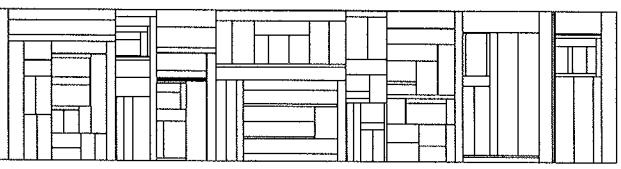
ISSN 0105-8517

Logics of Domains*)

Guo Qiang Zhang

DAIMI PB – 298
December 1989

COMPUTER SCIENCE DEPARTMENT
AARHUS UNIVERSITY
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark



*) A dissertation submitted for the degree of Doctor of Philosophy at the University of Cambridge, May 1989

Abstract

This dissertation studies the logical aspects of domains as used in the denotational semantics of programming languages. Frameworks of domain logics are introduced which serve as basic tools for the systematic derivation of proof systems from the denotational semantics of programming languages. The proof systems so derived are guaranteed to agree with the denotational semantics in the sense that the denotation of any program coincides with the set of assertions true of it.

The study focuses on two frameworks for denotational semantics: the **SFP** domains, and the less standard, but important, category of dI-domains with stable functions.

An extended form of Scott's information systems are introduced to represent **SFP** objects. They provide better understanding of the structure of finite elements and open sets of domains. These systems generalise to a logic of **SFP** which uses inequational formulae to axiomatise entailment and non-entailment of open-set assertions. Soundness, completeness, and expressiveness results of the logic are obtained, and possible applications are investigated. A mu-calculus of Scott domains is introduced to extend the expressive power of the assertion language.

Special kinds of open sets called stable neighbourhoods are introduced and shown to determine stable functions in a similar sense to that in which Scott-open sets determine continuous functions. Properties and constructions of stable neighbourhoods on various categories of dI-domains are investigated. Logical frameworks for Girard's coherent spaces and Berry's dI-domains are given in which assertions are interpreted as stable neighbourhoods. Various soundness, completeness, and expressiveness results are provided.

Acknowledgements

I am grateful to my supervisor, Glynn Winskel, for his guidance, for his encouragement, for his tolerance, and for his personal understanding. His advice and suggestions have had a fundamental influence on the development of this dissertation. I feel I have benefited so much from working with him, which has shaped better my way of thinking. I would like to thank Samson Abramsky, Thierry Coquand, Carl Gunter, Martin Hyland, Mogens Nielsen, and Edmund Robinson for their interest, concern and help during the development of the thesis.

My stay at Cambridge is made possible by a research studentship from Trinity College, for which I am very grateful. Many thanks to the Department of Computer Science of Aarhus University for their hospitality and for providing me a chance for an enjoyable visit.

Declaration

This dissertation is composed by myself and is the result of my own work, under the guidance of my supervisor, Dr Glynn Winskel. This dissertation includes nothing which is the outcome of work done in collaboration.

This dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University. I further state that no part of my dissertation/thesis has already been or is being concurrently submitted for any such degree, diploma or other qualification.

Contents

Chapter 1

Introduction

1.1 Domain Theory, Denotational Semantics [1]

Programming languages are the languages with which to perform computation. They range from the more theoretical languages like $\lambda-$calculi with various evaluation strategies, Milner's CCS [Mi80] and Hoare's CSP [Ho78], to commonly used languages like Pascal, Lisp, and assembler. In the development of high level programming languages people learned the necessity to give precise meanings to the variety of syntaxes. The most successful and well known approach to this problem is due to Scott and Strachey [St64], [ScSt71]. Their approach is based on the idea that the semantics of a programming language should be formally described in terms of a rather small number of basic mathematical constructions on partial orders of information. The mathematical part is called domain theory and the method which uses domain theory to specify the meaning of a programming language is called denotational semantics.

The denotational semantics of a programming language is given by assigning to each piece of program an element in a domain. Due to the possible self applicative nature of some programs, the domains must sometimes have some special properties. Typically, it can be required that a domain be isomorphic to its own function space. This is impossible within the classical set theoretical framework because exponentiation always produces a higher cardinality on non-trivial sets. Scott's idea was to work with, not directly the values and objects of computation, but rather information about computation. In the framework of complete partial orders and continuous functions Scott has shown that it is possible to get non-trivial solutions to equations such as $D \cong [D \to D]$. Now properties and constructions on cpos are well studied to the extent that it is possible to solve recursive equations of domains and use these constructions to give denotational semantics to most programming languages.

A complete partial order (cpo) is a partial order which has a bottom element and

[1] This section provides some historical background on domain theory and denotational semantics. A formal introduction to domain theory is given in Chapter 2.

least upper bounds of directed sets of elements. For two elements x and y, $x \sqsubseteq y$ means the information content of x is contained in the information content of y. The bottom element $\perp$ has, therefore, empty information content. Elements which have finite information content play an important role in the theory. Intuitively those elements have information which can be realised by a computation in finite time. We call them finite elements. As far as computation is concerned it is usually enough to work with $\omega-$algebraic cpos, which have a countable number of finite elements as its base. A function between complete partial orders D and E can be seen as a computation which makes use of some input information in D and produces some output information in E. According to Scott's thesis such a function should be continuous. Continuity requires that the function be monotonic – more information as input yields more information as output – and, if $x_0 \sqsubseteq x_1 \cdots \sqsubseteq x_i \sqsubseteq \cdots$ is a chain in D then the output information for $f(\bigsqcup_i x_i)$ should be possible to be approximated by $f(x_i)$'s as closely as one requires. An important feature of cpos is that the collection of continuous functions between cpos again form a cpo, with the pointwise order; this means that functions themselves are associated with elements of information and so enables the treatment of higher order computations. Unfortunately continuous functions between $\omega-$algebraic cpos need not form an $\omega-$algebraic cpo again. There are subcategories of $\omega-$algebraic cpos with this property however. The most well-known framework is Scott domains, those $\omega-$algebraic cpos which are consistently complete. Scott domains have many nice properties. In particular they are closed under sum, product, lifting, and function space constructions.

When it comes to specifying the semantics of programming languages with parallel constructs, powerdomain constructions are often used. Powerdomains resemble the powerset construction with elements which represent the 'sets' of different courses a nondeterministic computation can follow. However there are several ways to order the elements of a powerdomain: different orderings yield different powerdomain. There are the Hoare powerdomain, the Smyth powerdomain and the Plotkin powerdomain, based on, respectively, three views about what kind of information should be taken into account for nondeterministic processes. There is no problem with the Hoare and the Smyth powerdomain constructions – within the framework of Scott domains they produce a Scott domain from a Scott domain. The Plotkin powerdomain construction, however, does not produce a Scott domain from a Scott domain. This led Plotkin to the discovery [Pl76] of a more general framework called **SFP** objects, which are special kinds of $\omega-$algebraic cpos closed under sum, product, lifting, function space, and the three powerdomain constructions just

mentioned.

There are many other frameworks for denotational semantics [Me88], among which there is the less standard but important category of dI-domains. DI-Domains were discovered by Berry [Be78] from the study of the full-abstraction problem for typed λ-calculi. They are special kinds of Scott domains which have a more operational nature. The functions between dI-domains are stable functions under an order which takes into account the manner in which they compute. DI-domains can be represented as stable event structures of Winskel [Wi86], which casts light on their computational intuition. They can also be represented as information systems [Zh89], which casts light on the logical aspects of dI-domains. DI-domains are becoming more and more popular, mostly due to the fact that they are more elaborate in structure, and to the fact that there is another category of dI-domains discovered by Winskel [Wi86] where constructions like the partially synchronous product can be used to model languages like CCS and CSP quite smoothly. The close connection of this category with some other more concrete models for concurrent processes like Petri nets makes it even more attractive. An important subcategory of dI-domains is the coherent spaces brought into popularity by Girard. They have been recently used to model system F [Gi87a], and for a semantics of linear logic [Gi87b].

1.2 Program Logics, Proof Systems

Program logics[2] are logics which are used to reason about properties of programs. The process of conducting the reasoning is usually expressed as a set of rules, forming a proof system. One of the well-known program logics is the Floyd-Hoare logic (or Hoare logic) [Ho69], with assertions of the form $\{P\}\, C\, \{Q\}$, meaning that if program C starts at a state with property P and terminates, it terminates at a state with property Q. There are many other kinds of program logics, such as temporal logic [Pn77] and dynamic logic [Pr79].

Two of the most important theoretical issues related to a proof system are its soundness and completeness. Informally soundness means each assertion derived from the proof system is correct while completeness requires that every correct assertion be derived in this way. While soundness of Hoare logic is easy to establish completeness is impossible because of Gödel's incompleteness theorem for Peano arithmetic. However Cook in [Co78]

[2]The words 'proof system' and 'program logic' are often used as synonym in the literature.

showed the logic was complete relative to the truths of arithmetic, in the sense that if one were allowed to consult an oracle about truths of arithmetic one can derive all the correct program assertions.

During the past few years, a lot of work has been done to extend the axiomatic method of Hoare to parallel programming languages. One of the well known approaches is due to Owicki and Gries [OwGr76]. The language they considered was the usual sequential one expanded with the statements for parallel composition via shared variables and critical regions. Their proof system used assertions similar to Hoare triples. But the situation became much more complicated. Owicki and Gries needed interference-freedom for the soundness and auxiliary variables for the completeness of their proof system. Interference-freedom was a condition used in the proof rule for parallel composition, which says that processes do not interfere with each other. Recently, Brookes [Br85] presented a novel proof system for the parallel programming language which avoids the use of auxiliary variables and interference-freedom for the proof system. This is achieved by introducing more structures on assertions. The basic idea is to use assertions with a structure of labelled trees, where nodes are attached with predicates. In this way, one can not only describe the input-output property associated with a command, but also keep track of property of the intermediate steps in the execution of the command.

There are enormously many other kinds of proof systems for programming languages in the literature. Often there are even several proof systems for the same programming language. However, many of the proof systems are complicated or ad hoc, or even incorrect, not fulfilling what is claimed of them.

Faced with such a situation we may ask whether there is any principle on which our proof systems can be based. We may ask whether there is any general method that can guide us in building proof systems. Our idea is to look for help from denotational semantics. Our goal is to find general logical frameworks based on domain theory which can be used to derive proof systems like Brookes' but from the denotational semantics. We can expect the derived semantics based proof systems to be guaranteed to agree with the denotational semantics in the sense that the denotation of any program coincides with the set of assertions true of it.

1.3 Logic and Topology

Much recent work ([Pl80], [Sc82], [Sm83], [Wi83]) has been concerned with the relation between logic and domains or topology. The significance of this line of enquiry was eventually made clear by Abramsky ([Ab85], [Ab87]). He showed that it could be regarded as part of a general programme for extracting from a suitable semantics a program logic based on a syntax of types, terms and predicates. Here we briefly review the history of work on predicate transformers, information systems and the logical approaches to domain theory.

Predicate Transformers

Predicate transformers were introduced by Dijkstra [Di76] when he was dealing with the semantics of a simple nondeterministic language of guarded commands. His idea was to regard each program C as a 'predicate transformer' which transforms each assertion (predicate) Q into its weakest precondition $wp(C, Q)$ among P's such that the Hoare triple $\{ P \} C \{ Q \}$ is valid. On the other hand, one can give semantics to the programming language by considering each program as a state transformation function abstracted from a transition system.

Plotkin [Pl80] showed that for Dijkstra's language of guarded commands these two approaches agree. Using Smyth powerdomain he gave an isomorphism between the cpo of predicate transformers and the cpo of state transformation functions for the language of guarded commands.

Later Smyth [Sm83] investigated this connection from a broader topological view. He pointed out that the above connection is nothing more than a special case of the duality between continuous functions $D \xrightarrow{f} E$ and morphisms

$$\Omega(E) \xrightarrow{f^{-1}} \Omega(D).$$

Smyth emphasised the computational significance of topological ideas. A topological space X can be taken as a 'data type', with the open sets as 'properties' and functions between topological spaces as 'computations'.

Information Systems

Information systems were introduced by Scott [Sc82] initially with the intention of making domain theory accessible to a wider audience. In this representation the idea of *information* is made explicit–each element is seen as a collection of *information quanta.* It gives a logical approach to domain theory, in which properties of domains can be derived from assumptions about the entailments between propositions expressing properties of computations.

Intuitively, an information system is a structure describing the logical relations between propositions that can be made about computations. It consists of a set of propositions, a consistency predicate and an entailment relation. An information system determines a family of subsets of propositions called its elements. An element consists of a set of propositions that can be truly made about a single possible computation. Thus it is expected that the propositions should be consistent with each other and if a finite set of propositions is valid for the computation, all their logical consequences should also be valid for it. These elements form a Scott domain under inclusion. On the other hand, given any Scott domain, one can build an information system which determines a Scott domain isomorphic to the original one. Information systems form a category with the approximable mappings [Sc82] as morphisms. It is a category equivalent (in the sense of [Mac71]) to the category of Scott domains. Constructions such as product, sum, and function space have been proposed on information systems [Sc82], [LaWi84], corresponding to those on domains.

Information systems fit Smyth's topological view with propositions regarded as a notation for open sets. In fact, let X be a finite consistent set of propositions of an information system. Let

$$O(X) = \{ x \mid X \subseteq x \}$$

where x's are elements of the information system. $O(X)$ is clearly an open set in the Scott topology of the domain determined by the information system. On the other hand, to get an information system from a Scott domain D, one can take the propositions as the basic open sets, and get the entailment by letting

$$\{ O_1, O_2, \cdots, O_m \} \vdash O$$

iff $\bigcap_i O_i \subseteq O$. The consistency predicate is equally simple:

$$\{ O_1, O_2, \cdots, O_m \} \in Con$$

iff $\bigcap_i O_i \neq \emptyset$. Such a smooth transformation is guaranteed by properties of algebraic cpos, which include, among other things, that Scott topology on an algebraic cpo is both T_0 and sober. T_0 means that objects with the same properties should not be distinguished. Sober means that the domain is completely determined by its lattice of properties (for more detail see [Sm83]).

Domains and Logics

Complete Heyting algebras are complete lattices satisfying the infinite distributive law

$$a \wedge \bigvee S = \bigvee \{ a \wedge s \mid s \in S \}.$$

The category of *frames* has objects complete Heyting algebras, and morphisms functions which preserve finite meets and arbitrary joins. As a special kind of frames one has the set of open sets $\mathbf{\Omega}(D)$ of a topological space D ordered by inclusion; in this case the frame morphisms are precisely those functions

$$f^{-1} : \mathbf{\Omega}(E) \to \mathbf{\Omega}(D)$$

for which $f : D \to E$ is continuous. The category of *locales* is the opposite of the category of frames where the morphism direction is reversed. *Stone dualities* are contravariant equivalences between certain categories of topological spaces and corresponding categories of locales (Johnstone's book [Jo82] is recommended here). They have been proposed as providing the right framework for understanding the relationship between denotational semantics and program logic. Examples of dualities of this kind have already appeared in the literature. Information systems, for example, can be seen as description of locales where the relevant topological spaces consist of the Scott open sets of domains. The *duality* between the category of information systems and the category of Scott topologies of domains is just the *equivalence* between the category of information systems and the category of domains. There are other results of a similar nature, which we discuss now.

In [Wi82], Winskel observed a simple connection between powerdomains and modal assertions. The modalities are $\Box$, for 'inevitably', and $\Diamond$, for 'possibly', to make modal

assertions about nondeterministic computations. He showed that the Smyth powerdomain is built up from assertions about the inevitable behaviour of a process, the Hoare powerdomain is built up from assertions about the possible behaviour of a process, while the Plotkin powerdomain is built up from both kinds of assertions taken together.

Powerdomains are closely related to the classical Vietoris construction on topological spaces. Vietoris showed that for any compact Hausdorff space X, there is a compact Hausdorff topology on the set $\mathbf{K}(X)$ of non-empty, closed subsets of X which coincides with that induced by the Hausdorff metric, provided the topology on X is determined by a metric. However, the relevance to computer science was neglected until [Sm82], where Smyth pointed out the relationship between powerdomains and Vietoris topology. Given a multifunction $\Gamma : X \to Y$, there are three ways to define continuity: the upper semicontinuity, the lower semicontinuity, and both taken together[3]. Correspondingly three topologies were introduced: the upper topology, the lower topology, and the Vietoris topology. The three notions of continuity for a multifunction $\Gamma : X \to Y$ can then be characterised topologically as continuous functions from X to $\mathcal{P}Y$, where $\mathcal{P}Y$ is a suitable topological space constructed out of Y. Smyth showed that by removing the empty set, the specialization orders determined by the upper power space, lower power space, and convex power space are isomorphic to the Smyth powerdomain, the Hoare powerdomain, and the Plotkin powerdomain, respectively.

A more formal approach which takes powerdomains as constructions on locales was given by Robinson [Ro86]. He showed that Vietoris locale gives a Scott topology on the Plotkin powerdomain. The presentation of a domain by means of an algebraic description of its lattice of open sets supplies a way to 'transform' domains into sets of proof rules, such as

$$\Diamond\phi \wedge \Box\psi \leq \Diamond(\,\phi \wedge \psi\,)$$

and

$$\Box(\,\phi \vee \psi\,) \leq \Diamond\phi \vee \Box\psi$$

for the Plotkin powerdomain construction. Such rules have been known to some people, and were implicit in Winskel's work [Wi82], but was brought to the fore by Robinson's paper [Ro86]. It was also made clear that the finite elements of Plotkin powerdomain

[3]A multifunction $\Gamma : X \to Y$ is called upper semicontinuous if $\Gamma^+ : \Omega(Y) \to \Omega(X)$ is a function, where $\Gamma^+(P) = \{\, a \mid \Gamma a \subseteq P \,\}$; and lower semicontinuous if $\Gamma^- : \Omega(Y) \to \Omega(X)$ is a function, where $\Gamma^-(P) = \{\, a \mid \Gamma a \cap P \neq \emptyset \,\}$.

is captured by formulae of the form $(\Box \bigvee \phi_i) \wedge \bigwedge \Diamond \phi_i$. Robinson then related his locale description of powerdomains to Winskel's modal assertion language description.

A notable advance has been made by Abramsky [Ab87] when he showed how program logics could be extracted from domain theory. He developed further the ideas explained above and set a basic framework for relating denotational semantics with program logics in the light of Stone duality. His work provided successful treatments of morphisms, an important step which showed the promise of frameworks of this kind since they can, in a sense, generalise and express dynamic logic and Hoare logic. As an application of his framework, Abramsky [87a] introduced a domain equation for synchronisation trees. This domain equation automatically generates a logic as a special instance of the general framework, out of which Hennessy-Milner logic can be constructed.

1.4 An Overview of the Thesis Work

My thesis work can be viewed as extending the line of research on domains and logics.

Traditionally, there are three approaches to the semantics of programming languages: the operational semantics, the denotational semantics, and the axiomatic semantics. It is important, of course, to understand the relationships between these approaches and ideally one would like to ensure that different approaches end up as formally equivalent in some sense. The connection between operational semantics and denotational semantics has been well studied, as in e.g. full abstraction. However not a great deal of attention has been paid to the relationship between denotational semantics and axiomatic semantics. We raise the following basic issue: for a programming language with a denotational semantics, how can we, if possible, associate it with a proof system so that the axiomatic semantics *agrees* with the denotational semantics in the sense that for programs C, C', $[\![C]\!]_a \subseteq [\![C']\!]_a$ if and only if $[\![C]\!] \sqsubseteq [\![C']\!]$, where $\sqsubseteq$ is the order inherited from the domain, $[\![C]\!]_a$ is the set of assertions true of C?

