{-1}(V)$ which is a clopen upper set by continuity of the F with respect to the upper topology on $\mathcal{C}X$. □

Let RS denote the set of all opposite Priestley relations over $(X, \leq, \tau)$, and ST the set of all Priestley state transformers $F_R : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$. Then with the above translations I show there is an order-isomorphism between RS and ST.

4.2.4 Theorem ST *is order-isomorphic to* RS.
Proof: Define a mapping $\eta : \mathsf{ST} \to \mathsf{RS}$ by $\eta(F) = R_F$ for every $F \in \mathsf{ST}$, and a mapping $\nu : \mathsf{RS} \to \mathsf{ST}$ by $\nu(R) = F_r$ for every $R \in \mathsf{RS}$. Then $\eta \circ \nu(R) = R$ for every $R \in \mathsf{RS}$, and $\nu \circ \eta(F) = F$ for every $F \in \mathsf{ST}$ since: $\eta \circ \nu(R) = \eta(F_R) = R_{F_R} = \{(x, y) \mid y \in F_R(x)\} = \{(x, y) \mid y \in R(x)\} = R$, and $\nu \circ \eta(F)(x) = \nu(R_F)(x) = F_{R_F}(x) = \{y \mid (x, y) \in R_f\} = \{y \mid y \in F(x)\} = F(x)$. Finally monotonicity of η and ν follows from Lemma **4.1.1**. Therefore, η is an order-isomorphism between ST and RS. □

The results of this Section allow us to represent programs either as opposite Priestley relations or as Priestley state transformers without losing any information about the input-output behaviour of the programs.

4.3 Denotational semantics and predicate transformer semantics

In this Section I relate Priestley state transformers and Priestley predicate transformers, and thereby unify the order-theoretic and algebraic representations of programs based on Priestley spaces.

Let PT denote the set of all Priestley predicate transformers on $(\mathcal{U}X, \subseteq)$, and, as before, ST the set of all Priestley state transformers from $(X, \leq, \tau)$ to $(\mathcal{C}X, \supseteq)$.

Given a Priestley state transformer $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$, a mapping $F^+ : (\mathcal{U}X, \subseteq) \to (\mathcal{U}X, \subseteq)$ can be defined by:

$$F^+(V) = \{x \in X \mid F(x) \subseteq V\} \text{ for } V \in \mathcal{U}X.$$

The function F^+ is called the *upper inverse* of F [Smy83]. (This is the notation of [Ber59].) Then:

4.3.1 Theorem *F^+ is a Priestley predicate transformer.*
Proof: By continuity of F, $F^+(U) \in \mathcal{U}X$ for every $U \in \mathcal{U}X$. Moreover, F^+ is multiplicative because for every $\mathcal{U} \subseteq \mathcal{U}X$ we have:

$$\begin{aligned} F^+(\bigcap \mathcal{U}) &= \{x \in X \mid F(x) \subseteq \bigcap \mathcal{U}\} \\ &= \{x \in X \mid (\forall U \in \mathcal{U})[F(x) \subseteq U]\} \\ &= \bigcap \{F^+(U) \mid U \in \mathcal{U}\} \end{aligned}$$

Therefore, F^+ is a Priestley predicate transformer. □

Given a Priestley predicate transformer $f : (\mathcal{U}X, \subseteq) \to (\mathcal{U}X, \subseteq)$, a mapping $f^* : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ can be defined by:

$$f^*(x) = \bigcap \{V \in \mathcal{U}X \mid x \in f(V)\} \text{ for } x \in X.$$

Using a generalisation of Plotkin's stability lemma ([Plo80] Lemma 5.6), it follows that:

4.3.2 Theorem *f^* is a Priestley state transformer.*
Proof: Since the intersection of clopen upper sets is a closed upper set, $f^*(x) \in \mathcal{C}X$ for every $x \in X$. Moreover, f^* is continuous because for every $U \in \mathcal{U}X$ we have:

$$\begin{aligned} f^{*+}(U) &= \{x \in X \mid f^*(x) \subseteq U\} \\ &= \{x \in X \mid \bigcap \{P \mid x \in f(P)\} \subseteq U\} \\ &=^{\dagger} \{x \in X \mid x \in f(U)\} \\ &= f(U) \in \mathcal{U}X \end{aligned}$$

where $\subseteq^{\dagger}$ holds by monotonicity of f. For the converse, suppose $x \in f(U)$. Then, by the multiplicativity of f, $x \in f(U) \cap f(\bigcap \{P \mid x \in f(P)\}) = f(U \cap \bigcap \{P \mid x \in f(P)\}) = f(\bigcap \{P \mid x \in f(P)\})$ since $U \in \{P \mid x \in f(P)\}$. Thus $\bigcap \{P \mid x \in f(P)\} \subseteq U$. Therefore $f^{*+} = f$, and hence f^* is continuous. □

4.3.3 Theorem

(a) *For a Priestley predicate transformer* $f : (\mathcal{U}X, \subseteq) \to (\mathcal{U}X, \subseteq)$, $f^{*+} = f$.

(b) *For a Priestley state transformer* $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$, $F^{+*} = F$.

Proof: Part (a) is proved in Theorem **4.3.2**. For part (b), take any $x \in X$. Then

$$\begin{aligned} F^{+*}(x) &= \bigcap\{U \in \mathcal{U}X \mid x \in F^{+}(U)\} \\ &= \bigcap\{U \in \mathcal{U}X \mid F(x) \subseteq U\} \\ &=^{\dagger} F(x) \end{aligned}$$

where $\supseteq^{\dagger}$ is clear, and the converse holds because $y \in \bigcap\{U \mid F(x) \subseteq U\}$ implies $y \in U$ for every clopen upper set U such that $F(x) \subseteq U$. But $F(x) \in \mathcal{C}X$, so $F(x) = \bigcap\mathcal{U}$ for some $\mathcal{U} \subseteq \mathcal{U}X$, hence $y \in \bigcap\mathcal{U} = F(x)$. □

As a consequence of the above translations there is an order-isomorphism between PT and ST.

4.3.4 Theorem ST *is order-isomorphic to* PT.
Proof: Define a mapping $\eta : \mathtt{ST} \to \mathtt{PT}$ by $\eta(f) = f^{+}$ for every $f \in \mathtt{ST}$, and a mapping $\nu : \mathtt{PT} \to \mathtt{ST}$ by $\nu(F) = F^{*}$ for every $F \in \mathtt{PT}$. Then by Theorem **4.3.3**(a) $\eta \circ \nu(F) = F$ for every $F \in \mathtt{PT}$, and by Theorem **4.3.3**(b) $\nu \circ \eta(f) = f$ for every $f \in \mathtt{ST}$. Finally monotonicity of η and ν follows from Lemma **4.1.1**. Therefore, η is an order-isomorphism between ST and PT. □

Therefore a Priestley state transformer from $(X, \leq, \tau)$ to $(\mathcal{C}X, \supseteq)$ can be viewed as a Priestley predicate transformer between $(\mathcal{U}X, \subseteq)$, and *vice versa*.

4.4 Denotational semantics and information systems

In this Section I relate Priestley state transformers and Priestley information-respecting relations, and thereby unify the order-theoretic and logical representations of programs based on Priestley spaces.

Let IR denote the set of all Priestley information-respecting relation over $(\mathcal{U}X, \subseteq)$, and, as before, ST the set of all Priestley state transformers from $(X, \leq, \tau)$ to $(\mathcal{C}X, \supseteq)$.

Given a Priestley state transformer $F : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$, a relation $R_F \subseteq (\mathcal{U}X, \subseteq) \times (\mathcal{U}X, \subseteq)$ can be defined by:

$$U R_F V \text{ iff} (\forall u \in U)[F(u) \subseteq V] \text{ for } U, V \in \mathcal{U}X.$$

Then:

4.4.1 Theorem *R_F is a Priestley information-respecting relation.*
Proof: I show that $R_F = \mathcal{I}F^+$ where F^+ is the Priestley predicate transformer obtained from F (Theorem **4.3.1**). For $U, V \in \mathcal{U}X$, $U R_F V$ iff $U \subseteq F^+(V)$ iff $U \mathcal{I} F^+ V$. Thus by Theorem **3.4.2**, R_F is a Priestley information-respecting relation. □

Given a Priestley information-respecting relation $R \subseteq (\mathcal{U}X, \subseteq) \times (\mathcal{U}X, \subseteq)$, a mapping $F_R : (X, \leq, \tau) \to (\mathcal{C}X, \supseteq)$ can be defined by:

$$F_R(x) = \bigcap \{V \in \mathcal{U}X \mid (\exists U \in \mathcal{U}X)[x \in U \text{ and } U R V]\} \text{ for } x \in X.$$

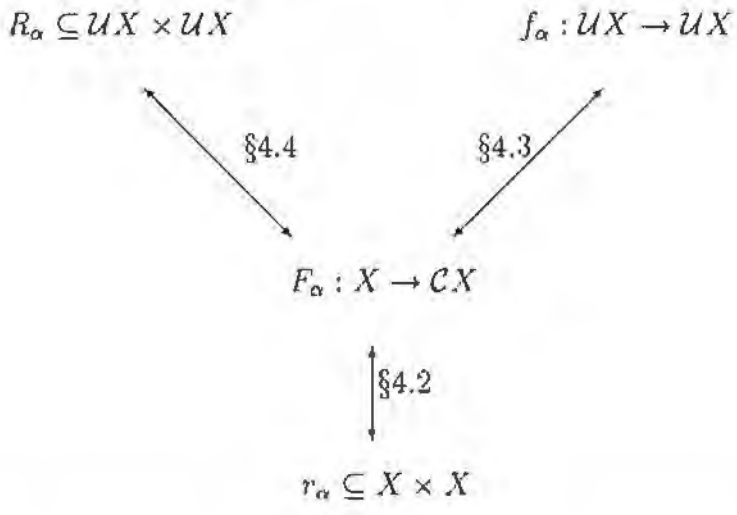
Then:

4.4.2 Theorem *F_R is a Priestley state transformer.*
Proof: I show that $F_R = (R_{\equiv})^*$ where $R_{\equiv}$ is the Priestley predicate transformer obtained from R (Theorem **3.4.5**). Take any $x \in X$. Then $F_R(x) = (R_{\equiv})^*(x)$ since $x \in R_{\equiv}(V)$ iff for some $U \in \mathcal{U}X$, $x \in U$ and $U R V$. Thus, by Theorem **4.3.1**, F_R is a Priestley state transformer. □

With the above translations I show that there is an order-isomorphism between IR and ST.

4.4.3 Theorem ST *is order-isomorphic to* IR.
Proof: Define a mapping $\eta : \mathtt{ST} \to \mathtt{IR}$ by $\eta(F) = R_F$ for every $F \in \mathtt{ST}$, and a mapping $\nu : \mathtt{IR} \to \mathtt{ST}$ by $\nu(R) = F_R$ for every $R \in \mathtt{IS}$. Then $\nu \circ \eta(F) = ((R_F)_{\equiv})^* = ((\mathcal{I}F^+)_{\equiv})^* = (F^+)^* = F$ for every $F \in \mathtt{ST}$, and $\eta \circ \nu(R) = \mathcal{I}(F_R^+) = \mathcal{I}((R_{\equiv})^{*+}) = \mathcal{I}(R_{\equiv}) = R$ for every $R \in \mathtt{IS}$. Finally monotonicity of η and ν follows from Lemma **4.1.1**. Therefore, η is an order-isomorphism between ST and IR. □

With the translations presented in Sections 4.2 to 4.4, I have established the arrows in the following diagram:



That is, I have shown that the same (up to isomorphism) order-theoretic presentation of program semantics can be obtained from the relational presentation of program semantics based on Priestley spaces, and its derived algebraic and logical presentations.

Chapter 5

The Program Semantics Triangle

In this thesis I have provided a positive answer to the conjecture raised in the Preface that the triangle of logic-algebra-semantics serves as a paradigm for unifying four versions of program semantics. The objective is achieved by using Priestley spaces to model the underlying state spaces.

In the **relational presentation**, Pspace+RS (Chapter 2),

- a state, taken as the basic notion, is represented as a point of a *Priestley space*,
- an observable positive property is constructed as the (clopen upper) set of states in which it is true, and
- a program is represented as an *opposite Priestley relation* between states.

In the **algebraic presentation**, Dlat+PT (Chapter 2),

- a state is constructed as the set (prime filter) of observable positive properties true in it,
- an observable positive property, taken as the basic notion, is represented as an element of a *bounded distributive lattice*, and
- a program is represented as a *Priestley predicate transformer* between observable positive properties.

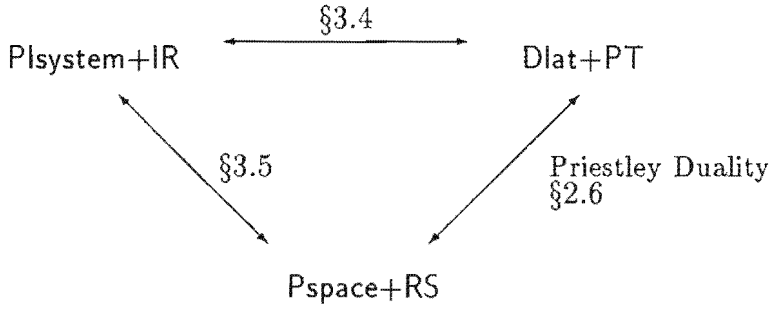
In the **logical presentation**, PIsystem+IR (Chapter 3),

- a state is constructed as the (deductively closed, decidable and consistent) set of observable positive properties true in it,
- an observable positive property, taken as the basic notion, is a proposition of a *Priestley information system*, and
- a program is represented as a *Priestley information-respecting relation* between observable positive properties.