We have a more ambitious project in mind aiming at finding a general framework to derive proof systems from denotational semantics. The resulting proof systems should be sound and complete and the axiomatic semantics should agree with the denotational semantics. It should be derived automatically, in a uniform manner from the denotational semantics in the following sense. The domains used should determine the style of assertions and their proof rules, while the denotational semantics should provide proof rules

for the programming language constructs. Such proof systems might justifiably be called *semantics based.*

When looking for semantics based proof systems, we were inspired by Brookes' work [Br85]. His system looked promising as a starting point for understanding the relationship between denotational semantics and proof systems. We tried to see whether it was an example of a semantics based proof system (in a more formal sense than that expressed by Brookes). Together with the realisation that the open sets of powerdomains were associated with assertions based on modal operators [Wi82], there was a hope that a more systematic derivation of proof systems like that of Brookes' but derived from the denotational semantics should be possible.

Later we learnt of Abramsky's work ([Ab85], [Ab87]) which contains important insights on how Stone dualities can be successfully applied to computer science. But after more understanding of both Abramsky's and Brookes' work we found that we were not quite in a position to do our job for two reasons. The first was that we had some trouble with Brookes' proof system, which motivated the work of Chapter 3. The second was that Abramsky's framework did not treat **SFP** objects, while Plotkin powerdomain is needed to cope with Brookes' proof system since it is, from our understanding, based on Plotkin's domain of resumptions. This motivated the work of Chapter 5. Chapter 5 of my thesis was finished summer 1988 based on an earlier draft [Zh87]. At the beginning of 1989 we received a recent technical report by Abramsky [Ab88], which includes a treatment of the Plotkin powerdomain, too. Work is done independently of Abramsky's paper [Ab88], and it provides an alternative to his treatment.

SFP domains are one of the most general frameworks for denotational semantics. There is another important framework for denotational semantics, the dI-domains discovered by Berry. DI-domains can be represented as event structures which have a close connection with some other models for concurrency and can be used to give semantics to languages like CCS and CSP. Special kinds of dI-domains – the coherent spaces – have been used by Girard to give semantics to linear logic. These facts make the project of developing a logic of dI-domains very interesting. The logic determined by dI-domains should induce logic for event structures. It might allow us to further explore its connection with linear logic.

However, due to the operational nature of stable functions, we cannot simply borrow the proof rules of domain logic as introduced in Chapter 5 for the logic of dI-domains. We tried to understand the form assertions should take and proof rules for them from the nature of *stable neighbourhood*. While the inverse image functions are not characterised as preserving arbitrary Scott open sets, stable functions are precisely those functions f whose inverse image function f^{-1} preserves a notion of stable neighbourhood. They are considered as proper open sets to interpret assertions of logic of dI-domains. Chapter 8 gives a quite through study of stable neighbourhoods. The results obtained there directly suggest the right kind of proof rules for the logic of dI-domains. Due to the disjoint nature of stable neighbourhoods we do not get a topology. To cope with this phenomenon the logic introduced in Chapter 9 is disjunctive – this is novel. A disjoint assertion language cannot be specified by a simple grammar. We have to employ a mutual recursion between syntactic rules and proof rules to get the full disjoint assertion language. The work of Diers [Di76] and Johnstone [Jo77] is undoubtedly relevant.

As mentioned above, this thesis work focused on developing domain logics for two frameworks of denotational semantics: the **SFP** objects and the dI-domains. Apart from the work described above, in my thesis we have given an information system representation of **SFP** objects in Chapter 4, a $\mu-$ calculus of domain theory in Chapter 6, and some results on different monoidal closed categories of dI-domains in Chapter 7. A brief summary of the contents of each chapter is given below.

Chapter 2 gives a summary of background knowledge needed to understand the work in later chapters. Some new results about the finite elements of the **SFP** objects are given.

Chapter 3 presents Brookes' proof system. A counterexample is given to show that his system is not complete as he claimed in [Br85]. An improved version of his proof system, which avoids the use of labels, is suggested.

Chapter 4 introduces an information system representation for **SFP** objects. We use a Gentzen style entailment $X \vdash Y$, instead of $X \vdash a$. A category of a special kind of such systems is shown to be equivalent to **SFP**. Constructions like the Plotkin powerdomain and the function space are given, as well as a cpo of such information systems to give meanings to recursively defined systems.

Chapter 5 gives a logic of **SFP**. A meta-language for denotational semantics is introduced with general type constructions like sum, product, function space, the three

powerdomains, and recursively defined types. For each type there is a language of open set assertions. Proof systems are given; they use inequational formulae to axiomatise entailment and non-entailment of assertions. Soundness and completeness results are obtained. As an application of the logic, the style of assertions of Brookes proof system is shown to be determined by the logic of Plotkin's domain of resumptions.

Chapter 6 studies the $\mu-$calculus of domain theory to extend the expressive power of the domain logic developed in Chapter 5. A least fixed point operator is added to the assertion language as the $\mu-$ construction, to represent the least fixed point of the function specified by an assertion. A $\mu-$calculus is introduced which includes proof rules for the fixed point induction. The $\mu-$calculus is shown to be sound. Many useful theorems are derived from the proof system, which are used to get the completeness result for a special case – the $\mu-$calculus of integers. As the proof of completeness is achieved by normal form theorems, the expressive power of the integer $\mu-$calculus is immediately clear.

Chapter 7 presents categories of dI-domains and event structures, to give background knowledge for the work of Chapter 8 and 9. We give an introduction to the category of dI-domains of Berry, the category of event structures of Winskel, and the category of coherent spaces of Girard. An effort is made to present coherent spaces as special kinds of event structures. A monoidal closed category of stable event structures is introduced, as well as a category of *finitary families*. Finally, to relate dI-domains with finitary families, a coreflection is given between the monoidal closed category of finitary families and the monoidal closed category of dI-domains with linear, stable function as morphisms.

Chapter 8 introduces *stable neighbourhoods*. Stable neighbourhoods characterise stable functions in a similar sense to that in which Scott-open sets characterise continuous functions. Constructions on stable neighbourhoods are given with respect to the constructions on dI-domains such as sum, product, tensor product, and stable function space. In the category of coherent spaces we introduce further constructions of stable neighbourhoods such as linear function space and shriek. In all these cases the construction preserves compactness. Results are given for the interaction of these constructions with disjoint union and intersection of sets. We also investigate stable neighbourhoods of event structures. Events as well as some relations among them are shown to determine stable neighbourhoods. However, the partially synchronous product does not preserve compactness of stable neighbourhoods.

Chapter 9 studies the logical aspects of dI-domains. Two logical frameworks are introduced, one is for $\mathbf{COH}_l$, coherent spaces with linear, stable functions; the other is for **DI**, dI-domains with stable functions. The logic of $\mathbf{COH}_l$ can be used as a logic for $\mathbf{COH}_s$, coherent spaces with stable functions, via the shriek construction. The assertion languages for the logics use a 'disjoint or', corresponding to the disjunctive nature of stable neighbourhoods. Because of the disjunctive property, assertions have to be formulated by using syntactic rules and proof rules together. Proof systems are introduced with novel proof rules to deal with different type constructions. Soundness, completeness, and expressiveness results are given.

Chapter 10 is the conclusion which contains suggestions for future research.

Chapter 2

Mathematical Prerequisites

This chapter gives a short introduction to domain theory. It sets up notations and results which will be used repeatedly in the sequel. Detailed definitions are given and known facts are stated, without proofs. Readers can find out relevant proofs else where, e.g. [Pl82], [Wi89]. Some results on **SFP** are new, which will be used in later chapters.

2.1 CPOs and Domains

A *partial order* is a set D with a binary relation on D which is reflexive ($x \sqsubseteq x$), transitive ($x \sqsubseteq y \,\&\, y \sqsubseteq z \Rightarrow x \sqsubseteq z$), and antisymmetric ($x \sqsubseteq y \,\&\, y \sqsubseteq x \Rightarrow x = y$). Without antisymmetry D is called a preorder. When $x \sqsubseteq y$, we say x is below y or y dominates x or x is less (smaller) or equal to y. Let $(D, \sqsubseteq)$ be a partial order and $X \subseteq D$ a subset. Say y is an upper bound of X if $\forall x \in X. x \sqsubseteq y$. Similarly, y is a lower bound of X if $\forall x \in X. y \sqsubseteq x$. The least upper bound (sup, supremum, lub, join) of X is an upper bound of X which is dominated by any other upper bound of X. It is written as $\bigsqcup X$, or $\bigsqcup_{i \in \omega} x_i$ if $X = \{ x_i \mid i \in \omega \}$ where ω is the set $\{ 0, 1, \cdots \}$. It is written as $a \sqcup b$ when X is $\{ a, b \}$. The greatest lower bound (inf, infimum, glb, meet) of X is a lower bound of X which dominates every other lower bound of X. It is written as $\bigsqcap X$, or $a \sqcap b$ when X is $\{ a, b \}$. When $X = \{ x_i \mid i \in \omega \}$ we write $\bigsqcap_{i \in \omega} x_i$ for $\bigsqcap X$. X is *compatible*, written $X\uparrow$, if X has an upper bound. When $\{ a, b \}$ is compatible we write $a \uparrow b$.

A *directed* set of a partial order $(D, \sqsubseteq)$ is a non-null subset $S \subseteq D$ such that $\forall s, t \in S \exists u \in S. s \sqsubseteq u \;\&\; t \sqsubseteq u$. A *complete partial order* (cpo) is a partial order $(D, \sqsubseteq)$ which has a least element $\bot$ and all least upper bounds for directed subsets. An *isolated (or finite)* element of a cpo $(D, \sqsubseteq)$ is an element $x \in D$ such that for any directed subset $S \subseteq D$ when $x \sqsubseteq \bigsqcup S$ there is an $s \in S$ such that $x \sqsubseteq s$. We write D^0 for the set of finite elements of D. A cpo $(D, \sqsubseteq)$ is *algebraic* if for all $x \in D$, $\{ e \sqsubseteq x \mid e \in D^0 \}$ is directed and $x = \bigsqcup \{ e \sqsubseteq x \mid e \in D^0 \}$. *Domains* are algebraic cpos. However we should not take this name too seriously; there have been different definitions of domains in the literature. When $(D, \sqsubseteq)$ is algebraic and D^0 is countable, D is said to be ω-algebraic. A cpo is a

Scott domain if it is ω-algebraic and *consistently complete*, *i.e.*, every compatible subset X of D has a least upper bound $\bigsqcup X$.

There is another definition of cpo based on $\omega-$increasing chains rather than directed sets. An ω-increasing chain in a partial order $(D, \sqsubseteq)$ is a countable set

$$\{x_1, x_2, \cdots x_n, \cdots\}$$

such that

$$x_1 \sqsubseteq x_2 \sqsubseteq \cdots \sqsubseteq x_n \cdots.$$

A complete partial order is a partial order $(D, \sqsubseteq)$ which has a least element $\perp$ and all least upper bounds of ω-increasing chains. An isolated (or finite) element of a cpo $(D, \sqsubseteq)$ is an element $x \in D$ such that for any ω-increasing chain $\{x_i \mid i \in \omega\} \subseteq D$, when $x \sqsubseteq \bigsqcup_{i\in\omega} x_i$ there is an x_n such that $x \sqsubseteq x_n$. A cpo $(D, \sqsubseteq)$ is *algebraic* iff for all $x \in D$ there is an ω-increasing chain $\{x_i \mid i \in \omega.\ x_i \sqsubseteq x\}$ of isolated elements such that $x = \bigsqcup_{i\in\omega} x_i$. When $(D, \sqsubseteq)$ is algebraic and the set of finite elements is countable, D is said to be *ω-algebraic.* These two definitions coincide for ω-algebraic cpos.

Theorem 2.1.1 (Plotkin) A partial order $(D, \sqsubseteq)$ is ω-algebraic in terms of directed subsets iff it is so in terms of ω-increasing chains.

For ω-algebraic cpos, then, directed subsets and ω-increasing chains can be used interchangeably.

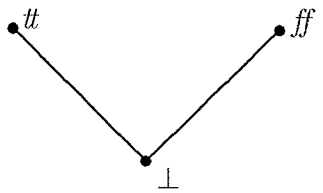
Lemma 2.1.1 If $X \subseteq D^0$ is such that $\forall x \in D.\ \{y \in X \mid y \sqsubseteq x\}$ is directed and $x = \bigsqcup\{y \in X \mid y \sqsubseteq x\}$, then $X \supseteq D^0$.

The following are a few examples of domains. Domains are sometimes drawn as Hasse diagrams with smaller elements below bigger ones, and immediately comparable pair of elements having a path between them.

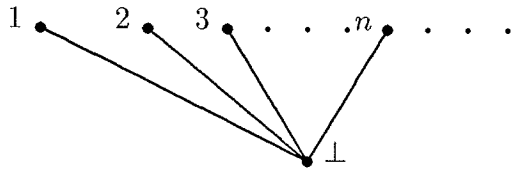
Example 2.1.1 The one-point domain $\perp$.

Example 2.1.2 The two-point domain $\mathcal{O}$, sometimes called Sierpinski space.

Example 2.1.3 The truth value cpo $\mathcal{T}$.



Example 2.1.4 $\mathcal{N}_\perp$, the domain of natural numbers.



2.2 Constructions on Cpos

The standard constructions on domains are the function space, the product, the sum, and the lifting.

Function space Let D, E be cpos. A function $f : D \to E$ is *continuous* if it is monotonic, *i.e.*

$$x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$$

and for any chain

$$x_0 \sqsubseteq x_1 \sqsubseteq x_2 \cdots$$

in D,

$$f(\bigsqcup_i x_i) = \bigsqcup_i f(x_i).$$

Cpos with continuous functions form a category. This means, in particular, that identity functions are continuous, and the composition of two continuous functions is continuous.

For cpos D and E, the function space $D \to E$ consists of elements

$$\{ f \mid f : D \to E \text{ is continuous } \}$$

ordered pointwise (coordinatewise), *i.e.*,

$$f \sqsubseteq g \text{ if } \forall x \in D.\, f(x) \sqsubseteq g(x).$$

The function space $D \to E$ is a cpo with the least element the function $\lambda x \in D.\, \bot_E$, and the sup of a chain of continuous functions is defined pointwisely, *i.e.*,

$$(\bigsqcup_i f_i)(x) = \bigsqcup_i f_i(x).$$

A function $f : D \to E$ is said to be *strict* if $f(\bot_D) = \bot_E$. The cpo of strict continuous functions of $D \to E$ form a cpo which is written as $D \to_\bot E$.

Product of cpos Let $\{ D_i \mid i \in I \}$ be an indexed family of cpos. Their *product* $\Pi_{i \in I} D_i$ consists of

$$\{ x : I \to \bigcup_{i \in I} D_i \mid \forall i \in I.\, x_i \in D_i \}$$

where we write x_i for $x(i)$, the i–th component of x. Some times we write $\langle x_i \rangle_{i \in I}$ for x. The order is defined by $x \sqsubseteq y$ if $\forall i \in I.\, x_i \sqsubseteq y_i$. When $I = \{ 1, 2 \}$ we write $\Pi_{i \in I} D_i$ as $D_1 \times D_2$. $\Pi_{i \in I} D_i$ is a cpo with the bottom element $\lambda i \in I.\, \bot_{D_i}$ and lubs of directed sets of elements determined coordinatewise. There are nature functions associated with the product called *projections*. The j–th projection $\pi_j : (\Pi_{i \in I} D_i) \to D_j$ takes each element x of the product to its j–th component x_j. Projections are continuous. There are also the *tupling operation* which takes a group of functions to the components a function to the product. Let $f_i : D \to D_i$ be continuous functions for all $i \in I$. Their tupling is a function $\langle f_i \rangle_{i \in I} : D \to \Pi_{i \in I} D_i$ which sends each element d of D to $\langle f_i(d) \rangle_{i \in I}$. This is a continuous function.

Let $h : D \to \Pi_{i \in I} D_i$. h is continuous iff for all $i \in I$ the functions $\pi \circ h : D \to D_i$ are continuous. Let $f : D_1 \times D_2 \times \cdots \times D_m \to E$ be a function. f is continuous iff it is continuous in each argument separately.

There are two important functions related to product and function space. One is the application, *app*, which is a continuous function from $[D \to E] \times D$ to E, defined as

$$app(f, x) = f(x).$$

The other is currying. *curry* is a continuous function from $(D \times E) \to F$ to $D \to [E \to F]$, defined as

$$curry(g) = \lambda x \in D\ (\lambda y \in E.\ g(x, y)).$$

Sum. Let $\{D_i \mid i \in I\}$ be an indexed family of cpos. Their *sum* $\Sigma_{i\in I} D_i$ consists of set

$$(\bigcup_{i\in I} \{i\} \times [D_i \setminus \{\perp_{D_i}\}]) \cup \{\perp\}$$

ordered by $x \sqsubseteq y$ if $x = \perp$ or $\exists i \in I.\ \exists d, d' \in D_i.\ d \sqsubseteq d'\ \&\ x = \langle i, d\rangle\ \&\ y = \langle i, d'\rangle$. When I is finite we write $D_1 + D_2 + \cdots + D_m$. $\Sigma_{i\in I} D_i$ is obviously a cpo. There are nature functions associated with the sum construction, called *injections*. The $j-$th injection $in_j : D_j \to \Pi_{i\in I} D_i$ is given by letting $in_j(\perp_{D_j}) = \perp$ and $in_j(d) = \langle j, d\rangle$ if $d \neq \perp_{D_j}$. It is easy to see that injections are continuous and strict.

Given a family of strict continuous function $f_i : D_i \to E$ ($i \in I$), we can extend them to a function

$$[f_i]_{i\in I} : \Sigma_i D_i \to E,$$

by letting

$$[f_i]_{i\in I}(\perp) = \perp$$

and

$$[f_i]_{i\in I}(\langle k, x\rangle) = f_k(x).$$

$[f_i]_{i\in I}$ is the unique strict continuous function f such that $f \circ in_i = f_i$ for all $i \in I$.

Lifting. Let D be a cpo. Its *lifting* $D_\perp$ consists of elements

$$\{\langle 0, d\rangle \mid d \in D\} \cup \{\perp\}$$

ordered by $x \sqsubseteq y$ if $x = \perp$ or $\exists d, d' \in D$ such that $d \sqsubseteq d'$, $x = \langle 0, d\rangle$ and $y = \langle 0, d'\rangle$. There is a one-one correspondence between strict continuous functions $D_\perp \to_\perp E$ and continuous functions $D \to E$. The separated sum of a family D_i, $i \in I$ of cpos is the cpo $\Sigma_{i\in I}(D_i)_\perp$.

Scott domains form a Cartesian closed category with the previous defined product and function space.

2.3 Fixed Point Theory

Let D be a cpo and $f : D \to D$ a monotonic function. $a \in D$ is called a *pre-fixed point* of f if $f(a) \sqsubseteq a$; $b \in D$ is called a *post-fixed point* of f if $b \sqsubseteq f(b)$; $c \in D$ is called a *fixed point* of f if $f(c) = c$. The following theorem is due to Tarski [Ta55].