In the **order-theoretic presentation**, Pdomain+ST (Chapter 4),

- a state, taken as the basic notion, is a point of a Priestley space,
- a refutable positive property, is an element of the *upper power space of a Priestley space*, and
- a program is represented as a *Priestley state transformer* from states to refutable positive properties of states.

The technical results of Chapters 1, 2 and 3 can be summarised by saying that: *Priestley spaces with opposite Priestley relations correspond on the logical side to Priestley information systems with Priestley information-respecting relations, and on the algebraic side to bounded distributive lattices with Priestley predicate transformers. Moreover, the logical and the algebraic structures also correspond to each other.* In other words, taking these structures as presentations of program semantics, we have shown that a relational, an algebraic and a logical presentation of program semantics stand to each other in the following formal version of the logic-algebra-semantics triangle presented in the Preface on p viii:



This allows us to take any one of these presentations as the basic presentation, and then derive each of the others using the translations provided in Chapters 1, 2, and 3.

For example, start with the relational presentation of program semantics (the bottom of the triangle). Then the state space is a Priestley space $(X, \leq, \tau)$ and a program α is represented as an opposite Priestley relation r_α between states in X. This relational presentation can be lifted one level to an algebraic presentation (top right-hand corner of the triangle) or to a logical presentation (top left-hand corner of the triangle). I consider each in turn.

First, an algebraic presentation can be obtained by forming the lattice $(\mathcal{U}X, \subseteq)$ of observable positive properties of states, and defining for each program α a Priestley predicate transformer $f_\alpha : \mathcal{U}X \to \mathcal{U}X$ by:

$$f_\alpha(Q) = \{s \in X \mid (s,t) \in r_\alpha \Rightarrow t \in Q\} \text{ for } Q \in \mathcal{U}X.$$

From this algebraic representation, a logical presentation (top left-hand corner of the triangle) can be obtained by forming the Priestley information system $(\mathcal{U}X, \subseteq, \cap, \cup, \emptyset, X)$ of observable positive properties and by defining for each program α a Priestley information-respecting relation $R_\alpha \subseteq \mathcal{U}X \times \mathcal{U}X$ by:

$$(P,Q) \in R_\alpha \text{ iff } P \subseteq f_\alpha(Q) \text{ for } P,Q \in \mathcal{U}X.$$

Then (an isomorphic copy of) the original relational presentation can be recovered by taking the state space to be the collection of states of the Priestley information system and by defining a relation r_α between states in $\mathcal{S}\mathcal{U}X$ by:

$$(\mathcal{S},\mathcal{T}) \in r_\alpha \text{ iff } (\forall Q \in \mathcal{U}X)\ [F_\alpha(Q) \in \mathcal{S} \Rightarrow Q \in \mathcal{T}] \text{ for } \mathcal{S},\mathcal{T} \in \mathcal{S}\mathcal{U}X$$

where $F_\alpha(Q) = \bigcup\{P \mid (P,Q) \in R_\alpha\}$ for $Q \in \mathcal{U}X$.

Second, from the relational presentation r_α, a logical presentation can be obtained by forming the Priestley information system $(\mathcal{U}X, \subseteq, \cap, \cup, \emptyset, X)$ of observable positive properties and by defining for each program α, a Priestley information-respecting relation $R_\alpha \subseteq \mathcal{U}X \times \mathcal{U}X$ by:

$$PR_\alpha Q \text{ iff } (\forall s,t \in X)[(s \in P \text{ and } (s,t) \in r_\alpha) \text{ imply } t \in Q] \text{ for } P,Q \in \mathcal{U}X.$$

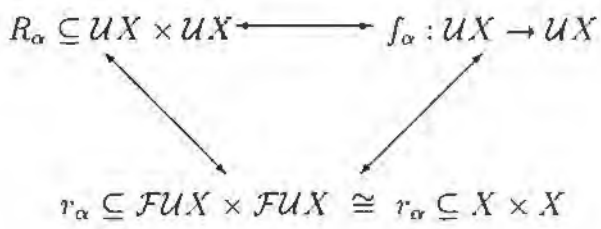
From this logical representation, an algebraic representation (top right-hand corner of the triangle) can be obtained by defining a Priestley predicate transformer $f_\alpha : \mathcal{U}X \to \mathcal{U}X$ between observable positive properties by:

$$f_\alpha(Q) = \bigcup\{P \mid (P,Q) \in R_\alpha\} \text{ for } Q \in \mathcal{U}X.$$

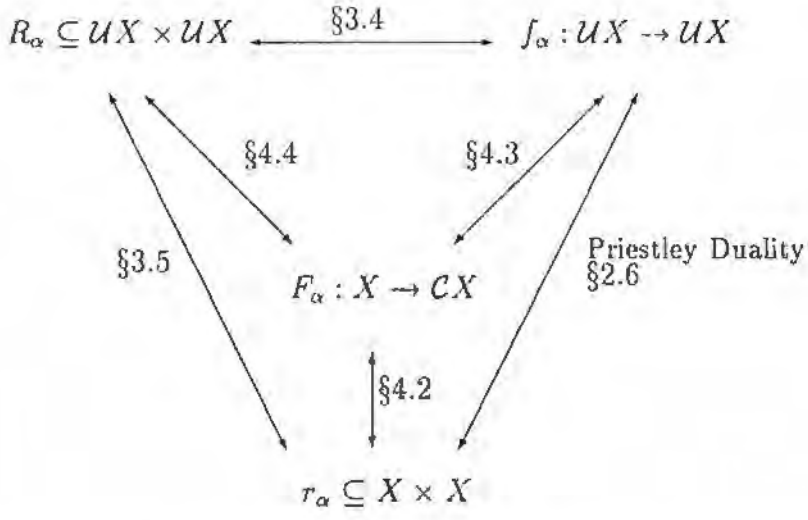
Then (an isomorphic copy of) the original relational presentation can be recovered by taking states to be sets (prime filters) of observable positive properties and by defining for each program α a relation r_α between states in $\mathcal{F}\mathcal{U}X$ by:

$$(\mathcal{S},\mathcal{T}) \in r_\alpha \text{ iff } (\forall Q \in \mathcal{U}X)[f_\alpha(Q) \in \mathcal{S} \Rightarrow Q \in \mathcal{T}] \text{ for } \mathcal{S},\mathcal{T} \in \mathcal{F}\mathcal{U}X.$$

These two traversals of the program semantics triangle can be pictured as follows:



Into this triangle we can in addition fit the *order-theoretic* presentation of programs, as is shown by the results of Chapter 4, and obtain the following diagram:



Take, once again, the relational presentation as the basic presentation. Then the order-theoretic presentation is obtained by forming the domain $(\mathcal{C}X, \supseteq)$ of refutable positive properties, and defining for each program α a Priestley state transformer $F_\alpha : X \to \mathcal{C}X$ by:

$$F_\alpha(x) = r_\alpha(x) \text{ for } x \in X.$$

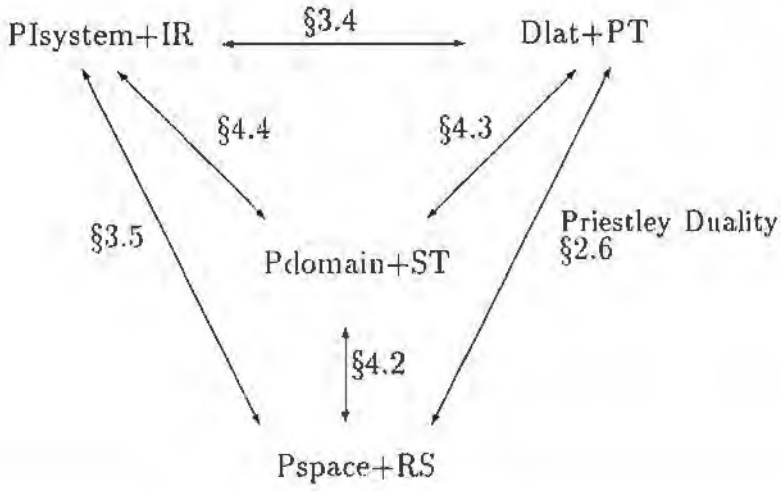
This presentation can also be obtained from the algebraic and logical presentations of program semantics derived from the relational presentation. Take the filter completion $(\mathcal{FU}X, \subseteq)$ of the algebraic presentation $(\mathcal{U}X, \subseteq)$ and define for each Priestley predicate transformer $f_\alpha : \mathcal{U}X \to \mathcal{U}X$ a Priestley state transformer $F_\alpha : X \to \mathcal{C}X$ by:

$$F_\alpha(x) = \bigcap\{V \in \mathcal{U}X \mid x \in f_\alpha(V)\} \text{ for } x \in X.$$

Take the information domain $(\mathcal{DU}X, \subseteq)$ of the logical presentation $(\mathcal{U}X, \subseteq, \cap, \cup, \emptyset, X)$ and define for each Priestley information-respecting relation R_α over $\mathcal{U}X$, a Priestley state transformer $F_\alpha : X \to \mathcal{C}X$ by:

$$F_\alpha(x) = \bigcap\{V \in \mathcal{U}X \mid (\exists U \in \mathcal{U}X)[x \in U \text{ and } URV]\} \text{ for } x \in X.$$

Therefore, taking the above-mentioned structures as presentations of program semantics, I have shown that a relational, an algebraic, a logical and an order-theoretical presentation of program semantics stand to each other in the following formal version of the *program semantics triangle*, presented in the Preface on p ix.



The unifying framework for program semantics presented in this thesis has three theoretical benefits. First, it uses a natural topological setting to unify the relational, algebraic, logical and order-theoretical presentations of program semantics. In doing so it capitalises on a firm mathematical foundation (Priestley duality) and brings this to bear on disparate versions of program semantics. Second, the framework captures and clarifies our intuitions about program semantics. Specifically, there are natural topological characterisations of computational concepts such as observability, nondeterminism, termination and predicate transformer. Third, it allows, through the use of Priestley spaces, the innovation of separating the logical and information-theoretic aspects of a state space. The logical aspect of a state space, arising from the topology, deals with the question of whether an observer can always determine the truth of a claim in finite time. The information-theoretic aspect, arising from the order, deals with the question of what information can be determined. I give effect to this idea by distinguishing two kinds of properties: a topological kind (that is, verifiable, refutable and observable) and an order-theoretic kind (that is, positive and negative).

The practical relevance of the results of this thesis invites further investigation. But it is hoped that the insight gained in setting up the unifying framework will help to clarify the foundations of Computing Science, and to determine the appropriate area of application of each version of program semantics.

The idea of relating various versions of program semantics is not entirely new, but the existing literature is fragmentary. A connection between Dijkstra's predicate transformers and Smyth's powerdomain is established by Plotkin [Plo80]

for *flat domains*. Smyth [Smy83] placed this result in a topological context by using *Scott domains* as state spaces. Later [BoK93, BoK94] showed that these results also hold in a more general topological context by modelling state spaces as arbitrary T_0-spaces. An even more general view of state spaces is taken in [BJK95] where *generalised topological spaces* (closed under arbitrary unions and arbitrary intersections) are used. Priestley spaces are more general structures than domains, but not as general as generalised topological spaces. Therefore the relationship between Dlat+PT and Pdom+ST is more general than that established for domains [Plo80, Smy83, BoK93], but it is generalised by a similar result for T_0-spaces and for generalised topological spaces. Recall that Priestley information systems generalised Scott information systems, so the relationships between Plsystem+IR and the other presentations generalises those established in domain theory. The relationship between Dlat+PT and Pspace+RS generalises the relationship between predicate transformer semantics and relational semantics established in [ReB95] in the same way that Priestley duality generalises Jónsson/Tarski duality.

The following challenges remain:

(a) to extend the unifying framework presented so as to deal with *operational semantics* such as that of (e.g.) Hoare [Hoa95];

(b) to investigate the applicability of the framework to the specification and construction of programs, by finding a more realistic example of a state-space-as-a-Priestley-space than the T^ω-example;

(c) to extend the separation of logical and information-theoretic aspects of state spaces, by investigating the *power domains* of the state space and presenting the power orders as orderings of *growth of information*;

(d) to find the relationships between the presentations of program semantics based on more general computational models than Priestley spaces (such as, for example, the generalised topological spaces of [BJK95]);

(e) to investigate whether Priestley spaces yield results comparable to those for Scott domains (e.g.) Cartesian closedness and solutions to domain equations;

(f) to investigate whether Priestley spaces can deal with the semantics of concurrent and/or recursive programs;

(g) to determine a logical language for talking about the notions of 'information' and 'finite observation' in a single unified setting.