Theorem 2.3.1 (Tarski, 1955) Let $(A, \sqsubseteq)$ be a complete lattice and

$$f : A \to A$$

a monotonic function. Then

$$\bigsqcap \{ a \in A \mid f(a) \sqsubseteq a \}$$

is the least fixed point of f and

$$\bigsqcup \{ b \in A \mid b \sqsubseteq f(b) \}$$

the greatest fixed point of f.

When it comes to cpos the meet of all pre-fixed points is still the least fixed point. However the greatest fixed point does not necessary always exist. We can write this fact about the least fixed point in the form of a rule

$$\frac{f(x) \sqsubseteq x}{fix\, f \sqsubseteq x}$$

where $fix\, f$ stands for the least fixed point of f. It is clear that $f(fix\, f) = fix\, f$. We can view fix again as a function, given by $fix(f) = fix\, f$. Such a $fix : [D \to D] \to D$ is continuous.

There is another way to get the least fixed point of a continuous function.

Theorem 2.3.2 (Kleene) Let D be a cpo, and $f : D \to D$ a continuous function. Then

$$\bigsqcup_{i \in \omega} f^i(\bot)$$

is the least fixed point of f, where $f^0 = \lambda x.\, x$, f^i stand for the function got by composing f with itself i times.

There is an induction principle called Scott induction which can be used to reason about properties of the least fixed points of continuous functions. Let P be a property of elements of a cpo D. P is *inductive* if $P(\bot)$ and for all chain

$$x_0 \sqsubseteq x_1 \sqsubseteq x_2 \cdots \sqsubseteq x_n \cdots$$

$\forall n.\, P(x_n)$ implies $P(\bigsqcup\{\, x_i \mid i \in \omega \,\})$.

Theorem 2.3.3 (Scott induction) Let D be a cpo, P an inductive property of D, and $f : D \to D$ a continuous function. If

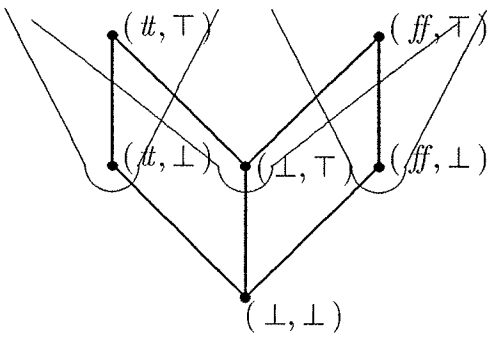
$$\forall x \in D.\, P(x) \Longrightarrow P(f(x))$$

then $P(fix\ f)$.

2.4 Scott Topology

A topology on a set S is a collection of subsets of S that is closed under finite intersection and arbitrary union. A set S and a topology $\Re$ on S forms a topological space $(S, \Re)$. A base of of the topology $\Re$ on S is a subset $\Re_1 \subseteq \Re$ such that every open set is the union of elements of $\Re_1$. Let $(\, D, \sqsubseteq \,)$ be a cpo. A subset O of D is said to be *Scott open* if O is upwards closed, *i.e.*, $\forall x \in O \forall y \in D$, $x \sqsubseteq y$ implies $y \in O$, and whenever $X \subseteq D$ is directed and $\bigsqcup X \in O$ we have $X \cap O \neq \emptyset$. The *Scott topology* on a cpo $(\, D, \sqsubseteq \,)$ consists of all the Scott open sets of $(D, \sqsubseteq)$, written $\Omega(\, D \,)$. There is a corresponding definition of Scott open set in terms of increasing chains. These two definitions agree on $\omega-$algebraic cpos and accordingly can be used interchangeably for $\omega-$algebraic cpos.

Example 2.4.1 Consider $\mathcal{T} \times \mathcal{O}$.



The three 'parabolas' in the picture represent three open sets:

$$A = \{\, (tt, \bot),\ (tt, \top) \,\},$$
$$B = \{\, (\bot, \top),\ (tt, \top),\ (ff, \top) \,\},$$
$$C = \{\, (ff, \bot),\ (ff, \top) \,\}.$$

A subset K of a space $(S, \Re)$ is compact provided that any family of open sets whose union contains K has a finite subfamily whose union also contains K. From now on all

domains concerned are supposed to be ω-algebraic and the topology is Scott topology unless otherwise indicated.

Lemma 2.4.1 $x\uparrow$ is open iff $x \in D^0$, where $x\uparrow =_{def} \{ y \in D \mid x \sqsubseteq y \}$.

Open sets of this form are called *prime open* since they are the primes in the lattice $\Omega(D)$ in the sense that if they are dominated by a join they are dominated by an element of the join.

Note we also used $\uparrow$ for compatible set. But it will always be clear from the context which we mean.

Lemma 2.4.2 $\{ x\uparrow \mid x \in D^0 \} \cup \{ \emptyset \}$ is a base for the Scott topology of an ω-algebraic domain D.

Open sets in $\{ x\uparrow \mid x \in D^0 \} \cup \{ \emptyset \}$ are said to be prime, due to the fact that whenever some open set p of the form contained in the union of a collection of open sets then p is contained in one of them.

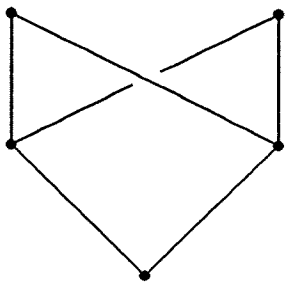
Lemma 2.4.3 A non-empty $O \subseteq D$ is open and compact iff $\exists x_i \in D^0$, $i = 1, 2, ...k$ such that $O = \bigcup_{i=1}^{k} x_i\uparrow$.

Scott topology characterises continuous functions. A function $f : D \to E$ is continuous iff $\forall O \in \Omega(E)$, $f^{-1}(O) \in \Omega(D)$. Let $f, g : D \to E$ be continuous functions. $f \sqsubseteq g$ iff $\forall O \in \Omega(E).\ f^{-1}(O) \subseteq g^{-1}(O)$.

2.5 SFP Objects

SFP objects are directed limits of Sequence of Finite inductive Partial orders introduced by Plotkin. A typical phenomenon for **SFP** objects is that compatible set of elements need not have a lub (consistently completeness does not hold).

Example 2.5.1 An **SFP** object which is not a Scott domain:



Rather than presenting the original definition given by Plotkin [Pl76], we mention an useful characterization of **SFP** objects in terms of the minimal upper bounds.

Let D be a cpo. A *minimal upper bound* of a subset $X \subseteq D$ is an upper bound of X and it is not strictly greater than any other upper bound of X. $MUB(X)$ is defined to be the set of all minimal upper bounds of X. $MUB(X)$ is said to be complete iff whenever u is an upper bound of X then $v \sqsubseteq u$ for some $v \in MUB(X)$. For each $X \subseteq D$ a sequence of subsets $U^i(X)$, $i = 0, 1, ...$ are defined as follows.

$U^0(X) = X$,

$U^{i+1}(X) = \{u \mid u \in MUB(S) \text{ for some finite } S \subseteq U^i(X)\}$,

$U^*(X) = \bigcup_{i \geq 0} U^i(X)$.

Lemma 2.5.1 If $X \subseteq D$ is a finite set of finite elements, then every element of $MUB(X)$ is finite.

Theorem 2.5.1 (Plotkin, 1976) A domain D is an **SFP** object iff
(i) D is ω-algebraic, (ii) whenever X is a finite set of finite elements of D, then $MUB(X)$ is a complete set of upper bounds of X, and (iii) $U^*(X)$ is finite.

Corollary 2.5.1 For an **SFP** object D, if K_1 and K_2 are compact open sets then so is $K_1 \cap K_2$.

2.6 Powerdomains

Let D be an $\omega-$algebraic cpo, $M[\,D\,]$ the finite, non-empty subsets of isolated elements

of D. There are three preorders on $M[D]$, $\underset{\sim}{\sqsubset}_0$, $\underset{\sim}{\sqsubset}_1$, $\underset{\sim}{\sqsubset}_2$, defined as

$$A \underset{\sim}{\sqsubset}_0 B \text{ if } \forall b \in B \exists a \in A.\, a \sqsubseteq b,$$

$$A \underset{\sim}{\sqsubset}_1 B \text{ if } \forall a \in A \exists b \in B.\, a \sqsubseteq b,$$

$$A \underset{\sim}{\sqsubset}_2 B \text{ if } A \underset{\sim}{\sqsubset}_0 B \;\&\; A \underset{\sim}{\sqsubset}_1 B.$$

Powerdomains can be directly constructed from these preorders by a technique called *ideal completion*. Let $(P, \underset{\sim}{\sqsubset})$ be a preorder with a least element. An ideal of $(P, \underset{\sim}{\sqsubset})$ is a subset $X \subseteq P$ which is non-empty, downwards-closed and directed, *i.e.*

$$X \neq \emptyset,$$

$$a, b \in X \;\&\; a \underset{\sim}{\sqsubset} b \Rightarrow a \in X,$$

and

$$a, b \in X \Rightarrow \exists c \in X.\, a \underset{\sim}{\sqsubset} c \;\&\; b \underset{\sim}{\sqsubset} c.$$

The set of ideals of $(P, \underset{\sim}{\sqsubset})$ is written as $I(P)$. $(I(P), \subseteq)$ is an algebraic domain with isolated elements $\{q \in P \mid q \underset{\sim}{\sqsubset} p\}$ for $p \in P$. $(I(P), \subseteq)$ is called the ideal completion of $(P, \underset{\sim}{\sqsubset})$.

The Smyth powerdomain, the Hoare powerdomain and the Plotkin powerdomain of an ω−algebraic cpo D are the ideal completions of $(M[D], \underset{\sim}{\sqsubset}_0)$, $(M[D], \underset{\sim}{\sqsubset}_1)$ and $(M[D], \underset{\sim}{\sqsubset}_2)$, respectively.

We will use a concrete form of powerdomains in this thesis. The idea is to pick up one biggest element (with respect to set inclusion) in each equivalent class with respect to the three preorders.

Let D be an ω−algebraic domain, and D^0 the set of its finite elements. For a non-null finite set $A \subseteq D^0$, define

$$Cl_S(A) = \{d \in D \mid \exists a \in A.\, d \sqsupseteq a\}$$

$$Cl_H(A) = \{d \in D \mid \exists a \in A.\, d \sqsubseteq a\}$$

$$Cl_P(A) = \{d \in D \mid \exists a, b \in A.\, b \sqsubseteq d \sqsubseteq a\}.$$

These operations are called closures. Clearly closures are idempotent, i.e., $Cl(Cl(A)) = Cl(A)$.

Suppose $Cl_S(A) \subseteq Cl_S(B,)$. Then $\forall a \in A, \exists b \in B, b \sqsubseteq a$. Hence $B \underset{\sim_0}{\sqsubseteq} A$. Suppose $B \underset{\sim_0}{\sqsubseteq} A$. Then $\forall a \in A, \exists b \in B, b \sqsubseteq a$. Therefore $\forall x \sqsupseteq a \in A, x \in Cl_S(B,)$. Hence $Cl_S(A) \subseteq Cl_S(B,)$ iff $B \underset{\sim_0}{\sqsubseteq} A$. The Smyth powerdomain of an $\omega-$algebraic domain D consists of, as finite elements, all subset $Cl_S(A)$ with A a non-null finite set of D^0 with the superset ordering. The limit point of an $\omega-$increasing chain

$$Cl_S(A_0) \supseteq Cl_S(A_1) \cdots \supseteq Cl_S(A_i) \supseteq \cdots$$

is the intersection

$$\bigcap_i Cl_S(A_i)$$

which is non-empty.

Similarly $Cl_H(A) \subseteq Cl_H(B,)$ iff $A \underset{\sim_1}{\sqsubseteq} B$. The Hoare powerdomain of an $\omega-$algebraic domain D consists of, as finite elements, all subsets $Cl_H(A)$ with A a non-null finite set of D^0 with the subset ordering. The limit point of an $\omega-$increasing chain

$$Cl_H(A_0) \subseteq Cl_H(A_1) \cdots \subseteq Cl_H(A_i) \subseteq \cdots$$

is the union

$$\bigcup_i Cl_H(A_i).$$

The Plotkin powerdomain of an $\omega-$algebraic domain D consists of, as finite elements (see Theorem 2.6.1), all subsets $Cl_P(A)$ with A a non-null finite set of D^0. The associated partial order on the finite elements is defined as

$Cl_P(A) \sqsubseteq_M Cl_P(B)$ iff

$(\forall b \in Cl_P(B) \exists a \in Cl_P(A).\, a \sqsubseteq b)$ & $(\forall a \in Cl_P(A) \exists b \in Cl_P(B).\, a \sqsubseteq b)$

It is easy to see that

$$Cl_P(A) \sqsubseteq_M Cl_P(B) \text{ if } A \underset{\sim_2}{\sqsubseteq} B$$

and

$$A \underset{\sim_2}{\sqsubseteq} B \text{ and } B \underset{\sim_2}{\sqsubseteq} A \text{ implies } Cl_P(A) = Cl_P(B).$$

The limit point of an $\omega-$increasing chain

$$Cl_P(A_0) \sqsubseteq_M Cl_P(A_1) \cdots \sqsubseteq_M Cl_P(A_i) \sqsubseteq_M \cdots$$

is defined to be the closure $Cl_P(A)$ where A consists of the least upper bounds of increasing chains

$$a_0 \sqsubseteq a_1 \sqsubseteq a_2 \cdots \sqsubseteq a_n \sqsubseteq \cdots$$

such that $\forall i \in \omega.\ a_i \in Cl_S(\ A_i\ )$.

Clearly $Cl_P(A)$ above has the following two properties:

$$(\ x, z \in Cl_P(A)\ \&\ x \sqsubseteq y \sqsubseteq z) \Longrightarrow y \in Cl_P(A),$$

$$(\ \forall a \in Cl(A_i) \exists x \in Cl_P(A).\ a \sqsubseteq x\ )\ \&\ (\ \forall x \in Cl_P(A) \exists a \in Cl_P(A_i).\ a \sqsubseteq x\ ).$$

Write $\mathcal{P}_P(D)$ for the set consists of closures $Cl_P(A)$ with $A \in M[D]$ together with all the limits of $\omega-$increasing chains given above.

Theorem 2.6.1 Let D be an $\omega-$algebraic cpo. The set $\mathcal{P}_P(D)$ with the order given by

$$A \sqsubseteq_M B \ \text{ iff } \ (\forall b \in B \exists a \in A.\ a \sqsubseteq b)\ \&\ (\forall a \in A \exists b \in B.\ a \sqsubseteq b)$$

forms an $\omega-$algebraic cpo which is isomorphic to the ideal completion of the preorder $(\ M[\ D\ ],\ \underset{\sim}{\sqsubseteq}_2)$.

Proof It is enough to show that $\mathcal{P}_P(D)$ is a cpo with all the finite elements being of the form $Cl_P(A)$, with $A \in M[D]$. Clearly for any $B \in \mathcal{P}_P(D)$, $Cl_P(B) = B$. Suppose $A,\ B \in \mathcal{P}_P(D)$, $A \sqsubseteq_M B$, and $B \sqsubseteq_M A$. Let $a \in A$. Because $A \sqsubseteq_M B$, there is some $b \in B$ for which $a \sqsubseteq_D b$. However, $B \sqsubseteq_M A$. Hence there is some $b' \in B$, $b' \sqsubseteq_D a$. This implies $a \in B$ since $Cl_P(B) = B$. Therefore $A \subseteq B$. Similarly $B \subseteq A$, hence $A = B$. So $\mathcal{P}_P(D)$ is a partial order.

It is obvious that $\{\ \perp_D\ \}$ is the bottom element of $\mathcal{P}_P(D)$.

Suppose

$$Cl_P(\ A_0\ ) \sqsubseteq Cl_P(\ A_1\ ) \cdots \sqsubseteq Cl_P(\ A_i\ ) \sqsubseteq \cdots$$

is a chain in $\mathcal{P}_P(D)$. Since each elements of $\mathcal{P}_P(D)$ is a limit of a similar chain but with $A_i \in M[D]$ for all i, we can, without loss of generality, form another chain

$$Cl_P(\ B_{00}\ ) \sqsubseteq_M Cl_P(\ B_{11}\ ) \cdots \sqsubseteq_M Cl_P(\ B_{ii}\ ) \sqsubseteq_M \cdots$$

where $B_{ii} \in M[D]$, and for any i, the limit of the chain

$$Cl_P(\ B_{i0}\ ) \sqsubseteq_M Cl_P(\ B_{i1}\ ) \cdots \sqsubseteq_M Cl_P(\ B_{ii}\ ) \sqsubseteq_M \cdots$$

is $Cl_P(\,A_i\,)$. We show that the limit of the above diagonal chain, $Cl_P(\,A\,)$, is equal to the least upper bound S of the set $\{\, Cl_P(\,A_i\,) \mid i \in \omega \,\}$. It then follows that S is already an element of $\mathcal{P}_P(D)$ and the limit exists. Clearly $Cl_P(\,A\,) \sqsubseteq_M S$. Hence it is enough to show that for all i,

$$Cl_P(\,A_i\,) \sqsubseteq_M Cl_P(\,A\,).$$

Assume $a \in A_i$. By definition $a = \bigsqcup_{k\in\omega} b_k$, where b_k's form a chain and for each k, $b_k \in B_{ik}$. By using the fact that each $Cl_P(B_{ij})$ is a finite element of $\mathcal{P}_P(D)$ shown in the next paragraph, we can further assume that for each j, $Cl_P(B_{ij}) \sqsubseteq_M Cl_P(B_{kj})$ whenever $i \leq j$. From the order required for B_{ij} we can easily find an element a' of A such that $a \sqsubseteq_D a'$. Similarly for any element a' of A we can find an element a of A_i such that $a \sqsubseteq_D a'$.

To show that $Cl_P(A)$ with $A \in M[D]$ are all the finite elements it is enough, by Lemma 2.1.1, to show that they themselves are finite elements, since other elements are built up as limits of $\omega-$increasing chains of those elements. Suppose $A \in M[D]$ and $Cl_P(A) \sqsubseteq_M Cl_P(B)$, where $Cl_P(B)$ is the limit of the chain

$$Cl_P(\,B_0\,) \sqsubseteq_M Cl_P(\,B_1\,) \cdots \sqsubseteq_M Cl_P(\,B_i\,) \sqsubseteq_M \cdots$$

with $B_i \in M[D]$ for each i. A is a finite set of finite elements. From the definition of B for $Cl_P(B)$ it is easy to see that there is some n, $A \underset{\sim}{\sqsubset}_1 B_n$. It remains to prove that for some $m > n$, $A \underset{\sim}{\sqsubset}_0 B_m$ since we have $A \underset{\sim}{\sqsubset}_1 B_m$ by the transitivity of $\underset{\sim}{\sqsubset}_1$. Assume this is not the case, i.e., for any $k > n$, there is some $b_k \in B_k$ such that $b_k \not\sqsupseteq_D a$ for any $a \in A$. Since $B_i \underset{\sim}{\sqsubset}_2 B_j$ when $i \leq j$, for each k there is a finite anti-chain

$$b_k \sqsupseteq_D c_{k-1} \sqsupseteq_D c_{k-2} \cdots \sqsupseteq_D c_n$$

such that $c_i \in B_i$ and $c_i \not\sqsupseteq_D a$ for any $a \in A$. Each B_k is finite. Therefore we get a finite branching, infinite tree, where each node d is an element of some B_k. By König's lemma, this tree has an infinite branch

$$d_n \sqsubseteq_D d_{n+1} \sqsubseteq_D \cdots \sqsubseteq_D d_i \cdots.$$

That infinite branch, by definition, determines an element $\bigsqcup_{i>n} d_i$ of B. However, for any element b of B there is an element a of A such that $a \sqsubseteq_D b$. Therefore for some a, $a \sqsubseteq_D \bigsqcup_{i>n} d_i$. But a is a finite element. So $a \sqsubseteq_D d_i$ for some i, which is a contradiction. ∎

For an $\omega-$algebraic cpo D, we write $\mathcal{P}_H(D)$, $\mathcal{P}_S(D)$, and $\mathcal{P}_P(D)$ for the Hoare powerdomain, the Smyth powerdomain, and the Plotkin powerdomain introduced above, respectively.