Perhaps further questions will occur to the reader.

. . . ——— . . .

Bibliography

[Abr91] Abramsky, S. [1991]. Domain theory in logical form. *Annals of Pure and Applied Logic* **51**. p 1–77.

[AlS85] Alpern, B. and F.B. Schneider. [1985]. Defining liveness. *Information Processing Letters* **21**. p 181–185.

[Bac80] Back, R.J.R. [1980]. Semantics of unbounded nondeterminism. In: *Proceedings of ICALP 80.* Lecture Notes in Computer Science 85. Berlin: Springer-Verlag. p 51–63.

[Bac81] Back, R.J.R. [1981]. On correct refinement of programs. *Journal of Computer and System Sciences* **23**. p 49–68.

[BaW89] Back, R.J.R. and J. von Wright. [1989]. Refinement calculus, Part I: Sequential programs. In: *Proceedings of the 1989 REX Workshop for Refinement of Distributed Systems.* Lecture Notes in Computer Science 430. Springer-Verlag: Berlin.

[BaW93] ———. [1993]. Statement inversion and strongest postcondition. *Science of Computer Programming* **20** (3). p 223–251.

[Ber59] Berge, C. [1959]. *Espaces Topologiques: Fonctions Multivoques.* Dunod, Paris.

[BeS71] Bell, J.L. and A.R. Slomson. [1971]. *Models and Ultraproducts.* 2nd Edition. Amsterdam: North-Holland.

[Bir33] Birkhoff, G. [1933]. On the combination of subalgebras. *Proceedings of the Cambridge Philosophical Society* **29**. p 441–464.

[Bir48] ———. [1948]. *Lattice Theory.* 2nd Edition. American Mathematical Society Colloquium Publications Volume XXV. New York: American Mathematical Society.

[BlJ72] Blyth, T.S. and M.F. Janowitz. [1972]. *Residuation Theory.* Germany: Pergamon Press.

[BoK93] Bonsangue, M. and J.N. Kok. [1993]. Isomorphisms between state and predicate transformers. In: *Mathematical Foundations of Computer Science 1993, Gdansk, Poland.* (AM Borzyszkowski and S Sokolowoski (eds)). Lecture Notes in Computer Science 711. p 310–310.

[BoK94] ——. [1994]. Relating multifunctions and predicate transformers through closure operations. In: *Proceedings of TACS'94, Sendai, Japan.* (M Hagiya and JC Mitchell). Lecture Notes in Computer Science 789. p 822–843.

[BJK95] Bonsangue, M., B. Jacobs and J.N. Kok. [1995]. Duality beyond sober spaces: Topological spaces and observable frames. *Theoretical Computer Science* **151** (1). (Special Issue dedicated to AMAST/MASK Workshop on Topology and Completion in Semantics, Chartres, France, 1993. (P Gastin and JM Rutten (eds)).) p 79–124.

[Bri93] Brink, C. [1993]. Power structures. *Algebra Universalis* **30**. p 177–216.

[BGO95] Brink, C., D.M. Gabbay and H.J. Ohlbach. [1995]. Towards automating duality. *Computers and Mathematics with Applications* **29** (2). p 73–90.

[BrG95] Brink, C. and J. Goslett. Propositional logic, powerdomains and information. Submitted to *Studia Logica* (Special Commemorative Issue for Helena Rasiowa).

[BuS84] Bull, R. and K. Segerberg. [1984]. Basic modal logic. In: *Handbook of Philosophical Logic* II. Dordrecht: Reidel. p 1 – 88.

[BuS81] Burris, S. and H.P. Sankappanavar. [1981]. *A Course in Universal Algebra.* Berlin: Springer-Verlag.

[CLP91] Cignoli, R., S. Laflace and A. Petrovich. [1991]. Remarks on Priestley duality for distributive lattices. *Order* **8**. p 299–315.

[DaP90] Davey, B. and H.A. Priestley. [1990]. *Introduction to Lattices and Order.* Cambridge: Cambridge University Press.

[Dij75] Dijkstra, E.W. [1975]. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM* **18** (8). p 453–458.

[Dij76] ——. [1976]. *A Discipline of Programming.* Englewood Cliffs, New Jersey: Prentice-Hall.

[DiS90] Dijkstra, E.W. and C.S. Scholten. [1990]. *Predicate Calculus and Program Semantics.* New York: Springer-Verlag.

[DOR94] Demri, S., E. Orlowska and I. Rewitzky. [1994]. Towards reasoning about Hoare relations. *Annals of Mathematics and Artificial Intelligence* **12**. p 265–289.

[DrG90] Droste, M. and R. Göbel. [1990]. Nondeterministic information systems and their domains. *Theoretical Computer Science* **75**. p 289–309.

[EdS93] Edalaat, A. and M.B. Smyth. [1993]. I-Categories as a framework for solving domain equations. *Theoretical Computer Science* **115**. p 77–106.

[GHK80] Gierz, G., K.H. Hofman, K. Keimel, J.D. Lawson, M. Mislove and D.S. Scott. [1980]. *A Compendium of Continuous Lattices.* Berlin: Springer-Verlag.

[Gol84] Goldbaltt, R. [1984]. *Topoi.* Revised edition. Studies in Logic 98. Amsterdam: North-Holland.

[Gol89] ———. [1989]. Varieties of complex algebras. *Annals of Pure and Applied Logic* **44**. p 173–242.

[Gor79] Gordon, M.J.C. [1979]. *The Denotational Description of Programming Languages: An Introduction.* New York: Springer-Verlag.

[Gri81] Gries, D. [1981]. *The Science of Programming.* Berlin: Springer-Verlag.

[Gue80] Guerreiro, P. [1980]. A relational model for nondeterministic programs and predicate transformers. In: *Proceedings of the 4th International Colloquium on Automata, Languages and Programming, Paris.* Lecture Notes in Computer Science 83. Berlin: Springer-Verlag. p 136–146.

[Gue82] ———. [1982]. Another characterisation of weakest preconditions. *Lecture Notes in Computer Science* 137. Springer-Verlag: Berlin. p 164–177.

[Gun90] Gunter, C.A. [1990]. Relating total and partial correctness interpretations of non-deterministic programs. In: *Principles of Programming Languages* (POPL'90) ACM: New York. p 306–319.

[GuS90] Gunter, C.A. and D.S. Scott. [1990]. Semantic domains. In: *Handbook of Theoretical Computer Science* Volume B, Formal Models and Semantics. (J. van Leeuwen (ed)). Amsterdam: Elsevier Science Publishers B.V. p 633–674.

[Hal74] Halmos, P.R. [1974]. *Lectures on Boolean Algebras.* Springer-Verlag: New York.