Write **SDom** for the category of Scott domains and **SFP** for the category of **SFP** objects. **SDom** is a full sub-category of **SFP**. **SFP** is closed under all the constructions mentioned above. **SDom** is closed under all but the Plotkin powerdomain. They are both cartesian closed.

2.7 Finite Elements

Now let us see what are the forms of finite elements related to constructions on domains.

Proposition 2.7.1 Let D and E be $\omega-$algebraic cpos and D^0, E^0 the set of finite elements of D and E, respectively, then

$$(D \times E)^0 = \{ \langle d, e \rangle \mid d \in D^0, e \in E^0 \}.$$

Proposition 2.7.2 Let D and E be $\omega-$algebraic cpos.

$$(D + E)^0 = \{ \langle 0, d \rangle \mid d \in D^0 \ \& \ d \neq \perp_D \} \cup \{ \langle 1, e \rangle \mid e \in E^0 \ \& \ e \neq \perp_E \} \cup \{ \perp \}.$$

Proposition 2.7.3 Let D be an $\omega-$algebraic cpo.

$$(D_\perp)^0 = \{ \langle 0, d \rangle \mid d \in D^0 \} \cup \{ \perp \}$$

Let D, E be cpos. A one-step function is a function $[a, b]$ defined as

$$[a, b](x) = \begin{cases} b & \text{if } x \sqsupseteq a \ , \\ \perp & \text{otherwise} \end{cases}$$

where $a \in D^0$, $b \in E^0$. For D, E Scott domains, $f : D \to E$ is called a step function if $\exists a_i \in D^0$, $b_i \in E^0$, $i \in I$, with I a finite set, such that

- $\forall J \subseteq I. \{ a_j \mid j \in J \}\uparrow \Rightarrow \{ b_j \mid j \in J \}\uparrow$
- $f = \bigsqcup_{i \in I} [a_i, b_i]$

Note that if

$$\forall J \subseteq I. \{ a_j \mid j \in J \}\uparrow \Rightarrow \{ b_j \mid j \in J \}\uparrow$$

then

$$\bigsqcup_{i\in I}[\,a_i,\,b_i\,] = \lambda x.\ \bigsqcup\{\,b_i \mid a_i \sqsubseteq x\,\}.$$

Proposition 2.7.4 Let D, E be Scott domains.

$$(\,D \to E\,)^0 = \{\,f \mid f \text{ is a step function in } D \to E\,\}$$

Proposition 2.7.5 If D is an ω–algebraic cpo then

$$(\mathcal{P}_S[D])^0 = \{\,Cl_S(A) \mid A \in M[\,D\,]\,\},$$

$$(\mathcal{P}_H[D])^0 = \{\,Cl_H(A) \mid A \in M[\,D\,]\,\},$$

$$(\mathcal{P}_P[D])^0 = \{\,Cl_P(A) \mid A \in M[\,D\,]\,\}.$$

2.8 Compact Open Sets

The following propositions show that all the constructions on open sets preserve compactness.

Proposition 2.8.1 Let D be an **SFP** object. D and $\emptyset$ are compact open sets of $\mathbf{\Omega}(\,D\,)$. The union and intersection of two compact open sets are compact open.

Proposition 2.8.2 If A and B are compact open sets of $\mathbf{\Omega}(\,D\,)$ and $\mathbf{\Omega}(\,E\,)$, respectively, where D, E are **SFP** objects, then

$$\{\,\langle\,u,\,v\,\rangle \mid u \in A\ \&\ v \in B\,\}$$

is a compact open set of $\mathbf{\Omega}(\,D \times E\,)$.

Proposition 2.8.3 If A and B are compact open sets of $\mathbf{\Omega}(\,D\,)$ and $\mathbf{\Omega}(\,E\,)$, respectively, where D, E are **SFP** objects, then

$$A \to B =_{def} \{\,f \in D \to E \mid A \subseteq f^{-1}(\,B\,)\,\}$$

is a compact open set of $\mathbf{\Omega}(\,D \to E\,)$.

Proof We have, for $a \in D^0$ and $b \in E^0$,

$$\begin{aligned} a\!\uparrow\!\to b\!\uparrow &= \{\,f \in D \to E \mid a\!\uparrow\subseteq f^{-1}(\,b\!\uparrow\,)\,\} \\ &= \{\,f \in D \to E \mid f(\,a\,) \sqsupseteq b\,\} \\ &= [\,a,\,b\,]\!\uparrow \end{aligned}$$

Therefore, $a\uparrow \to b\uparrow$ is a prime open set.

Let

$$A = \bigcup\{\, a_i\uparrow \mid a_i \in D^0 \;\&\; 1 \leq i \leq n \,\}$$

and

$$B = \bigcup\{\, b_j\uparrow \mid b_j \in E^0 \;\&\; 1 \leq j \leq m \,\}.$$

It is not difficult to see that

$$\begin{aligned} A \to B &= \bigcup_i a_i\uparrow \to \bigcup_j b_j\uparrow \\ &= \bigcap_i (\, a_i\uparrow \to \bigcup_j b_j\uparrow\,) \\ &= \bigcap_i (\bigcup_j a_i\uparrow \to b_j\uparrow\,) \\ &= \bigcap_i (\bigcup_j [a_i, b_j]\uparrow\,) \end{aligned}$$

Hence $A \to B$ is compact open as finite unions and intersections of compact open sets are compact open. ∎

Proposition 2.8.4 If A and B are compact open sets of $\mathbf{\Omega}(\,D\,)$ and $\mathbf{\Omega}(\,E\,)$, respectively, where $\perp_D \notin A$, $\perp_E \notin B$ and D, E are **SFP** objects, then

$$\{\,\langle 0, u\rangle \mid u \in A\,\},\quad \{\,\langle 1, v\rangle \mid v \in B\,\}$$

are compact open sets of $D + E$. If A is a compact open set of $\mathbf{\Omega}(\,D\,)$, with D an **SFP** object, then

$$\{\,\langle 0, u\rangle \mid u \in A\,\}$$

is a compact open set of $(\,D\,)_\perp$.

Proposition 2.8.5 If A is a compact open set of $\mathbf{\Omega}(\,D\,)$, with D an **SFP** object, then

$$\{\, U \in \mathcal{P}_S(\,D\,) \mid U \subseteq A\,\}$$

is a compact open set of $\mathcal{P}_S\,(\,D\,)$.

Proof Let

$$A = \bigcup\{\, a_i\uparrow \mid a_i \in D^0 \;\&\; 1 \leq i \leq n\,\}$$

where a_i's are incomparable with each other. Clearly $Cl_S(\,\{\,a_i \mid 1 \leq i \leq n\,\}\,) \in \mathcal{P}_S(\,D\,)$ and $Cl_S(\,\{\,a_i \mid 1 \leq i \leq n\,\}\,) \subseteq A$. It is routine to check that

$$\{\, U \in \mathcal{P}_S(\,D\,) \mid U \subseteq A\,\} = [\,Cl_S(\,\{\,a_i \mid 1 \leq i \leq n\,\}\,)\,]\uparrow\,.$$ ∎

Proposition 2.8.6 If A is a compact open set of $\mathbf{\Omega}(D)$, with D an **SFP** object, then

$$\{U \in \mathcal{P}_P(D) \mid U \subseteq A\}$$

is a compact open set of $\mathcal{P}_P(D)$.

Proof Let

$$A = \bigcup\{a_i\uparrow \mid a_i \in D^0 \ \& \ 1 \leq i \leq n\}$$

where a_i's are incomparable with each other. We have

$$\begin{aligned}
&U \subseteq \bigcup_i a_i\uparrow \\
&\Leftrightarrow \forall x \in U\ \exists i.\ x \sqsupseteq a_i \\
&\Leftrightarrow \exists I \subseteq \{1, 2, \cdots, n\}.\ \forall x \in U\ \exists i \in I.\ x \sqsupseteq a_i\ \&\forall i \in I\ \exists x \in U.\ a_i \sqsubseteq x \\
&\Leftrightarrow \exists I \subseteq \{1, 2, \cdots, n\}.\ \{a_i \mid i \in I\} \underset{\sim_2}{\sqsubseteq} U
\end{aligned}$$

Hence

$$\{U \in \mathcal{P}_P(D) \mid U \subseteq A\} = \bigcup_I [Cl_P(\{a_i \mid i \in I\})]\uparrow,$$

a compact open set. ∎

Proposition 2.8.7 If A is a compact open set of $\mathbf{\Omega}(D)$, with D an **SFP** object, then

$$\{L \in \mathcal{P}_H(D) \mid L \cap A \neq \emptyset\}$$

is a compact open set of $\mathbf{\Omega}\mathcal{P}_H(D)$.

Proof Let

$$A = \bigcup\{a_i\uparrow \mid a_i \in D^0 \ \& \ 1 \leq i \leq n\}$$

where a_i's are incomparable with each other. Clearly each $[Cl_H(\{a_i\})]\uparrow$ is a prime open set and

$$\{L \in \mathcal{P}_H(D) \mid L \cap A \neq \emptyset\} = \bigcup_{1 \leq i \leq n} [Cl_H(\{a_i\})]\uparrow.$$

Hence

$$\{L \in \mathcal{P}_H(D) \mid L \cap A \neq \emptyset\}$$

is compact open. ∎

Proposition 2.8.8 If A is a compact open set of $\Omega(D)$, with D an **SFP** object, then

$$\{ L \in \mathcal{P}_P(D) \mid L \cap A \neq \emptyset \}$$

is a compact open set of $\Omega\mathcal{P}_P(D)$.

Proof Let

$$A = \bigcup \{ a_i\uparrow \mid a_i \in D^0 \ \& \ 1 \leq i \leq n \}$$

where a_i's are incomparable with each other. Clearly each $[Cl_P(\{ a_i, \perp_D \})]\uparrow$ is a prime open set and

$$\{ L \in \mathcal{P}_P(D) \mid L \cap A \neq \emptyset \} = \bigcup_{1 \leq i \leq n} [Cl_P(\{ a_i, \perp_D \})]\uparrow .$$

Hence

$$\{ L \in \mathcal{P}_P(D) \mid L \cap A \neq \emptyset \}$$

is compact open. ∎

2.9 Solving Recursive Domain Equations

There are three ways to get solutions for recursively defined domains.

One is Scott's method of *universal domain*. Instead of solving domain equations one by one, we solve one big domain equation, and the solution is called a universal domain. The solution for any other domain equation can then be expressed as a *retract* on the universal domain, *i.e.*, a sub-domain D such that $f(D) = D$, where f is the function associated with the required domain equation from the universal domain to itself.

The other is the inverse limit method proposed by Smyth and Plotkin [SmPl82]. The idea is to consider the category of domains as a cpo itself and use the fixed point theory to get solution for recursively defined domains.

Let D and E be cpos and $p : D \to E$ and $q : E \to D$ be continuous functions. $\langle p, q \rangle$ froms a embedding-projection pair from D to E if $q \circ p = \lambda x \in D.\, x$ and $p \circ q \sqsubseteq \lambda x \in E.\, x$. If $\langle p, q \rangle$ is an embedding-projection pair then p, q are strict. For two projection pairs $\langle p, q \rangle$ and $\langle p', q' \rangle$ from D to E, $p \sqsubseteq p'$ iff $q' \sqsubseteq q$. Therefore, if there is an embedding-projection pair between D and E, then the embedding $p : D \to E$ uniquely determines the projection $q : E \to D$, hence we write q as p^R. Cpos with embedding-projection

pairs form a category $\mathbf{CPO}^E$, with composition of embedding-projection pairs defined as $\langle p, q\rangle \circ \langle p', q'\rangle = \langle p \circ p', q' \circ q\rangle$. Least upper bound on $\mathbf{CPO}^E$ corresponding to directed limits. Each domain equation induces a continuous 'function' on $\mathbf{CPO}^E$, and the least fixed point of the function is the initial solution of the domain equation.

The third method is due to Larsen and Winskel[LaWi84]. The idea is to work on information systems, a representation of Scott domains. Information systems are based concretely on sets and relations. They can be ordered to form a cpo. All the usual domain constructions have their counterparts as continuous operations on information systems. Recursive domain equations can then be solved using a simple result for finding the least fixed point of a continuous function. The construction of the fixed point makes the domain isomorphism an equality.

2.10 More about SFP

This section gives some results on **SFP** objects, which are useful in later chapters.

Proposition 2.10.1 Let A, B be finite sets of isolated elements of an ω–algebraic domain D. $MUB(A) = B$ and B is complete iff [1]

$$\bigcap_{a\in A} a\uparrow = \bigcup_{b\in B} b\uparrow \quad \text{and } \forall b, b' \in B.\ b \sqsubseteq b' \Rightarrow b = b'.$$

Proof

$\Rightarrow$: Let $MUB(A) = B$ and B be complete. Clearly $\forall b, b' \in B.\ b \sqsubseteq b' \Rightarrow b = b'$. Since B is complete, for each upper bound u of A there is some $b \in B$ such that $u \sqsupseteq b$. Therefore

$$\bigcap_{a\in A} a\uparrow \subseteq \bigcup_{b\in B} b\uparrow .$$

The other direction of the inclusion is obvious.

$\Leftarrow$: Assume $\bigcap_{a\in A} a\uparrow = \bigcup_{b\in B} b\uparrow$ and $\forall b, b' \in B.\ b \sqsubseteq b' \Rightarrow b = b'$. Suppose m is a minimal upper bound of A. Then it is an upper bound of A. Therefore $m \in \bigcap_{a\in A} a\uparrow$. By the provided equation, $m \in \bigcup_{b\in B} b\uparrow$. Hence $m \sqsupseteq b$ for some $b \in B$. As each member of B is clearly an upper bound of A, we must have $m = b$. So, $MUB(A) \subseteq B$.

[1] I realise that a very similar statement has already been made in [Gu85] as Lemma 4.18. It seems some condition like ours on B is needed, though missing from [Gu85].

Suppose $b \in B$. Clearly it is an upper bound of A. Assume $b \sqsupseteq q$, where q is an upper bound of A. We have $q \in \bigcap_{a \in A} a\uparrow$, which implies the existence of some $b' \in B$ such that $q \sqsupseteq b'$, by the given condition. Therefore $b \sqsupseteq b'$. Hence $b = b'$, also $b = q$. This means $b \in MUB(A)$.

Therefore $MUB(A) = B$. It is easy to see that B is complete. ∎

Definition 2.10.1 (Gunter) A set P of prime open sets of domain D is *quasi-conjunctive* iff for all non-null finite subset S of P there is a subset R of P such that

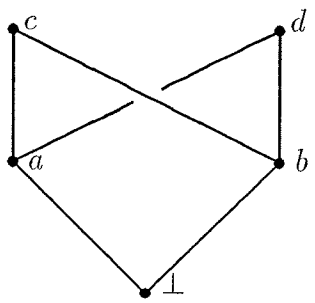
$$\bigcap S = \bigcup R.$$

Let D and E be **SFP** objects and D^0, E^0 the set of finite elements of D and E, respectively. What are the finite elements of $D \to E$? We know that for Scott domains the finite elements of the function space have the form $\bigsqcup_{i \in I}[a_i, b_i]$ where I is a finite set and $a_i \in D^0$, $b_i \in E^0$ are such that

$$\forall J \subseteq I.\ \{a_j \mid j \in J\}\uparrow \Rightarrow \{b_j \mid j \in J\}\uparrow .$$

For **SFP** objects, however, we need more information to get a finite element in function space. Although each one-step function still specifies a finite element, it is not true that the finite elements of the function space of **SFP** objects take the same form as those of the function space of Scott domains.

Consider the function space of the following domain to itself.



$[a, a] \sqcup [b, b]$ is a proper form of step function for Scott domains. But it does not give a well defined function because it does not tell us where c and d should be sent to.

A reasonable further condition to put on $\bigsqcup_{i\in I}[a_i, b_i]$ would be to require $\{a_i \mid i \in I\}$ to be MUB closed. But even that would not be enough:

$$[a, a] \sqcup [b, b] \sqcup [c, a] \sqcup [d, b]$$

meets all the requirements, and yet it does not provide full information where c should be mapped to; it can be mapped to c as well as to d.

Because of the above reasons we give the following definition of step functions between **SFP** objects.

Definition 2.10.2 Let D, E be **SFP** objects and $\{(a_i, b_i) \mid i \in I\} \subseteq D^0 \times E^0$. $\{(a_i, b_i) \mid i \in I\}$ is called *joinable* if

- $\forall X \subseteq \{a_i \mid i \in I\}. MUB(X) \subseteq \{a_i \mid i \in I\}$,
- $a_i \sqsubseteq a_j \Rightarrow b_i \sqsubseteq b_j$.

A function $f : D \to E$ is a *step function* if there is a finite joinable set $\{(a_i, b_i) \mid i \in I\}$ such that $f = \bigsqcup_{i\in I}[a_i, b_i]$.

From the definition it can be seen that if $\bigsqcup_{i\in I}[a_i, b_i]$ is a step function then

$$\forall J \subseteq I. \{a_j \mid j \in J\}\uparrow \Rightarrow \{b_j \mid j \in J\}\uparrow .$$

Theorem 2.10.1 Let D, E be **SFP** objects.

$$([D \to E])^0 = \{f \mid f \text{ is a step function of } D \to E\}.$$

Proof We prove that (i) step functions are well defined and continuous, (ii) they are finite elements of $[D \to E]$ and (iii) every $f \in [D \to E]$ is a lub of an ω-increasing chain of step functions.

(i) Let $s = \bigsqcup_{i=1}^{k}[x_i, y_i]$ be a step function. Let $E_s(x) = \{y_i \mid x_i \sqsubseteq x\}$ for $x \in D$. It is easy to see that $\{x_i \mid x_i \sqsubseteq x\}$ is directed. Hence, by Definition 2.10.2, $E_s(x)$ is also directed. So $\bigsqcup E_s(x)$ exists. We have s is well defined, and $s(x) = \bigsqcup E_s(x)$. s is clearly continuous.

(ii) To see s is a finite element of $[D \to E]$ it is enough to check that one-step functions are finite, as the lub of two (hence finitely many) isolated elements is isolated. But that one-step functions are finite is trivial.