[Han83] Hansoul, G. [1983]. A duality for Boolean algebras with operators. *Algebra Universalis* **17**. p 37–49.

[Hoa69] Hoare, C.A.R. [1969]. An axiomatic basis for computer programming. *Communications of the ACM* **12** (10). p 576–583.

[Hoa87] Hoare, C.A.R., I.J. Hayes, He Jifeng, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey and B.A. Sufrin. [1987]. Laws of programming. *Communications of the ACM* **30**. p 672–686.

[Hoa95] Hoare, C.A.R. [1995]. Unified theories of programming. Manuscript. 58 pages.

[HoL74] Hoare, C.A.R. and P.E. Lauer. [1974]. Consistent and complementary formal theories of the semantics of programming languages. *Acta Informatica* **3**. p 135–153.

[HuC68] Hughes, G.E. and M.J. Cresswell. [1968]. *An Introduction to Modal Logic*. Methuen and Co.

[JaG85] Jacobs, D. and D. Gries. [1985]. General correctness: A unification of partial and total correctness. *Acta Informatica* **22**. p 67–83.

[JoT51] Jónsson, B. and A. Tarski. [1951]. Boolean algebras with operators I. *American Journal of Mathematics* **73**. p 891–939.

[JoT52] ———. [1952]. Boolean algebras with operators II. *American Journal of Mathematics* **74**. p 127–167.

[Joh82] Johnstone, P.T. [1982]. *Stone Spaces*. Cambridge: Cambridge University Press.

[Kei77] Keisler, H.J. [1977]. Fundamentals of model theory. In: *Handbook of Mathematical Logic*. (J. Bairwise (ed)). Amsterdam: North-Holland. p 47–103.

[Kel42] Kelley, J.L. [1942]. Hyperspaces of a continuum. *Transactions of the American Mathematical Society* **52**. p 23–36.

[Kel55] ———. [1955]. *General Topology*. Princeton: Van Nostrand.

[Kri63] Kripke, S. [1963]. Semantic analysis of modal logic. *Zentrum für Mathematik, Logik und Grundl. Mathematik* **9**. p 67–96.

[Kwi91] Kwiatkowska, M.Z. [1991]. On topological characterisation of behavioural properties. In: *Topology and Category Theory in Computer Sciences. Proceedings of the Oxford Topology Symposium 1989*. (GM Reed, AW Roscoe and RF Wachter (eds)). Oxford: Oxford University Press. p 153 - 177.

[Law87] Lawson, J.D. [1987]. The versatile continuous order. In: *Proceedings of the Third Workshop on Mathematical Foundations of Programming Languages Semantics.* (M Main, A Melton, M Mislove, D Schmidt (eds)). Lecture Notes in Computer Science 298. Berlin: Springer-Verlag. p 134–160.

[Mai87] Main, M.G. [1987]. A powerdomain primer. *Bulletin of the European Association for Theoretical Computer Science* **33**. p 115–147.

[MaC80] Majester-Cederbaum, M.E. [1980]. A simple relation between relational and predicate transformer semantics for nondeterministic programs. *Information Processing Letters* **4**. p 190–192.

[Mic51] Michael, E. [1951]. Topologies on spaces of subsets. *Transactions of the American Mathemetical Society* **71**. p 152–182.

[MiS87] Mirkowska, G. and A. Salwicki. [1987]. *Algorithmic Logic.* Dordrecht: Reidel.

[Mos90] Mosses. [1990]. Denotational Semantics. In: *Handbook of Theoretical Computer Science* Volume B, Formal Models and Semantics. (J. van Leeuwen (ed)). Amsterdam: Elsevier Science Publishers B.V. p 577–631.

[Nel89] Nelson, G. [1989]. A generalisation of Dijkstra's calculus. *ACM Transactions of Programming Languages and Systems* **11** (4). p 517–561.

[Par81] Parikh, R. [1981]. Propositional Dynamic Logics of programs: A survey. In: *Logic of Programs* (E. Engeler (ed)). Lecture Notes in Computer Science 125. Berlin: Springer-Verlag. p 102–144.

[Par83] ———. [1983]. Some applications of topology to program semantics. *Mathematical Systems Theory* **16**. p 111–131.

[Plo76] Plotkin, G.D. [1976]. A powerdomain construction. *SIAM Journal of Computing* **5**. p 452–487.

[Plo78] ———. [1978]. T^{ω} as a universal domain. *Journal of Computer and System Sciences* **17**. p 209–236.

[Plo80] ———. [1980]. Dijkstra's predicate transformers and Smyth's powerdomains. In: *Proceedings of the 1979 Copenhagen Winter School in Abstract Software Specifications.* (D. Bjorner (ed)). Lecture Notes in Computer Science 86. p 525–553.

[Pra76] Pratt, V.R. [1976]. Semantical considerations on Floyd-Hoare logic. In *Proceedings on the 17th IEEE Symposium on Foundations of Computer Science.* p 109–121.

[Pri70] Priestley, H.A. [1970]. Representation of distributive lattices by means of ordered Stone spaces. *Bulletin of the London Mathematical Society* **2**. p 186–190.

[Pri72] ———. [1972]. Ordered topological spaces and the representation of distributive lattices. In: *Proceedings of the London Mathematical Society* **24** (3). p 507–530.

[Pri84] ———. [1984]. Ordered sets and duality for distributive lattices. *Annals of Discrete Mathematics* **23**. p 39–60.

[Pri85] ———. [1985]. Algebraic lattices as dual spaces of distributive lattices. In: *Continuous Lattices and their Applications*. (RE Hoffmann and Kh Hoffmann (eds).) Lecture Notes in Pure and Applied Mathematics 101. New York: Marcell Dekker. p 237–249.

[Pri94a] ———. [1994a]. Spectral spaces. *Journal of Pure and Applied Algebra* **94**. p 101–114.

[Pri94b] ———. [1994b]. Intrinsic spectral topologies. Eighth Summer Conference at Queen's College (G Itzkowitz *et al* (ed)). *Annals of the New York Academy of Sciences* **728**. p 78–95.

[RaS63] Rasiowa, H. and R. Sikorski. [1963]. *The Mathematics of Metamathematics*. Warsaw: PWN-Polish Scientific Publishers.

[Ras74] Rasiowa, H. [1974]. *An algebraic approach to non-classical logics*. Studies in Logic and the Foundation of Mathematics. Vol 78. Amsterdam: North-Holland.

[ReB95] Rewitzky, I. and C. Brink. [1995]. Predicate transformers as power operations. *Formal Aspects in Computing* **7** (2). p 169–182.

[SaV88] Sambin, G. and V. Vaccaro. [1988]. Topology and duality in modal logic. *Annals of Pure and Applied Logic* **37** (3). p 249–296.

[Sch86] Schmidt, D.A. [1986]. *Denotational Semantics*. Massachusetts: Allyn and Bacon.

[Sco70] Scott, D.S. [1970]. Outline of a Mathematical Theory of Computations. In: *Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems*. p 169–176.

[Sco72] ———. [1972]. Lattice Theory, Data Types and Semantics. In: *Formal Semantics of Programming Languages* (R Rustin (ed)). Courant Computer Science Symposium 2. Prentice-Hall. p 65–106.

[Sco76] ———. [1976]. Data types as lattices. *Siam Journal of Computing* **5**. p 522–587.

[Sco82] Scott, D.S. [1982]. Domains for denotational semantics. In: *Proceedings of ICALP 9.* (M. Nielsen and E.M. Schmidt (eds)). Lecture Notes in Computer Science 140. p 577–613.

[Smy78] Smyth, M.B. [1978]. Power domains. *Journal of Computer and System Sciences* **16**. p 23–36.

[Smy83] ———. [1983]. Power domains and predicate transformers: a topological view. In: *Proceedings of ICALP 10.* (J. Diaz (ed)). Lecture Notes in Computer Science 154. p 662–675.

[Smy92] ———. [1992]. Topology. In: *Handbook of Logic in Computer Science.* Volume 1. (S Abramsky, DM Gabbay, TSE Maibaum (eds)). Oxford: Oxford University Press.

[Sto36] Stone, M.H. [1936]. The theory of representations for Boolean algebras. *Transactions of the American Mathematical Society* **40**. p 37–111.

[Sto37] ———. [1937]. Topological representations of distributive lattices and Brouwerian logics. *Casopis Pro Potování Mathematiky* **67**. p 1–25.

[Sto77] Stoy, J.E. [1977]. *Denotational Semantics. The Scott-Strachey Approach to Programming Language Theory.* Cambridge, Massachusetts: MIT Press.

[Str66] Strachey, C. [1966]. Towards a formal semantics. In *IFIP TC2 Working Conferences on Formal Language Description Languages for Computer Programming.* North-Holland: Amsterdam. p 198–220.

[Tar35] Tarski, A. [1935]. Zur Grundlegung der Boole'schen Algebra I. *Fundamenta Mathematicae* **24**. p 177–198.

[Tar41] ———. [1941]. On the calculus of relations. *Journal of Symbolic Logic* **6**. p 73–89.

[Tar44] ———. [1944]. The algebra of topology. *Annals of Mathematics* **45**. p 141–191.

[Tar56] ———. [1956]. *Logic, Semantics, Metamathematics.* Oxford: Oxford University Press. p 320–341.

[Tho74] Thomason, S.K. [1974]. An incompleteness theorem in modal logic. *Theoria* **40**. p 30–34.

[vaB84] van Benthem, J. [1984]. Modal correspondence theory. In: *Handbook of Philosophical Logic* Volume II. (P. Gabbay and F. Guenthner (eds)). Dordrecht: D. Reidel. (Synthese Library Vol 166). p 167–247.

[Vic89] Vickers, S. [1989]. *Topology via Logic.* Cambridge: Cambridge University Press.

[Vie21] Vietoris, L. [1921]. Stetige Mengen. *Monatshefte für Mathematik und Physik* **31**. p 173–204.

[Vie22a] ———. [1922a]. Bereiche zweiter ordnung. *Monatshefte für Mathematik und Physik* **32**. p 258–280.

[Vie22b] ———. [1922b]. Kontinua zweiter ordnung. *Monatshefte für Mathematik und Physik* **32**. p 49–62.

[Wan77] Wand, M. [1977]. A characterisation of weakest preconditions. *Journal of Computer and System Sciences* **15**. p 209–212.

[Wil70] Willard, S. [1970]. *General Topology*. Addison-Wesley Publishing Company.

[Wri57] Wright, F.B. [1957]. Some remarks on Boolean duality. *Portugal Math.* **16**. p 109–117.

[Zha94] Zhang, G-Q. [1994]. A representation of SFP. *Information and Computation* **110**. p 233–263.

Appendix 1

Some Notions from Lattice Theory

This Appendix provides a concise summary of notions and results from Lattice Theory used in this thesis. Standard references include [GHK80, BuS81, DaP90]

Let X be a non-empty set. The set-theoretic *complement* of a subset $A \subseteq X$ is the set $\{x \in X \mid a \notin A\}$ and will be denoted by $\overline{A}$.

A *preorder* (or *quasi-order*) on X is a binary relation $\leq$ on X which is reflexive ($\forall x \in X, x \leq x$), and transitive ($\forall x, y, z \in X, x \leq y$ and $y \leq z$ imply $x \leq z$). A set together with a pre-order is called a *pre-ordered set*. If, in addition, $\leq$ is antisymmetric ($\forall x, y \in X$, $x \leq y$ and $y \leq x$ imply $x = y$) then the pre-order is called a *partial order*. A set together with a partial order is called a *partially ordered set* or *poset*.

Let $(X, \leq)$ be a poset. For any subset $A \subseteq X$, the set $\{x \in X \mid (\exists a \in A)[a \leq x]\}$ is called the *upclosure* of A, denoted $\uparrow A$, and the set $\{x \in X \mid (\exists a \in A)[x \leq a]\}$ is called the *downclosure* of A, denoted $\downarrow A$. A subset $A \subseteq X$ is called an *upper set* if $A = \uparrow A$ (that is, if $x \in A$ and $x \leq y$ together imply $y \in A$). Dually, a subset $A \subseteq X$ is called a *lower set* if $A = \downarrow A$ (that is, if $x \in A$ and $y \leq x$ together imply $y \in A$). Upper sets and lower sets are also related by set-theoretic complementation. That is, for any $A \subseteq X$, A is an upper set if $\overline{A}$ is a lower set. For any $x, y \in X$, $x \leq y$ iff $\downarrow\{x\} \subseteq \downarrow\{y\}$ iff $\uparrow\{y\} \subseteq \uparrow\{x\}$.

Many properties of ordered sets are expressed in terms of least upper bounds and lower bounds. Let A be a subset of a poset $(X, \leq)$. Then $x \in X$ is an *upper bound* for A if for every $a \in A$, $a \leq x$. An upper bound x of A is the *least upper bound* (abbreviated as lub) if $x \leq y$ for any other upper bound y of A. Dually, we may define the concepts *lower bound* and *greatest lower bound* (abbreviated as glb). The lub of a set A is denoted $\bigvee A$; the glb is denoted $\bigwedge A$. An element $y \in X$ is called the *least element* or *bottom* of X if for all $x \in X$, $y \leq x$. The least element of $(X, \leq)$ is usually denoted $\bot_X$ (or $\bot$ when no confusion arises). Dually, the *greatest element* or *top* of X is an element $z \in X$ such that for all $x \in X$, $x \leq z$. The greatest element of $(X, \leq)$ is usually denoted by $\top_X$ (or $\top$ when no confusion arises).