(iii) Now we prove that $\forall f \in [D \to E]$ there is an ω-increasing chain $\{s_i \mid i \in \omega\}$ of step functions such that $f = \bigsqcup_{i\in\omega} s_i$. Write $D^0 = \{ d_i \mid i \in \omega \}, E^0 = \{ e_i \mid i \in \omega \}$ (if $\mid D^0 \mid = n$ let $d_i = d_n$ for $i \geq n$, and similarly for E^0). Any continuous function is totally specified by its values at the isolated points. Let $\{e_{ij} \mid j \in \omega\}$, $i = 1, 2, \ldots$ be ω-increasing chains of E^0 such that $\forall i. f(d_i) = \bigsqcup_{j\in\omega} e_{ij}$. Let

$$A_0 = \{ d_0 \}$$
$$A_1 = U^\star(A_0 \cup \{ d_1 \})$$
$$\ldots$$
$$A_i = U^\star(A_{i-1} \cup \{ d_i \})$$
$$\ldots$$

where $U^\star(B)$ denotes the minimal MUB closed set containing B. We have $A_i \subseteq A_{i+1}, i \geq 0$ and each A_i is a finite set of isolated elements. Let

$$s_0 = [d_0, e_{00}]$$
$$s_1 = \bigsqcup_{d\in A_1} [d, E_1(d)]$$
s_1 is a step function with $\{ E_1(d) \mid d \in A_1 \} \subseteq \{ e_{ij} \mid j \geq k_1 \}$
$$\ldots$$
$$s_i = \bigsqcup_{d\in A_i} [d, E_i(d)]$$
s_i is a step function with $\{ E_i(d) \mid d \in A_i \} \subseteq \{ e_{ij} \mid j \geq k_i \}$
$$\ldots$$

where $k_1 = 1$ and, in general, $k_i = 1 + \max \{ k \mid e_{jk} \in \{ E_{i-1}(d) \mid d \in A_{i-1} \} \}$. The existence of such step functions is guaranteed by the fact that when $d_i \sqsubseteq d_j$ we have $f(d_i) \sqsubseteq f(d_j)$, and, therefore, $\forall r \in \omega\ \exists t \in \omega$ such that $e_{ir} \sqsubseteq e_{jt}$.

It is easy to see that s_i's form an increasing chain of step functions such that $\bigsqcup_{i\in\omega} s_i = f$. ∎

Another description of finite elements in the function space of **SFP** objects is given by Gunter[Gu87].

Definition 2.10.3 (Gunter) Let D, E be **SFP** objects and $\{ (a_i, b_i) \mid i \in I \} \subseteq D^0 \times E^0$. $\{ (a_i, b_i) \mid i \in I \}$ is called *Gunter joinable* if for any $a \in D^0$, $\{ (a_i, b_i) \mid i \in I \ \&\ a_i \sqsubseteq a \}$ has a maximum in $D^0 \times E^0$.

Theorem 2.10.2 (Gunter) Let D, E be **SFP** objects and $\{ (a_i, b_i) \mid i \in I \}$ Gunter joinable. Then $\bigsqcup_{i \in I} [a_i, b_i]$ is a finite element in $D \to E$. Moreover functions of this form represent all the finite elements in $D \to E$.

The relationship between joinable and Gunter joinable sets is given by the following proposition.

Proposition 2.10.2 Let D, E be **SFP** objects. If $\{ (a_i, b_i) \mid i \in I \}$ is joinable then $\{ (a_i, b_i) \mid i \in I \}$ is Gunter joinable.

Proof Suppose $\{ (a_i, b_i) \mid i \in I \}$ is joinable. We have 1. $\{ a_i \mid i \in I \}$ is MUB closed; 2. $a_i \sqsubseteq a_j$ implies $b_i \sqsubseteq b_j$. Let $a \in D^0$. To show $\{ (a_i, b_i) \mid i \in I \ \& \ a_i \sqsubseteq a \}$ has a maximum in $D^0 \times E^0$ it is enough to prove that $\{ a_i \mid i \in I \ \& \ a_i \sqsubseteq a \}$ has a maximum. Suppose $S = MUB(\{ a_i \mid i \in I \ \& \ a_i \sqsubseteq a \})$. $S \subseteq \{ a_i \mid i \in I \}$ and for some $b \in S$. $b \sqsubseteq a$ since S is complete. This b is clearly the maximum. ∎

We remark that the converse of the proposition is not true. In later chapters only joinable sets are used because they are easier to use.

Chapter 3

Brookes' Proof System

There has been a lot of work on extending Hoare logic to parallel programming languages. One of the well-known approaches is due to Owicki and Gries [OwGr76]. The language they considered was the while-programs augmented by adding the statements for parallel composition (via shared variable) and critical regions. Their proof system used assertions similar to the Hoare triples. But the situation became much more complicated. Owicki and Gries needed interference-freedom for the soundness and auxiliary variables for the completeness of their proof system. Recently, Brookes [Br85] presented a novel proof system which avoided the use of auxiliary variables and interference-freedom. This is achieved by introducing more structures on assertions. The basic idea is to use assertions with a structure of labelled trees, where nodes are attached with predicates. In this way, one can not only describe the input-output property associated with a command, but also keep track of property of the intermediate steps in the execution of the command.

The purpose of this chapter is to introduce Brookes' proof system and present an improved, semantics derived, version for it. Brookes' proof system serves as an example of an ad hoc proof system close to the one which is semantics derived. In chapter 5 we will show that the style of Brookes' assertions is determined by Plotkin's domain of resumptions for the denotational semantics of the parallel programming language.

In the first section we give an introduction to Brookes' proof system. A counterexample is given in section 2 to show that Brookes' proof system is not complete as it appeared. Section 3 proposes a simpler proof system, avoiding the use of labels. Our work in chapter 5 shows that it is important not to use labels if one wants the proof system to be 'semantics-based'. The new proof system is shown to be sound. The completeness issue of the proposed proof system is discussed in section 4.

3.1 Brookes' Proof System

The programming language Brookes deals with is the same as the one considered

by Owicki and Gries. The syntactic categories of the programming language are **Iexp**, integer expressions, which is ranged over by E; **Bexp**, boolean expressions, which is ranged over by B; **Loc**, locations, which is ranged over by I (we also use x for a location) and **Com**, commands, by Γ. The syntax for **Iexp**, **Bexp** are taken for granted (for more detail of the basic settings for operational semantics and proof system. See e.g. Winskel's book[Wi89]). The syntax for **Com** is:

$$\Gamma ::= \alpha : \textbf{skip} \mid \alpha : I := E \mid \Gamma_1; \Gamma_2 \mid \Gamma_1 \,||\, \Gamma_2 \mid \textbf{await } \beta : B \textbf{ then } \alpha : \Gamma$$

$$\mid \textbf{if } \beta : B \textbf{ then } \Gamma_1 \textbf{ else } \Gamma_2 \mid \textbf{while } \beta : B \textbf{ do } \Gamma$$

Nested await statements are not allowed. The symbol **null** is used to represent termination. Brookes gave an operational semantics to the programming language in the standard way using labelled transition systems (To have a idea for the operational semantics see Section 3.3 for a closely related, unlabelled, transition system). He then proposed a novel proof system for the programming language using tree-like assertions which take the form

$$\varphi ::= P \sum_{i=1}^{n} \alpha_i P_i \varphi_i \mid \varphi_1 \oplus \varphi_2 \mid P\circ \mid P\bullet$$

where P's are logical formulae of some conditional language and α_i's are atomic action labels. The symbol $\oplus$ is understood as the logical 'and'. A different symbol was used to distinguish it from the logical 'and' used in the condition language.

States Σ consists of functions from locations to truth values and integers. $\sigma[E/I]$ represents a state which agrees with σ except possibly at location I where it stores the value of E evaluated at state σ.

A typical conditional language is given as follows:

$$P ::= \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid P_0 \wedge P_1 \mid P_0 \vee P_1 \mid \neg P \mid P_0 \Rightarrow P_1 \mid \forall i.P \mid \exists i.P$$

where a's are arithmetic expressions specified by

$$a ::= n \mid I \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

with n ranges over numbers **Num**, I, x ranges over locations **Loc** and i ranges over integer variables **Intvar**. Clearly if one wants to define $\sigma \models P$, the relation which expressing when a state σ satisfies a condition P, one has to specify what values the integer variables are assigned. Therefore it is natural to define $\sigma \models P$ with respect to an *interpretation* **It** of the

integer variables, written as $\sigma \models^{It} P$ [Wi89]. In [Br85] the relation $\sigma \models B$ is mentioned, but it is not explained how to define $\sigma \models P$ without referring to an interpretation. As one of the possible way, let us assume all the conditions P we are going to use are closed, i.e. all the integer variables are bound. This is a little restrictive but there should be no problem extending our treatment to the general case.

There are some facts about $\sigma \models P$ which will be used later, from time to time. We collect them here; the proofs can be found in ordinary text book on foundations of verification, e.g. [Wi89]. We have

$$P \Leftrightarrow Q \text{ iff } \sigma \models P \text{ implies } \sigma \models Q,$$

$$\sigma \models \bigvee_{i=1}^{n} P_i \text{ iff } \exists i.\ \sigma \models P_i,$$

$$\sigma[E/I] \models P \text{ iff } \sigma \models P[E/I].$$

A command Γ satisfies an assertion φ, which Brookes writes $\models \Gamma \textbf{ sat } \varphi$, when

$$\varphi \equiv \oplus_{1=1}^{n} \varphi_i \ \&\ \forall i\ (1 \leq i \leq n).\ \models \Gamma \textbf{ sat } \varphi_i$$

or

$$\varphi \equiv P \sum_{i=1}^{n} \alpha_i\, P_i\, \varphi_i,$$

which satisfies the following two properties:

1 $\forall\sigma\forall\alpha.[\,(\sigma \models P)\ \&\ \langle\Gamma,\sigma\rangle \xrightarrow{\alpha} \langle\Gamma',\sigma'\rangle \Rightarrow \exists i.\ \alpha = \alpha_i\ \&\ \sigma' \models P_i\ \&\ \models \Gamma' \textbf{ sat } \varphi_i\,]$

2 $\forall i\forall\sigma.[\,(\sigma \models P) \Rightarrow \exists\langle\Gamma',\sigma'\rangle.\ \langle\Gamma,\sigma\rangle \xrightarrow{\alpha_i} \langle\Gamma',\sigma'\rangle\ \&\ \models \Gamma' \textbf{ sat } \varphi_i\ \&\ \sigma' \models P_i\,]$

Brookes calls Γ **sat** φ a *command assertion.* A proof system is proposed by Brookes to derive properties of commands expressed in such assertions. It consists of the following proof rules:

Brookes' Proof System

$$(B0)\quad \frac{\neg \mathbf{root}\ (\varphi)}{\Gamma\ \mathbf{sat}\ \varphi}$$

$$(B1)\quad \alpha : \mathbf{skip}\ \ \mathbf{sat}\ \{\, P \,\}\alpha\{\, P \,\}\{\, \mathbf{true} \,\}\bullet$$

$$(B2)\quad I := E\ \mathbf{sat}\ \{\, P[E/I] \,\}\alpha\{\, P \,\}\{\, \mathbf{true} \,\}\bullet$$

$$(B3)\quad \frac{\Gamma\ \mathbf{sat}\ \varphi \quad \mathbf{safe}(\varphi)}{\mathbf{await}\ \beta : B\ \mathbf{then}\ \alpha : \Gamma\ \mathbf{sat}\ \{\, \mathbf{root}(\varphi) \& B \,\}\alpha\{\, \mathbf{leaf}(\varphi) \,\}\{\, \mathbf{true} \,\}\bullet}$$

$$(B4)\quad \mathbf{await}\ B\ \mathbf{then}\ \Gamma\ \mathbf{sat}\ \{\, \neg B \,\}\circ$$

$$(B5)\quad \frac{\Gamma_1\ \mathbf{sat}\ \varphi \quad \Gamma_2\ \mathbf{sat}\ \psi}{[\Gamma_1 \,||\, \Gamma_2]\ \mathbf{sat}\ [\varphi \,||\, \psi]}$$

$$(B6)\quad \frac{\Gamma_1\ \mathbf{sat}\ \varphi \quad \Gamma_2\ \mathbf{sat}\ \psi}{(\,\Gamma_1 ; \Gamma_2\,)\ \mathbf{sat}\ \ (\varphi ; \psi\,)}$$

$$(B7)\quad \frac{\Gamma_1\ \mathbf{sat}\ \{\, P \,\} \sum_{i=1}^{n} \alpha_i P_i\, \varphi_i}{\mathbf{if}\ \beta : B\ \mathbf{then}\ \Gamma_1\ \mathbf{else}\ \Gamma_2\ \ \mathbf{sat}\ \ \{\, P \& B \,\} \sum_{i=1}^{n} \alpha_i P_i\, \varphi_i}$$

$$(B8)\quad \frac{\Gamma_2\ \mathbf{sat}\ \{\, P \,\} \sum_{i=1}^{n} \alpha_i P_i\, \varphi_i}{\mathbf{if}\ \beta : B\ \mathbf{then}\ \Gamma_1\ \mathbf{else}\ \Gamma_2\ \ \mathbf{sat}\ \ \{\, P \& \neg B \,\} \sum_{i=1}^{n} \alpha_i P_i\, \varphi_i}$$

$$(B9)\quad \frac{\mathbf{while}\ \beta : B\ \mathbf{do}\ \Gamma\ \mathbf{sat}\ \rho \quad \Gamma\ \mathbf{sat}\ \{\, P \,\} \sum_{i=1}^{n} \alpha_i P_i\, \varphi_i}{\mathbf{while}\ \beta : B\ \mathbf{do}\ \Gamma\ \mathbf{sat}\ \{\, P \,\&\, B \,\} \sum_{i=1}^{n} \alpha_i P_i\, (\,\varphi_i ; \rho\,)}$$

$$(B10)\quad \mathbf{while}\ \beta : B\ \mathbf{do}\ \Gamma\ \mathbf{sat}\ \{\, P \& \neg B \,\}\beta\{\, P \,\}\{\, \mathbf{true} \,\}\bullet$$

$$(C1)\quad \frac{\Gamma\ \mathbf{sat}\ \varphi \quad \Gamma\ \mathbf{sat}\ \psi}{\Gamma\ \mathbf{sat}\ (\,\varphi \oplus \psi\,)}$$

$$(C2)\quad \frac{\Gamma\ \mathbf{sat}\ (\,\varphi \oplus \psi\,)}{\Gamma\ \mathbf{sat}\ \varphi \quad \Gamma\ \mathbf{sat}\ \psi}$$

$$(C3)\quad \frac{\Gamma\ \mathbf{sat}\ \varphi \quad \varphi \Rightarrow \psi}{\Gamma\ \mathbf{sat}\ \psi}$$

In the proof system || and ; are constructions on assertions introduced in [Br85].

It is claimed in [Br85] that

" Although we do not provide a proof in this paper, the proof system formed

by (B0)-(B10), (C1)-(C3), is sound: all provable assertions are valid. The system is also relatively complete in the sense of Cook: every true assertion of the form Γ **sat** φ is provable, given that we can prove all of the conditions necessary in applications of the critical region rule and of modus ponens. Both of these rules require assumptions which take the form of implication between conditions. Write $\mathbf{Th} \vdash \Gamma$ **sat** φ if this can be proved using assumptions from **Th**.

Theorem 3. If $\mathbf{Th} \vdash \Gamma$ **sat** φ then $\mathbf{Th} \models \Gamma$ **sat** φ.

Theorem 3. If $\mathbf{Th} \models \Gamma$ **sat** φ then $\mathbf{Th} \vdash \Gamma$ **sat** φ. "

3.2 A Counterexample

Brookes' proof system as presented in [Br85] is, however, not complete. Consider the command

$$\Gamma_0 \equiv \textbf{await } \alpha : \ x > 0 \textbf{ then } \beta : \ W,$$

where

$$W \equiv \textbf{while } \gamma : \ x > 0 \textbf{ do } \delta : \ x := x - 1.$$

We have

Proposition 3.2.1 $\models \Gamma_0$ **sat** $\{ x > 0 \}\ \beta\ \{ x = 0 \}\ \{ x = 0 \}\ \bullet$.

Proof By the transition rule for the operational semantics we have, for await statements,

$$\frac{\langle \Gamma, \sigma \rangle \rightarrow^* \langle \textbf{null}, \sigma' \rangle \quad \sigma \models B}{\langle \textbf{await } \beta : \ B \textbf{ then } \alpha : \ \Gamma, \sigma \rangle \xrightarrow{\alpha} \langle \textbf{null}, \sigma' \rangle}.$$

Therefore, an atomic transition for the await statement **await** $\beta : B$ **then** $\alpha : \Gamma$ is possible at a state σ iff Γ can make a number of atomic transitions from σ and then terminates. It is then easy to see from definition that we have

$$\models \Gamma_0 \textbf{ sat } \{ x > 0 \}\ \beta\ \{ x = 0 \}\ \{ x = 0 \}\ \bullet .$$

∎

This valid command assertion is, however, not provable from Brookes' proof system. To show that it is not provable we need to use some notions like the depth, root and leaf of assertions, as well as safe assertions, as introduced in [Br85].

Definition 3.2.1 The depth of assertions are integers defined by

$$\begin{aligned}
&\mathbf{depth}\,(P\circ) && = 0\\
&\mathbf{depth}\,(P\bullet) && = 0\\
&\mathbf{depth}\,(P\sum_{i=1}^{n}\alpha_i P_i\varphi_i) && = 1+\mathbf{max}\{\,\mathbf{depth}\,(\varphi_i)\mid 1\le i\le n\,\}\\
&\mathbf{depth}\,(\bigoplus_{i=1}^{n}\varphi_i) && = \mathbf{max}\{\,\mathbf{depth}\,(\varphi_i)\mid 1\le i\le n\,\}
\end{aligned}$$

Clearly every assertion has a finite depth.

Definition 3.2.2 The root of an assertion is defined as

$$\begin{aligned}
&\mathbf{root}\,(P\bullet) && = P\\
&\mathbf{root}\,(P\circ) && = P\\
&\mathbf{root}\,(P\sum_{i=1}^{n}\alpha_i P_i\varphi_i) && = P\\
&\mathbf{root}\,(\bigoplus_{i=1}^{n}\varphi_i) && = \bigvee_{i=1}^{n}\mathbf{root}\,(\varphi_i)
\end{aligned}$$

Definition 3.2.3 The leaf of an assertion is defined as

$$\begin{aligned}
&\mathbf{leaf}(P\bullet) && = P\\
&\mathbf{leaf}(P\circ) && = \mathbf{false}\\
&\mathbf{leaf}(P\sum_{i=1}^{n}\alpha_i P_i\varphi_i) && = \bigvee_{i=1}^{n}\mathbf{leaf}(\varphi_i)\\
&\mathbf{leaf}(\bigoplus_{i=1}^{n}\varphi_i) && = \bigvee_{i=1}^{n}\mathbf{leaf}\ (\varphi_i)
\end{aligned}$$

The operation **leaf** was not introduced for assertions of the form $\bigoplus_{i=1}^{n}\varphi_i$ in Brookes's paper. But apparently it is needed.

Note that for an assertion $P\sum_{i=1}^{n}\alpha_i P_i\varphi_i$, no restriction is imposed on the adjacent pair of conditions P_i, $\mathbf{root}\,(\varphi_i)$. It reflects the fact that interruption between atomic actions is allowed in the presence of parallelism. Sometimes however, it is useful to have assertions whose adjacent conditions are logically related. They are called safe assertions.