A poset $(L, \leq)$ is a *bounded lattice* if L has a least element and a greatest element, and for every $x, y \in L$ $\text{glb}\{x, y\}$ $(= x \wedge y)$ and $\text{lub}\{x, y\}$ $(= x \vee y)$ exist. For any lattice, $\bigvee A$ and $\bigwedge A$ exist for finite $A \subseteq L$. Alternatively, a bounded lattice can be defined algebraically to be a set L together with a unit 1, a zero 0 and two binary operations $\wedge$ and $\vee$ (meet and join respectively) satisfying idempotence, commutativity, associativity, absorptive and identity laws. The equivalence of the two characterisations is shown using the following connection between $\leq$, $\wedge$,

and $\vee$:

$$a \leq b \text{ iff } a \wedge b = a \text{ iff } a \vee b = b.$$

So a lattice can be thought of either as a special kind of poset or as an algebraic structure.

A bounded distributive lattice is a bounded lattice which satisfies a distributivity law. A bounded distributive lattice $(L, \vee, \wedge, 0, 1)$ in which every element $a \in L$ has a (necessarily unique) complement $a' \in L$ (that is, $a \wedge a' = 0$ and $a \vee a' = 1$) is called a *Boolean algebra*.

Certain subsets of a bounded distributive lattice are important in Priestley duality namely, prime filters. Let $(L, \vee, \wedge, 0, 1)$ be a bounded distributive lattice. A subset $F \subseteq L$ is called a *filter* if F is a non-empty upper set closed under finite meets. A filter is called a *prime filter* if for all $x, y \in L$, $x \vee y \in F$ implies $x \in F$ or $y \in F$. A *principal filter* is a filter arising from a single element by upclosure. Dually, a subset $I \subseteq L$ is called an *ideal* if I is a non-empty lower set closed under finite joins. An ideal is called a *prime ideal* if for all $x, y \in L$, $x \wedge y \in I$ implies $x \in I$ or $y \in I$. A *principal ideal* is a filter arising from a single element by downclosure. Prime filters and prime ideals are related by set-theoretic complementation.

Let L and K be lattice. The a map $f : L \to K$ is called a *meet homomorphism* if f preserves the zero 0 and finite meets, and is called a *join homomorphism* if f preserves the unit 1 and finite joins. A map $f : L \to K$ is called a *homomorphism* (or, for emphasis, a *lattice homomorphism*) if f is a meet and join homomorphism. A bijective homomorphism is a *(lattice-)isomorphism*. If $f : L \to K$ is a one-one homomorphism, then the sublattice $f(L)$ of K is isomorphic to L, and f is called an *embedding* (of L into K).

A *bounded distributive lattice with unary operators* is a bounded distributive lattice $(L, \vee, \wedge, 0, 1)$ with a set of meet or join homomorphisms over L.

Appendix 2

Some Notions from Topology

This Appendix outlines some topological definitions and results used in this thesis. (Standard references on topology include [Kel55, Wil70]).

Let X be a non-empty set. A collection τ of subsets of X is called a *topology* on X if τ is closed under the formation of arbitrary unions and finite intersections. A *topological space* consists of a non-empty set X and a topology τ on X. The sets in τ are the *open sets* of the topological space (X, τ); the elements of X are called the *points*. A *closed set* in a topological space is a set whose (set-theoretic) complement in X is an open set. The collection of closed sets is closed under arbitrary intersections and finite unions. Sets which are both open and closed are called *clopen sets*. So any topological space has at least two clopen sets — the empty set and the full space X (since they are respectively the union and intersection of the empty class of sets). A sufficient condition for the existence of other clopen sets is a notion of *total disconnectedness*. That is, a topological space (X, τ) is *totally disconnected* if for any two distinct elements $x, y \in X$, there exists a clopen subset of X containing x and not y. A topology τ on a non-empty set X can be specified without describing every open set, by specifying a family $\mathcal{S}$ of subsets of X to be open. If $\mathcal{S}$ is closed under finite intersections, then the unions of sets in $\mathcal{S}$ is the topology on X, and $\mathcal{S}$ is called a *basis* for τ. If $\mathcal{S}$ is not closed under finite intersections, the topology τ on X can be obtained by first forming the collection $\mathcal{B}$ of all finite intersections of elements from $\mathcal{S}$, and then the union of sets in $\mathcal{B}$ is the topology on X. In this case, $\mathcal{S}$ is called a *subbasis* for τ, and $\mathcal{B}$ a *basis* for τ.

Let (X, τ) be a topological space. A collection $\mathcal{U} = \{U_i\}_{i \in I}$ of open subsets of X is called an *open cover* of X if each point in X belongs to at least one U_i, that is, if $\bigcup_{i \in I} U_i = X$. A subcollection of an open cover which is itself an open cover is called an *open subcover*. If every open cover of X has a finite open subcover then (X, τ) is said to be *compact*. A useful tool in proving compactness of a topological space is the *Alexander Subbasis theorem* (for a proof of which see [DaP90]).

Theorem *Let (X, τ) be a topological space with subbasis $\mathcal{S}$. Then X is compact if every open cover of X by members of $\mathcal{S}$ has a finite open subcover.* □

Let (X, τ_x) and (Y, τ_y) be topological spaces and let $f : X \to Y$ be a map. Then the following conditions are equivalent:

- $f^{-1}(U)$ is open in X whenever U is open in Y;
- $f^{-1}(U)$ is closed in X whenever U is closed in Y; and

- $f^{-1}(U)$ is open in X for every $U \in \mathcal{S}$, where $\mathcal{S}$ is a basis or a subbasis for τ_y.

The map is called *continuous* if it satisfies any of these conditions. Then a map $f : X \to Y$ from X to Y is called a *homeomorphism* if f is one-one, onto and continuous and f^{-1} is also continuous. In this case, X and Y are said to be *homeomorphic*. So a homeomorphism is a map which preserves X set-theoretically and topologically.

Any topological space (X, τ) has a non-trivial order induced by the topology, its *specialisation order*, defined by

$$x \leq_\tau y \text{ iff } \forall O \in \tau \; x \in O \text{ implies } y \in O \text{ for } x, y \in X.$$

Index of Notation

Programs

Execution relational semantics

Predicate transformer semantics

States

Priestley spaces

Priestley relations

Distributive lattices

Homomorphisms

Information systems

Information-respecting relations

Domains

State transformers