Definition 3.2.4 Define the predicate **safe** on assertions such that

$$\begin{aligned}
&\mathbf{safe}\ (P\bullet)\\
&\mathbf{safe}\ (P\sum_{i=1}^{n}\alpha_i P_i\varphi_i)\ \Leftrightarrow [\,\forall i.\ P_i\Rightarrow\mathbf{root}\,(\varphi_i)\,]\ \&\ \forall i.\ \mathbf{safe}\ (\varphi_i)\\
&\mathbf{safe}\ (\bigoplus_{i=1}^{n}\varphi_i)\Leftrightarrow\forall i.\ \mathbf{safe}\ (\varphi_i)
\end{aligned}$$

Proposition 3.2.2 $\forall n \geq 0$,

$$(\models W \textbf{ sat } \varphi \;\&\; \exists\sigma.\, \sigma[n/x] \models \textbf{root}(\varphi) \;\&\; \textbf{safe}(\varphi)) \;\Rightarrow\; \textbf{depth}\ (\varphi) \geq n.$$

where

$$W \equiv \textbf{while}\ \ \gamma :\ x > 0\ \textbf{do}\ \ \delta :\ x := x - 1.$$

Proof By mathematical induction on n. The base case is trivial.

Induction step: Suppose the conclusion is true for all $n = N$. We show that it is true for $n = N + 1$.

Assume $\models W$ **sat** φ & $\exists\sigma.\ \sigma[N+1/x] \models \textbf{root}(\varphi)$ & $\textbf{safe}(\varphi)$. Let $\varphi \equiv \bigoplus_{i=1}^{k} \varphi_i$, where $k \geq 1$, and φ_i's are of the form $Q \sum_{j=1}^{m} \beta_j Q_j \psi_j$. Since $\textbf{root}(\varphi) = \bigvee_{i=1}^{k} \textbf{root}\ (\varphi_i\ )$, $\exists i.\ \sigma[N+1/x] \models \textbf{root}\ (\varphi_i\ )$. Write $Q \sum_{j=1}^{m} \beta_j Q_j \psi_j$ for this φ_i. We have $\sigma[N+1/x] \models Q$ and $\models W$ **sat** $Q \sum_{j=1}^{m} \beta_j Q_j \psi_j$. By operational semantics we have a transition

$$\langle \delta :\ x := x - 1,\ \sigma[N+1/x] \rangle \xrightarrow{\delta} \langle \textbf{null},\ \sigma[N/x] \rangle,$$

which enables the transition

$$\langle W,\ \sigma[N+1/x] \rangle \xrightarrow{\delta} \langle W,\ \sigma[N/x] \rangle.$$

Therefore $\exists j.\ \sigma[N/x] \models Q_j$ & $\models W$ **sat** ψ_j. As $Q \sum_{j=1}^{m} \beta_j Q_j \psi_j$ is safe, we have $Q_j \Rightarrow \textbf{root}(\psi_j)$ and $\textbf{safe}(\psi_j)$. Hence $\sigma[N/x] \models \textbf{root}(\psi_j)$. By induction hypothesis $\textbf{depth}(\psi_j) \geq N$. Therefore $\textbf{depth}(\varphi) \geq N + 1$. ∎

Proposition 3.2.3 Γ_0 **sat** $\{x > 0\}\ \beta\ \{x = 0\}\{x = 0\}\bullet$ is not provable from Brookes's proof system.

Proof By inspecting the proof rules it is easy to see that to derive the desired command assertion we must prove W **sat** φ, with $\{x > 0\} \Rightarrow \textbf{root}(\varphi)$ and $\textbf{safe}(\varphi)$ and then use $(B3)$. But it is impossible to prove W **sat** φ, with $\{x > 0\} \Rightarrow \textbf{root}(\varphi)$ and $\textbf{safe}(\varphi)$. Suppose $\varphi = \oplus_{i=1}^{n} \varphi_i$ where φ_i's are of the form $P_i \sum \alpha_k Q_k \psi_k$. Since $\textbf{root}(\varphi) = \bigvee_{i=1}^{n} P_i$ and $\{x > 0\} \Rightarrow \textbf{root}(\varphi)$, there must be some i for which there is a infinite subset ϵ of natural numbers such that $\forall k \in \epsilon.\ \sigma[k/x] \models P_i$. We must have, therefore, $\models W$ **sat** φ_i, $\textbf{safe}(\varphi_i)$ and $\forall k \in \epsilon.\ \sigma[k/x] \models P_i$. By proposition 3.2.2 $\textbf{depth}(\varphi_i) \geq k$ for any $k \in \epsilon$, i.e. $\textbf{depth}(\varphi) = \infty$, a contradiction. ∎

Note that whether we allow free integer variables in condition or not does not affect the proof of proposition 3.2.3 because from the condition $\{ x > 0 \} \Rightarrow \mathbf{root}(\varphi)$ we can assume $\mathbf{root}(\varphi)$ to be closed, i.e. it does not have free integer variables.

The counterexample is not surprising if one notices the fact that with await statements and Brookes's assertion language it is possible to express the termination property of sequential while-programs. In other words, the proof system is concerned with not only partial correctness but also total correctness in an indirect way. We have

Proposition 3.2.4 Suppose Γ is a sequential while-program. It terminates from any state with property P iff

$$\models (\, \mathbf{await}\ \beta : \mathbf{true}\ \mathbf{then}\ \alpha : \Gamma\,)\ \mathbf{sat}\ P\alpha\{\,\mathbf{true}\,\}\{\,\mathbf{true}\,\}\bullet .$$

Proof (only if): Obvious.

(if): Suppose $\models (\mathbf{await}\ \beta : \mathbf{true}\ \mathbf{then}\ \alpha : \Gamma)\ \mathbf{sat}\ P\alpha\{\,\mathbf{true}\,\}\{\,\mathbf{true}\,\}\bullet$. By definition for any $\sigma \models P$ there exists $\langle \Gamma', \sigma' \rangle$ such that $\langle \mathbf{await}\ \beta : \mathbf{true}\ \mathbf{then}\ \alpha : \Gamma\ ,\ \sigma \rangle \xrightarrow{\alpha} \langle \Gamma', \sigma' \rangle$ and $\models \Gamma'\ \mathbf{sat}\ \mathbf{true}\bullet$, $\sigma' \models \mathbf{true}$. It can be easily seen from the transition rules that we must have

$$\Gamma' = \mathbf{null}\ \&\langle \Gamma\ , \sigma \rangle \rightarrow^* \langle \mathbf{null},\ \sigma' \rangle,$$

that is, Γ terminates from state σ, as Γ is deterministic. ∎

3.3 Removing the Labels

As is shown in the previous section, Brookes' proof system is not complete, and the trouble comes from await statements. What we are going to do in the rest of this chapter is quite modest: We present a neater proof system than Brookes', without using labels. The point of not using labels is, more importantly, that the proof system naturally derived from semantics does not involve labels (see chapter 5). We show that in the absence of labels one still has a sound proof system. However, unfortunately, though I believe this proof system is complete, I cannot yet prove it.

The programming language we are considering is specified by

$$\Gamma ::= \mathbf{skip} \mid\ I := E \mid \Gamma_1; \Gamma_2 \mid \Gamma_1 \parallel \Gamma_2 \mid\ \mathbf{if}\ B\ \mathbf{then}\ \Gamma_1\ \mathbf{else}\ \Gamma_2 \mid \mathbf{while}\ B\ \mathbf{do}\ \Gamma.$$

We use an unlabelled transition system

$$((\mathbf{Com} \times \Sigma) \cup \Sigma, \rightarrow)$$

to give an operational semantics for the programming language. An element of $\mathbf{Com} \times \Sigma$, say, $\langle \Gamma, \sigma \rangle$ represents a stage (configuration) in a computation at which the remaining command to be executed is Γ, and the current state is σ. Configurations of the form $\langle \mathbf{null}, \sigma \rangle$ used by Brookes are denoted by a termination state σ. The relation $\rightarrow$ is the atomic transitions specified by the following rules:

Atomic Transitions

$$(S1) \quad \langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma$$

$$(S2) \quad \langle I := E, \sigma \rangle \rightarrow \sigma[E/I]$$

$$(S3) \quad \frac{\langle \Gamma_1, \sigma \rangle \rightarrow \langle \Gamma_1', \sigma' \rangle}{\langle \Gamma_1; \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_1'; \Gamma_2, \sigma' \rangle}$$

$$(S4) \quad \frac{\langle \Gamma_1, \sigma \rangle \rightarrow \langle \Gamma_1', \sigma' \rangle \quad \sigma \models B}{\langle \mathbf{if}\ B\ \mathbf{then}\ \Gamma_1\ \mathbf{else}\ \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_1', \sigma' \rangle}$$

$$(S5) \quad \frac{\langle \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2', \sigma' \rangle \quad \sigma \models \neg B}{\langle \mathbf{if}\ B\ \mathbf{then}\ \Gamma_1\ \mathbf{else}\ \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2', \sigma' \rangle}$$

$$(S6) \quad \frac{\langle \Gamma, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle \quad \sigma \models B}{\langle \mathbf{while}\ B\ \mathbf{do}\ \Gamma, \sigma \rangle \rightarrow \langle \Gamma'; \mathbf{while}\ B\ \mathbf{do}\ \Gamma, \sigma' \rangle}$$

$$(S7) \quad \frac{\sigma \models \neg B}{\langle \mathbf{while}\ B\ \mathbf{do}\ \Gamma, \sigma \rangle \rightarrow \sigma}$$

$$(S8) \quad \frac{\langle \Gamma_1, \sigma \rangle \rightarrow \langle \Gamma_1', \sigma' \rangle}{\langle \Gamma_1 \mid\mid \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_1' \mid\mid \Gamma_2, \sigma' \rangle}$$

$$(S9) \quad \frac{\langle \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2', \sigma' \rangle}{\langle \Gamma_1 \mid\mid \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_1 \mid\mid \Gamma_2', \sigma' \rangle}$$

When $\langle \Gamma, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle$ appear above the line in a transition rule it is understood that $\langle \Gamma', \sigma' \rangle$ may take the form σ', and in that case the command Γ' also disappears below the line, together with the possibly related ; or || construction. For example, our transition system has a rule instance

$$\frac{\langle \Gamma_1, \sigma \rangle \rightarrow \sigma'}{\langle \Gamma_1; \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2, \sigma' \rangle}.$$

The syntax of the assertion language is given by:

$$\varphi ::= P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i \mid \varphi_1 \wedge \varphi_2 \mid \bullet$$

where P's are closed logical formulae of the conditional language, $\sum_{i=1}^{n} P_i \times \varphi_i$ stands for $P_1 \times \varphi_1 + P_2 \times \varphi_2 + \cdots P_n \times \varphi_n$. We use $\rightarrow$ and $\times$ in the assertion language to make the structure of assertion more transparent. Later we will see that the structure of the assertion is derivable form the domain of resumptions, which gives the use of $\rightarrow$ and $\times$ a justification from denotational semantics. Since $\wedge$ can appear at different levels of an assertion we don't need an extra $\oplus$ for conjunction. Note we don't have $P\circ$ and $P\bullet$ in our assertion language, but only $P \times \bullet$, for $P\bullet$. This is because in the absence of await statements there is no possibility of 'improper' termination.

Definition 3.3.1 A command Γ satisfies an assertion φ, written $\Gamma \models \varphi$, when

$$\varphi \equiv \bigwedge_{1=1}^{n} \varphi_i \;\&\; \forall 1 \leq i \leq n \,.\, \Gamma \models \varphi_i$$

or

$$\varphi \equiv P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i,$$

which satisfies the following two properties:

1 $\forall \sigma.[\,(\sigma \models P) \;\&\; \langle \Gamma, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle \Rightarrow \exists i.\sigma' \models P_i \;\&\; \Gamma' \models \varphi_i \;\&$

$(\sigma \models P) \;\&\; \langle \Gamma, \sigma \rangle \rightarrow \sigma' \Rightarrow \exists i.\sigma' \models P_i \;\&\; \varphi_i = \bullet\,]$

2 $\forall i \forall \sigma.(\sigma \models P) \Rightarrow [\,\exists \langle \Gamma', \sigma' \rangle.\ \langle \Gamma, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle \;\&\; \Gamma' \models \varphi_i \;\&\; \sigma' \models P_i$ **or**

$\exists \sigma'.\ \langle \Gamma, \sigma \rangle \rightarrow \sigma' \;\&\; \varphi_i = \bullet \;\&\; \sigma' \models P_i\,]$

Note $\bullet$ is used more as a symbol than an assertion. Its only function is to help us to write down the syntax for assertion uniformly in one line. Notice after avoiding the **null**, there is no command which satisfies $\bullet$!

Suppose $\Gamma_1 \models \varphi_1$ and $\Gamma_2 \models \varphi_2$. What kind of assertions will $\Gamma_1 \parallel \Gamma_2$ and $\Gamma_1 ; \Gamma_2$ satisfy? It seems reasonable to expect that $\Gamma_1 \parallel \Gamma_2$ satisfies an assertion formed by some kind of 'interleaving' of Γ_1 and Γ_2 and $\Gamma_1 ; \Gamma_2$ satisfies an assertion formed by some kind of 'concatenation' of Γ_1 and Γ_2. There are, then, the formal constructions of parallel composition and sequential composition of assertions for proof rules for parallel composition and sequential composition.

Definition 3.3.2 For parallel composition we have $\bullet \parallel \varphi = \varphi \parallel \bullet = \varphi$. If $\varphi = P \to \sum_{i=1}^{n} P_i \times \varphi_i$ and $\psi = Q \to \sum_{j=1}^{m} Q_j \times \psi_j$ then

$$\varphi \parallel \psi = (P \wedge Q) \to [\sum_{i=1}^{n} P_i \times (\varphi_i \parallel \psi) + \sum_{j=1}^{m} Q_j \times (\varphi \parallel \psi_j)].$$

If φ and ψ are conjunctions, $\varphi \parallel \psi$ is defined to be a conjunction consists of conjuncts of the form

$$(P \wedge Q) \to \sum_{i=1}^{n} P_i \times (\varphi_i \parallel \psi) + \sum_{j=1}^{m} Q_j \times (\varphi \parallel \psi_j)$$

where $P \to \sum_{i=1}^{n} P_i \times \varphi_i$ is a conjunct of φ and $Q \to \sum_{j=1}^{m} Q_j \times \psi_j$ is a conjunct of ψ.

Definition 3.3.3 For sequential composition, define

$$\bullet ; \varphi = \varphi$$

$$[\, Q \to \textstyle\sum_{j=1}^{m} Q_j \times \psi_j \,] ; \varphi = Q \to \sum_{j=1}^{m} Q_j \times (\psi_j ; \varphi)$$

$$(\bigwedge_{k=1}^{n} \varphi_k) ; \psi = \bigwedge_{k=1}^{n} (\varphi_k ; \psi)$$

The depth, root and leaf of an assertion are defined similar to definitions 3.2.1, 3.2.2 and 3.2.3, respectively.

The omission of labels from Brookes' proof system changes the proof system more than one would expect. We have, for example,

$$(\textbf{if } x \geq 0 \textbf{ then } x := 1 \textbf{ else } x := 1) \models \{\, \textbf{true} \,\} \to \{\, x = 1 \,\} \times \bullet$$

but not

$$\models \quad (\textbf{if } \alpha : x \geq 0 \textbf{ then } \beta : x := 1 \textbf{ else } \gamma : x := 1)$$

$$\textbf{sat } \{\, \textbf{true} \,\} \, [\beta \{\, x = 1 \,\} \{\, \textbf{true} \,\} \bullet + \gamma \{\, x = 1 \,\} \{\, \textbf{true} \,\} \bullet]$$

in Brookes' proof system.

In general, a command satisfies more assertions after the removing of labels.

Proposition 3.3.1 Let Γ be a command without involving any await statement and φ be an assertion without the use of o. Then $\models \Gamma$ **sat** φ in Brookes' proof system implies $\Gamma^* \models \varphi^*$, where Γ^* is the command got by removing all the labels in Γ and φ^* is the assertion got by taking $P\bullet$ to $\bullet$, $P \sum \alpha_j Q_j \psi_j$ to $P \to \sum Q_j \times \psi_j^*$ and $\oplus$ to $\wedge$.

Proof By a simple mathematical induction on the depth of assertions. ∎

Note that the reverse of the proposition is not true because of the example given above.

The proof system has two parts. The first part is concerned with the logical relation between assertions. We axiomatise $\varphi \Rightarrow \psi$, standing for φ implies ψ. $\Leftrightarrow$ stands for both $\Leftarrow$ and $\Rightarrow$.

Sequent Calculus

$(\bullet)$ $\quad \bullet \wedge \bullet \Leftrightarrow \bullet$

$(\Rightarrow \wedge)$ $\quad \dfrac{\varphi \Rightarrow \varphi_1 \quad \varphi \Rightarrow \varphi_2}{\varphi \Rightarrow \varphi_1 \wedge \varphi_2}$

$(\Rightarrow 1)$ $\quad \varphi_1 \wedge \varphi_2 \Rightarrow \varphi_1$

$(\Rightarrow 2)$ $\quad \varphi_1 \wedge \varphi_2 \Rightarrow \varphi_2$

$(\wedge)$ $\quad (P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i) \wedge (Q \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i) \Rightarrow (P \vee Q) \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$

$(\Rightarrow)$ $\quad \dfrac{\models Q \Rightarrow P \quad \forall i\, \exists j\, .\, \models P_i \Rightarrow Q_j \,\&\, \varphi_i \Rightarrow \psi_j \quad \forall j\, \exists i\, .\, \models P_i \Rightarrow Q_j \,\&\, \varphi_i \Rightarrow \psi_j}{(P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i) \Rightarrow (Q \rightarrow \sum_{j=1}^{m} Q_j \times \psi_j)}$

Write $\vdash \varphi \Rightarrow \psi$ when $\varphi \Rightarrow \psi$ is derivable from the sequent calculus.

Proposition 3.3.2 $\vdash \varphi \parallel \psi \Leftrightarrow \psi \parallel \varphi$, $\quad \vdash \varphi \parallel (\psi_1 \wedge \psi_2) \Rightarrow \varphi \parallel \psi_1$.

Proof Trivial.

Proposition 3.3.3 If $\vdash \varphi \Rightarrow \psi$ then $\Gamma \models \varphi$ implies $\Gamma \models \psi$.

Proof Obvious.

The second part of the proof system consists of the following rules.

Correctness Rules

$(P0)$ $\quad \Gamma \vdash \{\mathbf{false}\} \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$

$(P1)$ $\quad \mathbf{skip} \ \vdash P \rightarrow P \times \bullet$

$(P2)$ $\quad (I := E) \vdash P[E/I] \rightarrow P \times \bullet$

$(P3)$ $\quad \dfrac{\Gamma_1 \vdash \varphi \quad \Gamma_2 \vdash \psi}{(\Gamma_1 \,||\, \Gamma_2) \vdash \varphi \,||\, \psi}$

$(P4)$ $\quad \dfrac{\Gamma_1 \vdash \varphi \quad \Gamma_2 \vdash \psi}{(\Gamma_1 ; \Gamma_2) \vdash \varphi ; \psi}$

$(P5)$ $\quad \dfrac{\Gamma_1 \vdash P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i}{\mathbf{if}\ B\ \mathbf{then}\ \Gamma_1 \mathbf{else}\ \Gamma_2 \ \vdash\ (P \wedge B) \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i}$

$(P6)$ $\quad \dfrac{\Gamma_2 \vdash P \rightarrow \sum_{j=1}^{m} Q_j \times \psi_j}{\mathbf{if}\ B\ \mathbf{then}\ \Gamma_1 \mathbf{else}\ \Gamma_2 \ \vdash\ (P \wedge \neg B) \rightarrow \sum_{j=1}^{m} Q_j \times \psi_j}$

$(P7)$ $\quad \dfrac{\mathbf{while}\ B\ \mathbf{do}\ \Gamma \vdash \rho \quad \Gamma \vdash P \ \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i}{\mathbf{while}\ B\ \mathbf{do}\ \Gamma \vdash (P \wedge B) \rightarrow \sum_{i=1}^{n} P_i \times (\varphi_i \,;\, \rho)}$

$(P8)$ $\quad \mathbf{while}\ B\ \mathbf{do}\ \Gamma \vdash (P \wedge \neg B) \rightarrow (P \wedge \neg B) \times \bullet$

$(S1)$ $\quad \dfrac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \ \vdash\ (\varphi \wedge \psi)}$

$(S2)$ $\quad \dfrac{\Gamma \ \vdash\ \varphi \quad \vdash \varphi \Rightarrow \psi}{\Gamma \ \vdash\ \psi}$

The proof system (sequent calculus and correctness rules) is sound, i.e. $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$. The soundness can be validated by showing that all the rules are sound, i.e. the rules preserve valid formulae. For rules $(P0)$, $(P1)$, $(P2)$, $(P8)$, $(S1)$ and $(S2)$ this is trivial while for $(P5)$, $(P6)$ and $(P7)$ it is an easily verified fact. We give a proof for the more complicated case $(P3)$. The proof for $(P4)$ is similar to that of $(P3)$ but simpler, and hence omitted.

Proposition 3.3.4 If $\Gamma_1 \models \varphi$ and $\Gamma_2 \models \psi$ then $(\Gamma_1 \,||\, \Gamma_2) \models (\varphi \,||\, \psi)$.

Proof Define A to be a property on pairs of natural numbers, with

$A(s, t) \Longleftrightarrow$
$(\mathbf{depth}\ (\varphi) \leq s\ \&\ \mathbf{depth}\ (\psi) \leq t\ \&\ \Gamma_1 \models \varphi\ \&\ \Gamma_2 \models \psi) \Longrightarrow (\Gamma_1 \,||\, \Gamma_2) \models (\varphi \,||\, \psi)$

We prove by induction $\forall s \geq 1 \forall t \geq 1.\ A(s, t)$. First we prove $A(1, t)$ for $t \geq 1$. Let $t = 1$ and $\Gamma_1 \models P \rightarrow \sum_{i=1}^{n} P_i \times \bullet$, $\Gamma_2 \models Q \rightarrow \sum_{j=1}^{m} Q_j \times \bullet$. Suppose $\sigma \models P \wedge Q$ and $\langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle$. By the transition rules for the operational semantics we have either $\Gamma' = \Gamma_2$ & $\langle \Gamma_1, \sigma \rangle \rightarrow \sigma'$ or $\Gamma' = \Gamma_1$ & $\langle \Gamma_2, \sigma \rangle \rightarrow \sigma'$. For the first case we have, for some i, $\sigma' \models P_i$ and $\Gamma' \models Q \rightarrow \sum_{j=1}^{m} Q_j \times \bullet$ since $\Gamma_1 \models P \rightarrow \sum_{i=1}^{n} P_i \times \bullet$. Similarly for the second case we have $\sigma' \models Q_j$ and $\Gamma' \models P \rightarrow \sum_{i=1}^{n} P_i \times \bullet$ for some j.

On the other hand, for any i, any σ, if $\sigma \models P \wedge Q$ then $\exists \sigma'.\ \langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2, \sigma' \rangle$ with $\sigma' \models P_i$ since $\Gamma_1 \models P \rightarrow \sum_{i=1}^{n} P_i \times \bullet$. For a similar reason given any j, any σ, if $\sigma \models P \wedge Q$ then $\exists \sigma'.\ \langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_1, \sigma' \rangle$ with $\sigma' \models Q_j$.

In summary, we have shown that $\Gamma_1 \parallel \Gamma_2 \models \varphi \parallel \psi$, where $\varphi = P \rightarrow \sum_{i=1}^{n} P_i \times \bullet$, and $\psi = Q \rightarrow \sum_{j=1}^{m} Q_j \times \bullet$. It is easy to see from the proof that the conclusion also holds for those φ and ψ which are conjunctions.

Now assume $A(1, k)$. Following the same pattern above we can prove $A(1, k + 1)$. Hence $A(1, t)$ for all $t \geq 1$. By symmetry we have $A(s, 1)$ for all $s \geq 1$.

Let

$$B(k) \Longleftrightarrow [\forall s \leq k \forall t \geq 1.\ A(s, t)] \ \& \ [\forall t \leq k \forall s \geq 1.\ A(s, t)].$$

We use induction in k to show $B(k)$ for all $k \geq 1$.

Base case: $B(1)$ has just been proved above.

Induction step: Assume $B(k)$, we want to prove $B(k + 1)$. The task is divided into two sub-inductions to show that $\forall s \leq k + 1 \forall t \geq 1.\ A(s, t)$ and $\forall t \leq k + 1 \forall s \geq 1.\ A(s, t)$. We need only check the first one because of symmetry.

The base case $\forall s \leq k + 1.\ A(s, 1)$ for the sub-induction is implied by $B(1)$. Let $\forall s \leq k + 1 \forall t \leq N.\ A(s, t)$. We prove $\forall s \leq k + 1 \forall t \leq N + 1.\ A(s, t)$. Suppose $\Gamma_1 \models \varphi$ and $\Gamma_2 \models \psi$ where $\varphi = P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$, $\psi = Q \rightarrow \sum_{j=1}^{m} Q_j \times \psi_j$ and $\mathbf{depth}(\varphi) = k + 1$, $\mathbf{depth}(\psi) = N + 1$. Suppose $\sigma \models P \wedge Q$ and $\langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle$. By the transition rules for the operational semantics we have either

$$\Gamma' = \Gamma_1' \parallel \Gamma_2 \ \& \ \langle \Gamma_1, \sigma \rangle \rightarrow \langle \Gamma_1', \sigma' \rangle$$

or

$$\Gamma' = \Gamma_1 \parallel \Gamma_2' \ \& \ \langle \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma_2', \sigma' \rangle.$$

Since $\Gamma_1 \models \varphi$, for the first case we have, for some i, $\sigma' \models P_i$ and $\Gamma_1' \models \varphi_i$. But $\mathbf{depth}(\varphi_i) \leq k$. Using the induction hypothesis $B(k)$ we have $\Gamma' \models \varphi_i \parallel \psi$. Similarly for the second case we have $\sigma' \models Q_j$ and $\Gamma_2' \models \psi_j$. As $\mathbf{depth}(\psi_j) \leq N$ and $\mathbf{depth}(\varphi) = k+1$, we have $\Gamma' \models \varphi \parallel \psi_j$ by the sub-induction hypothesis $\forall s \leq k+1 \forall t \leq N.\ A(s,\ t)$.

On the other hand, for any i, any σ, if $\sigma \models P \wedge Q$ then $\exists \sigma'.\ \langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle$ with $\sigma' \models P_i$ and $\Gamma' \models \varphi_i \parallel \psi$ because $\Gamma_1 \models P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$, and for similar reason given any j, any σ, if $\sigma \models P \wedge Q$ then $\exists \sigma'.\ \langle \Gamma_1 \parallel \Gamma_2, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle$ with $\sigma' \models Q_j$, $\Gamma' \models \varphi \parallel \psi_j$.

In summary, we have shown $\Gamma_1 \parallel \Gamma_2 \models \varphi \parallel \psi$. It is easy to see the proof still works if φ and ψ are conjunctions.

We have proved $\forall s \leq k+1 \forall t \leq N+1.\ A(s,\ t)$. Hence $\forall s \leq k+1 \forall t \geq 1.\ A(s,\ t)$. This finishes the inner induction and we have $B(k+1)$.

Therefore $\forall k \geq 1.\ B(k)$. ∎

Proposition 3.3.5 If $\Gamma_1 \models \varphi$ and $\Gamma_2 \models \psi$ then $(\Gamma_1\ ;\ \Gamma_2) \models (\varphi\ ;\ \psi)$.

Proof Similar to but simpler than that of proposition 3.3.4; use mathematical induction on the depth of assertions. ∎

3.4 Completeness

As with Brookes' original proof system we do not attain completeness because of await statements. It is reasonable to expect that our system is relatively complete without using the await statements. But it is not clear to me at the moment how to prove the completeness, though I believe the system is complete. The difficulty comes from the parallel composition where one wants to somehow decompose an assertion satisfied by a parallel composition into assertions satisfied by its components.

The idea of the proof is to use structured induction on commands. For each program construction $cons(\Gamma_0,\ \Gamma_i)$ where $i = 0$ or 1 (e.g. **if** B **then** Γ_0 **else** Γ_1) we show that if $\Gamma_0 \models \varphi_0$ implies $\Gamma_0 \vdash \varphi_0$ and $\Gamma_i \models \varphi_i$ implies $\Gamma_i \vdash \varphi_i$ then we have $cons(\Gamma_0,\ \Gamma_i) \vdash \varphi$ when

$cons(\Gamma_0, \Gamma_i) \models \varphi$. We hope to establish completeness in this way. We have

Proposition 3.4.1 **skip** $\models \varphi$ implies **skip** $\vdash \varphi$.

Proof Let **skip** $\models P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$. The case where $P \Leftrightarrow$ **false** is trivial because of rule $(P0)$. Otherwise there must be some state σ for which $\sigma \models P$. From definition 3.3.1, **skip** $\models P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$ implies

$$\forall i \forall \sigma.(\sigma \models P) \Longrightarrow \exists \sigma'.\ \langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma' \ \&\ \varphi_i = \bullet \ \&\ \sigma' \models P_i.$$

Hence $\forall i.\ \varphi_i = \bullet$. By the transition rule for **skip** we have $\sigma' = \sigma$, which implies $\models P \Rightarrow P_i$ for all i. We have **skip** $\vdash P \rightarrow P \times \bullet$ and

$$\vdash P \rightarrow P \times \bullet \Rightarrow P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$$

by rule $(\Rightarrow)$ for the sequent calculus. Therefore **skip** $\vdash P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$. ∎

Note for each case we need only check for assertions of the form $P \rightarrow \sum P_i \times \phi_i$. Once it is done for assertions of this particular form it is an easy step to get things work in general, for supposing $\Gamma \models \varphi$ and φ is a conjunction, we must have $\Gamma \models \psi$ for each conjunct ψ of φ. Hence $\Gamma \vdash \psi$, and using $(S1)$ we get $\Gamma \vdash \varphi$.

Proposition 3.4.2 $(I := E) \models \varphi$ implies $(I := E) \vdash \varphi$.

Proof Let $(I := E) \models P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$. Suppose $P \not\Leftrightarrow$ **false**. $(I := E) \models P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$ implies

$$\forall i \forall \sigma.(\sigma \models P) \Longrightarrow \exists \sigma'.\ \langle I := E, \sigma \rangle \rightarrow \sigma' \ \&\ \varphi_i = \bullet \ \&\ \sigma' \models P_i.$$

Hence $\forall i.\ \varphi_i = \bullet$, for $P \not\Leftrightarrow$ **false**. By the transition rule for $I := E$ we have $\sigma' = \sigma[E/I]$, which implies $\sigma \models P_i[E/I]$, hence $\models P \Rightarrow P_i[E/I]$ for all i. By $(P2)$, $(I := E) \vdash [(\bigwedge_{i=1}^{n} P_i[E/I]) \rightarrow (\bigwedge_{i=1}^{n} P_i) \times \bullet]$. Also, by $(\Rightarrow)$ for the sequent calculus

$$\vdash (\bigwedge_{i=1}^{n} P_i[E/I]) \rightarrow (\bigwedge_{i=1}^{n} P_i) \times \bullet] \Rightarrow P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i.$$

Therefore $(I := E) \vdash P \rightarrow \sum_{i=1}^{n} P_i \times \varphi_i$. ∎

The next step is to try to prove a similar result for parallel composition, i.e. to prove that given $[\Gamma_1 \models \varphi$ implies $\Gamma_1 \vdash \varphi]$ and $[\Gamma_2 \models \psi$ implies $\Gamma_2 \vdash \psi]$ we have $[\Gamma_1 \parallel \Gamma_2 \vdash \eta$ if $\Gamma_1 \parallel \Gamma_2 \models \eta]$. This is reduced to the proof of a slightly stronger statement

$$\Gamma_1 \parallel \Gamma_2 \models P \to \sum_{i=1}^{n} P_i \times \varphi_i$$
$$\Longrightarrow \exists \psi_1, \psi_2.\ \Gamma_1 \models \psi_1 \ \&\ \Gamma_2 \models \psi_2 \ \&\ \vdash \psi_1 \parallel \psi_2 \Rightarrow P \to \sum_{i=1}^{n} P_i \times \varphi_i.$$

It is a kind of decomposition of assertion with respect to parallel composition. One would guess that such ψ_1 and ψ_2 take the form $P \to \sum Q_j \times \xi_j$ but the following example shows that it is not always the case.

Let

$$\Gamma_1 = \textbf{if } x > 0 \textbf{ then } x := 1 \textbf{ else } x := 0,$$

$$\Gamma_2 = \textbf{if } x > 0 \textbf{ then } x := 0 \textbf{ else } x := 1$$

and

$$\phi_0 = \{\textbf{false}\} \to \{\textbf{true}\} \times \bullet.$$

It is easy to check that

$$\Gamma_1 \parallel \Gamma_2 \models \{\textbf{true}\} \to [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ].$$

However, we have neither

$$\Gamma_1 \models \{\textbf{true}\} \to [\ \{x = 0\} \times \bullet + \cdots\ ]$$

nor

$$\Gamma_1 \models \{\textbf{true}\} \to [\ \{x = 1\} \times \bullet + \cdots\ ].$$

Hence we cannot decompose the assertion

$$\{\textbf{true}\} \to [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ]$$

in a way we expected.

Nevertheless, we can derive

$$\Gamma_1 \parallel \Gamma_2 \models \{\textbf{true}\} \to [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ]$$

in the following way.

$$x := 0 \vdash \{\mathbf{true}\} \rightarrow \{x = 0\} \times \bullet \qquad (P2)$$

$$\Gamma_1 \vdash (\{x \leq 0\} \rightarrow \{x = 0\} \times \bullet) \wedge \phi_0 \qquad (P6),\ (P0)$$

$$\Gamma_2 \vdash (\{x \leq 0\} \rightarrow \{x = 1\} \times \bullet) \wedge \phi_0$$

$$\Gamma_1 \parallel \Gamma_2 \vdash \{x \leq 0\} \rightarrow [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ] \qquad (P3),\ (\Rightarrow)$$

Similarly

$$\Gamma_1 \parallel \Gamma_2 \vdash \{x > 0\} \rightarrow [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ],$$

hence

$$\Gamma_1 \parallel \Gamma_2 \vdash \{\mathbf{true}\} \rightarrow [\ \{x = 0\} \times \phi_0 + \{x = 1\} \times \phi_0\ ]$$

by $(\wedge)$.

It seems in general the decomposition is complicated and I haven't yet found one. But still, this chapter obtained a neater proof system whose assertion language is derived from the denotational semantics, as will be shown in Chapter 5.

Chapter 4

An Information System Representation of SFP

Information systems were introduced by Scott [Sc82] initially with the intention of making domain theory accessible to a wider audience. In this representation the idea of *information* is made explicit–each element is seen as a collection of *information quanta*. It gives a logical approach to domain theory, in which properties of domains can be derived from assumptions about the entailments between propositions expressing properties of computations.

Scott domains form a foundational framework for denotational semantics. As shown by Plotkin [Pl76], however, they cannot treat parallelism and concurrency adequately in some aspects. A more general framework is the **SFP** objects introduced by Plotkin [Pl76]. It has been an open question how best one can represent the **SFP** objects as information systems since the work of Dana Scott [Sc82]. Scott himself has made some observations [Sc84] about how to characterise the 2/3-**SFP** objects. Gunter's work [Gu85] provides useful results about **SFP** objects. The manuscript of Winskel [Wi83a] introduced the idea of adding disjunction to the information systems to represent **SFP**.

Although itself an interesting topic, our motivation to work on information systems is to gain understanding about the logical theories which characterise **SFP** objects. It provides the foundation and clue which suggests the right kind of axioms to put for the logic of **SFP** objects. In this chapter we give a representation of **SFP** objects called *strongly finite information systems* (SFIFs), which are particular kinds of *generalised information systems* (GIS). Instead of using entailment of the form $X \vdash a$ we work with the classical Gentzen style $X \vdash Y$, which provides the power to express disjunction of propositions. Generalised information systems represent complete partial orders. They are shown to be reasonable neat structures which can represent different kinds of domains. As a by-product, we get a compositional way to build up MUB-closed sets of an **SFP** object from those of the component **SFP** objects.

The contents of this chapter is organised as follows. Section 1 is a brief review of infor-

mation systems. Section 2 introduces generalised information systems and studies some of their basic properties. Section 3 presents a category of strongly finite information systems and introduces constructions on them, in particular the function space and Plotkin powerdomain constructions. Section 4 is about the powerdomain constructions. Section 5 proposes a cpo of strongly finite information systems and shows that all the constructions induce continuous functions on this cpo so that it is possible to solve recursive equations with these systems.

4.1 Information Systems

Intuitively, an information system is a structure describing the logical relations among propositions that can be made about computations. It consists of a set of propositions, a consistency predicate and an entailment relation specified as follows[1].

Definition 4.1.1 An *information system* is a structure $\underline{A} = (A, Con, \vdash)$ where

- A is a set of propositions
- $Con \subseteq \mathrm{Fin}(A)$, the consistent sets
- $\vdash \subseteq Con \times A$, the entailment relation

which satisfy

1. $X \subseteq Y \;\&\; Y \in Con \Rightarrow X \in Con$
2. $a \in A \Rightarrow \{a\} \in Con$
3. $X \vdash a \Rightarrow X \cup \{a\} \in Con$
4. $a \in X \;\&\; X \in Con \Rightarrow X \vdash a$
5. $(\forall b \in Y.\, X \vdash b \;\&\; Y \vdash c) \Rightarrow X \vdash c$

Notation. We write $\mathrm{Fin}(A)$ for the set of finite subsets of A. Write $X \vdash^* Y$ to mean $\forall b \in Y.\ X \vdash b$; $X \subseteq^{fin} y$ to mean X is a finite subset of y. π_0 and π_1 are projections which give the first and the second argument, respectively, when applied to a pair; When they are applied to a set S of pairs, we write $\pi_0 S$ and $\pi_1 S$ for the set of first arguments and second arguments of elements in S, respectively.

Propositions are basic facts that can be affirmed about computations. They are

[1]For convenience of getting a cpo of information systems we use a definition slightly different from the original one given in [Sc82], without using a distinguished Δ standing for the proposition that is always true. The definition we use is the same as the one given in [LaWi84].

the units of information. *Con* contains all finite subsets of propositions that are non-contradictory, in a sense related to the computation under consideration. A relation $X \vdash a$ can be interpreted as: If the propositions in X are true of a computation, then a is also true of the computation. Thus the relation $\vdash$ stands for entailment.

An information system determines a family of subsets of propositions called its elements. Intuitively, an element consists of a set of propositions that can be truly made about a fixed possible computation. Thus it is expected that the propositions should be in consistency with each other and, if a finite set of propositions is valid for a computation, all the logical consequences should also be valid for it.

Definition 4.1.2 The *elements* $Pt(\underline{A})$, of an information system $\underline{A} = (A, Con, \vdash)$ consists of subsets x of propositions which are

1. finitely consistent: $X \subseteq^{fin} x \Rightarrow X \in Con$,
2. closed under entailment: $X \subseteq x \;\&\; X \vdash a \Rightarrow a \in x$.

For an information system $\underline{A}$, $(Pt\,\underline{A}, \subseteq)$ is a Scott domain[Sc82]. More generally, information systems form a category with the approximable mappings as morphisms, which is equivalent [Mac71] to the category of Scott domains. Constructions such as product, sum and function space have been proposed on information systems [Sc82] [LaWi84], corresponding to those on domains. By using information systems it is possible to solve recursive equations concretely [LaWi84] with the resulting isomorphism being an equality.

4.2 Generalised Information Systems

This section introduces generalised information systems. Basic properties of GISs are given, as well as an axiom which is shown to characterise **SFP** objects. Generalised information systems are based on the same idea as information systems. They too express the logical relations between propositions about computations. However, as the name suggests, the structures are more general. In the framework of GIS, we can represent both Scott domains and **SFP** objects by putting different axioms on it. Instead of using entailment of the form $X \vdash a$ we work with the classical Gentzen style sequent $X \vdash Y$ with the understanding that the logical relation among the propositions on the left hand side of $\vdash$ as $\wedge$ (the logial 'and') and on the right hand side as $\vee$ (the logical 'or').

Definition 4.2.1 A *Generalised information system* (GIS) is a structure

$$\underline{A} = (A, \vdash)$$

where

- A is a set of propositions
- $\vdash \subseteq (\mathrm{Fin}(A) \setminus \{\emptyset\}) \times \mathrm{Fin}(A)$ is the entailment relation

which satisfy:

1. $\forall a \in A.\ \{a\} \nvdash \emptyset$
2. $a \in X \Rightarrow X \vdash \{a\}$
3. $((\forall b \in Y.\ X \vdash \{b\}) \ \&\ Y \vdash Z) \Rightarrow X \vdash Z$
4. $\{a\} \vdash X \Rightarrow \exists b \in X.\ \{a\} \vdash \{b\}$
5. $(X \vdash Y \ \&\ Y' \subseteq Y \ \&\ \forall b \in Y'.\ \{b\} \vdash Z) \Rightarrow X \vdash (Y \setminus Y') \cup Z$
6. $X \nvdash \emptyset \ \&\ X \vdash Y \Rightarrow \exists b \in Y.\ X \cup \{b\} \nvdash \emptyset$

We write $X \nvdash Y$ for $\neg(X \vdash Y)$, *i.e.*, X does not entail Y. As a common practice $\bigvee \emptyset$ is understood as 'false'. Therefore axiom 1 says that each single proposition is consistent. Axiom 2 is a kind of reflexivity and axiom 3 and 5 transitivity. Obviously it is not appropriate to insist $X \vdash Y \ \&\ Y \vdash Z \Rightarrow X \vdash Z$. Axiom 4 expresses that each proposition behaves like an 'atom'. The intuition behind axiom 6 does not appear to be so obvious at a first sight, but if we think of each proposition as a set, axiom 6 makes perfect sense. It says that if $\bigcap S_i \not\subseteq \emptyset$ and $\bigcap S_i \subseteq \bigcup T_j$, then for some j, $T_j \cap \bigcap S_i \not\subseteq \emptyset$. In addition, axiom 6 is a special case of axiom 3 in definition 4.1.1.

We have not used a consistency predicate *Con* in the definition of GIS. We can, however, recover one from a GIS in the following way. Let $\underline{A} = (A, \vdash)$ be a GIS. Define *Con* to be the set such that $X \in \mathit{Con}$ iff $X \nvdash \emptyset$. This is a reasonable definition of consistency since it satisfies the corresponding axioms required for an information system. Axiom 1 implies that $a \in A \Rightarrow \{a\} \in \mathit{Con}$. Let $X \subseteq Y$ and $Y \nvdash \emptyset$. Then we must have $X \nvdash \emptyset$ because if $X \vdash \emptyset$ we can derive $Y \vdash \emptyset$ from axiom 3, which is a contradiction.

To be more precise sometimes we write a GIS with subscripts, such as $\underline{A} = (A, \vdash_{\underline{A}})$. For convenience we abbreviate

$$Y \in \mathrm{Fin}(A) \ \&\ \forall a \in Y.\ X \vdash \{b\} \qquad \text{as} \qquad X \vdash^* Y$$

and

$$X \vdash Y \;\&\; \forall b \in Y \,.\, \{ b \} \vdash^* X \qquad \text{as} \qquad X \dashv\vdash Y.$$

Proposition 4.2.1 For a GIS $\underline{A}$, we have

1. $a \in X'$ implies $\{ a \} \vdash X'$,
2. $X' \supseteq X \vdash Y \subseteq Y'$ implies $X' \vdash Y'$,
3. $X \nvdash \emptyset \;\&\; X \vdash^* Y$ implies $X \cup Y \nvdash \emptyset$.

Proof 1. Let $X = Y = \{ a \}$, $Y' = \emptyset$, and $Z = X \setminus \{ a \}$ in axiom 5 of definition 4.2.1. By axiom 2 it is vacuously true that $\forall b \in \emptyset . \{ b \} \vdash Z$. Hence $\{ a \} \vdash X'$.

2. Notice that by axiom 2 of definition 4.2.1 we have $X' \vdash \{ a \}$ for all $a \in X'$. Therefore $X' \vdash \{ b \}$ for all $b \in X$. Then by using axiom 3 of definition 4.2.1 we get $X' \vdash Y$. Since $Y \subseteq Y'$, we have $\forall b \in Y . \{b\} \vdash Y'$ by the first conclusion of the proposition. Thus $X' \vdash Y'$, by axiom 5 of definition 4.2.1.

3. Assume $X \vdash \{ b \}$ for all $b \in Y$ and $X \nvdash \emptyset$. We show $X \cup Y \nvdash \emptyset$ by induction on the size of Y. Clearly it holds when Y has only one element because of axiom 6 of definition 4.2.1. Suppose it holds for all Y with a size not greater than n. Let $Y = Y' \cup \{ c \}$ where Y' is of size n. Then $X \vdash \{ c \}$ and $X \subseteq X \cup Y'$. By the second conclusion of this proposition $X \cup Y' \vdash \{ c \}$. Hence $X \cup Y' \cup \{ c \} = X \cup Y \nvdash \emptyset$ by axiom 6 of definition 4.2.1. ∎

A GIS determines a family of subsets of propositions called its elements.

Definition 4.2.2 The *elements* $| \underline{A} |$, of a GIS $\underline{A} = (A, \vdash)$ consists of subsets x of propositions which are *closed under entailment:*

$$(X \subseteq x \;\&\; X \vdash Y) \Longrightarrow x \cap Y \neq \emptyset.$$

Note the finite consistency (*i.e.* $X \subseteq^{fin} x \Rightarrow X \nvdash \emptyset$) of an element is implied by its closedness under entailment which prevents $X \vdash \emptyset$ for $X \subseteq x$. We have the obvious fact that $\emptyset \in | \underline{A} |$. Clearly GISs do not necessarily represent Scott domains. A GIS represents, however, an algebraic cpo. This is illustrated by a sequence of propositions.

Proposition 4.2.2 Let $\underline{A}$ be a generalised information system. Let

$$\overline{a} = \{ b \mid b \in A \;\&\; \{ a \} \vdash \{ b \} \}$$

where $a \in A$. Then $\overline{a} \in |\underline{A}|$.

Proof Suppose $X \subseteq^{fin} \overline{a}$ and $X \vdash Y$. Then $\{a\} \vdash^* X$. Hence $\{a\} \vdash Y$ by axiom 3 for GIS. By axiom 4 of definition 4.2.1, there is some $c \in Y$ such that $\{a\} \vdash \{c\}$, which implies $c \in \overline{a}$. ∎

Proposition 4.2.3 For a GIS $\underline{A}$, $|\underline{A}|$ ordered by inclusion is a cpo.

Proof Obviously the bottom element is $\emptyset$ and $|\underline{A}|$ is a partial order.

Let S be a directed subset of $|\underline{A}|$. It is easy to check that $\bigcup S \in |\underline{A}|$; this is because for any finite subset X of $\bigcup S$ there is an $s \in S$ such that $X \subseteq s$, by the directedness of S and finiteness of X. It is then trivial to show the closedness of $\bigcup S$ under entailment. ∎

Proposition 4.2.4 For each $a \in A$, $\overline{a}$ is a finite element of $|\underline{A}|$.

Proof For any directed $S \subseteq |\underline{A}|$, if $\overline{a} \subseteq \bigsqcup S$, *i.e.*, $\overline{a} \subseteq \bigcup S$, then we have $a \in \bigcup S$. Hence $a \in s$ for some $s \in S$, and $\overline{a} \subseteq s$. ∎

Proposition 4.2.5 $|\underline{A}|$ is algebraic.

Proof This is because for any $x \in |\underline{A}|$, $x = \bigsqcup\{\overline{a} \mid a \in x \cap A\}$. ∎

From now on we consider GISs with the underlying proposition set to be countable. I am not sure at the moment whether $|\underline{A}|$ is ω−algebraic even when A is countable. But whether it is true or not does not affect any of the forth coming result.

As remarked before, for a GIS $\underline{A}$, $|\underline{A}|$ is usually not a Scott domain. It is possible, however, to get Scott domains by an extra axiom.

Definition 4.2.3 A GIS $\underline{A}$ is called *definite* if it satisfies the following axiom

$$(\text{Definite}) \qquad X \vdash Y \& X \nvdash \emptyset \Rightarrow \exists b \in Y.\, X \vdash \{b\}$$

This axiom has the effect that every entailment $X \vdash Y$ with $X \nvdash \emptyset$ is a consequence of some entailment of the form $X \vdash \{b\}$ with $b \in Y$, which makes it possible to recover an information system (in the original sense of Scott) from a GIS in an obvious way (though they are special information systems since $\emptyset \vdash^* Y$ implies $Y = \emptyset$).

Proposition 4.2.6 Let $\underline{A}$ be definite. Let

$$\overline{X} = \{\, b \mid b \in A \,\&\, X \vdash \{b\}\,\}$$

where $X \nvdash \emptyset$. $\{\, \overline{X} \mid X \nvdash \emptyset \,\}$ is the set of finite elements of $|\,\underline{A}\,|$.

Proof Suppose $Y \subseteq^{fin} \overline{X}$ and $Y \vdash Z$. We have $X \vdash^* Y$. Hence $X \vdash Z$, by axiom 3 of definition 4.2.1. Since $X \nvdash \emptyset$, by the axiom Definite, there is some $c \in Z$ such that $X \vdash \{c\}$, which implies $c \in \overline{X}$. Hence each $\overline{X}$ is an element of $|\,\underline{A}\,|$.

If S is a directed subset of $|\,\underline{A}\,|$ and $\overline{X} \subseteq \bigcup S$ it is easy to see that we have $\overline{X} \subseteq s$ for some $s \in S$; this is because if $\overline{X} \subseteq x$ then $X \subseteq x$ and for any finite subset X of $\bigcup S$ there is an $s \in S$ such that $X \subseteq s$, by the directedness of S and finiteness of X. Therefore $\overline{X}$ is a finite element.

Suppose x is a finite element. Clearly $\{\, \overline{X} \mid X \subseteq x \,\}$ is a directed set since $X \subseteq Y$ implies $\overline{Y} \subseteq \overline{X}$. Also $x \subseteq \bigcup\{\, \overline{X} \mid X \subseteq x \,\}$. Therefore $x \subseteq \overline{X}$ for some $X \subseteq x$ by the finiteness of x. And we must have $x = \overline{X}$. ∎

Proposition 4.2.7 If $\underline{A}$ is definite then $|\,\underline{A}\,|$ is a Scott domain.

Proof By proposition 4.2.5 and the countability of A we know $|\,\underline{A}\,|$ is $\omega-$algebraic.

Suppose S is a consistent subset of $|\,\underline{A}\,|$, *i.e.*, $\exists x \,.\, \forall s \in S \,.\, s \subseteq x$. Let

$$\overline{\bigcup S} = \{\, b \in A \mid \exists X \subseteq^{fin} \bigcup S \,.\, X \vdash \{b\}\,\}.$$

It is easy to see that $\overline{\bigcup S}$ is an element of $|\,\underline{A}\,|$. It is also obvious that $\overline{\bigcup S}$ is the (unique) least upper bound of S. Hence $|\,\underline{A}\,|$ is consistently complete. ∎

We now proceed to give an axiom which specifies the **SFP** objects.

Definition 4.2.4 A GIS $\underline{A}$ is called *strongly finite* if it satisfies the axiom of *finite closure*: for all $X \subseteq^{fin} A$ there is a finite super set $Y \supseteq X$ such that

$$Y \supseteq Y' \neq \emptyset \Longrightarrow \exists Y'' \subseteq Y \,.\, Y' \dashv\vdash Y''.$$

Recall $Y' \dashv\vdash Y''$ is an abbreviation for $Y' \vdash Y'' \& \forall b \in Y''.\ \{b\} \vdash^* Y'$. If we put in the logical operations explicitly and write $Y' = \{a_i \mid i \in I\}$ and $Y'' = \{b_j \mid j \in J\}$, $Y' \dashv\vdash Y''$ means $\bigwedge a_i \Leftrightarrow \bigvee b_j$. The axiom of strongly finiteness says that for any finite set of propositions there is a super set which has the property that any conjunction of

propositions of a subset of the super set is equivalent to a disjunction of propositions of some subset of the super set. This is more or less a restatement of quasi-conjunctive closedness of Gunter [Gu85].

Proposition 4.2.8 Let $\underline{A}$ be a strongly finite information system and let $X \dashv\vdash Y$. Then there is a $Y' \subseteq Y$ with $X \dashv\vdash Y'$, such that

$$\forall b,\ b' \in Y'.\ \{b\} \vdash \{b'\} \Longrightarrow b = b'.$$

Proof We can get such a Y' by repeatly using axiom 5 of definition 4.2.1: $X \vdash Y'$ and $\{b\} \vdash \{b'\}$ with $b,\ b' \in Y'$ implies $X \vdash Y' \setminus \{b\} \cup \{b'\} = Y' \setminus \{b\}$. Thus each such step reduces the number of elements of Y' by one if $b \neq b'$ and we must stop somewhere because Y' is finite (the argument for the case where Y is already an empty set is trivial). ∎

Suppose $X \dashv\vdash Z$ and $X \vdash \emptyset$. We have $X \vdash Z$ and $\{b\} \vdash^* X$ for all $b \in Z$. Therefore $\{b\} \vdash \emptyset$ for all $b \in Z$ because of axiom 3 of definition 4.2.1. But $\{b\} \nvdash \emptyset$; this implies $Z = \emptyset$.

We remark that if a GIS $\underline{A}$ is definite, it is not necessary strongly finite. However, if $\underline{A}$ is definite and *closed under conjunction* (see an extended version of [Sc82]), in the sense that $X \nvdash \emptyset$ implies $X \dashv\vdash \{\, a \,\}$ for some $a \in A$, then it is clearly also strongly finite.

In what follows we give a sequence of propositions to establish the result that SFISs give **SFP** objects. We need to use some results stated in Chapter 2 when doing so.

Proposition 4.2.9 Let $\underline{A}$ be a strongly finite information system. $|\ \underline{A}\ |$ is an ω−algebraic cpo with all its non-bottom finite elements being of the form $\overline{a}$, where $a \in A$.

Proof By Proposition 4.2.4 $\overline{a}$'s are finite elements. Suppose x is a finite element of $|\ \underline{A}\ |$. Clearly

$$x = \bigcup \{\, \overline{a} \mid a \in x \,\}.$$

Now we show that $\{\, \overline{a} \mid a \in x \,\}$ is directed. Let $a,\ b \in x$. By the axiom of finite closure there is some finite Y such that $\{\, a,\ b \,\} \vdash Y$, $\forall c \in Y.\ \{\, c \,\} \vdash \{\, a \,\}$ and $\{\, c \,\} \vdash \{\, b \,\}$. However, x is closed under entailment, so $\{\, a,\ b \,\} \vdash Y$ implies there is some $c_0 \in Y$ such that $c_0 \in x$. For this particular c_0 we also have $\{\, c_0 \,\} \vdash \{\, a \,\}$ and $\{\, c_0 \,\} \vdash \{\, b \,\}$. Hence $\overline{a} \subseteq \overline{c_0}$ and $\overline{b} \subseteq \overline{c_0}$, which means $\{\, \overline{a} \mid a \in x \,\}$ is directed. As x is a finie element there is some $\overline{a_0} \in \{\, \overline{a} \mid a \in x \,\}$ with $x \subseteq \overline{a_0}$, which is only possible when $x = \overline{a_0}$. We have shown that evey finite element of $|\ \underline{A}\ |$ is of the form $\overline{a}$.

That $|\underline{A}|$ is $\omega-$algebraic is then obvious. ∎

Recall in Chapter 2 we mentioned the notion of minimal upper bounds. For a cpo $(D, \sqsubseteq)$ and a subset $X \subseteq D$, d is a minimal upper bound of X if d is an upper bound of X and it is not strictly greater than any other upper bound of X. Write $MUB(X)$ for the set of minimal upper bounds of X. In general this set can be empty or infinite. $MUB(X)$ is said to be complete if whenever u is an upper bound of X then $u \sqsupseteq v$ for some $v \in MUB(X)$.

Proposition 4.2.10 Let $\underline{A}$ be a strongly finite information system. If S is a finite, consistent subset of finite elements of $|\underline{A}|$, then $MUB(S)$ is non-empty, finite and complete.

Proof Let S be a finite, consistent subset of finite elements of $|\underline{A}|$. We know, from Proposition 4.2.9 that S can be written as

$$S = \{\overline{a_1}, \overline{a_2} \cdots, \overline{a_n}\}$$

with a_i's in A. By the axiom of finite closure there is some finite Y such that $\{a_i \mid 1 \leq i \leq n\} \dashv\vdash Y$. Since S is consistent, *i.e.*,

$$\exists b \in A. \forall i. \overline{a_i} \subseteq \overline{b},$$

the above Y must be non-empty because $\overline{a_i} \subseteq \overline{b}$ implies $\{b\} \vdash \{a_i\}$ for each i, and $\{b\} \nvdash \emptyset$.

Let Y' be a subset of Y such that $\forall b,\ b' \in Y'.\ \{b\} \vdash \{b'\} \Rightarrow b = b'$. It is not difficult to check that

$$MUB(S) = \{\overline{b} \mid b \in Y'\},$$

which implies that $MUB(S)$ is finite and non-empty.

$MUB(S)$ is complete because assuming $x \supseteq s$ for all $s \in S$ we have

$$x \supseteq \{a_i \mid 1 \leq i \leq n\}.$$

By closedness of x under entailment there is some $b \in Y$ with $b \in x$. Therefore $x \supseteq \overline{b}$. ∎

For a subset X of finite elements of a cpo we can recursively apply the MUB (operator) to the consistent subsets of X and join the minimal upper bounds to the set just produced. The collection of all such elements is written as $U^*(X)$ (for more detail see Chapter 2).

Proposition 4.2.11 Let $\underline{A}$ be a strongly finite information system. If S is a finite subset of finite elements of $|\underline{A}|$, then $U^{\star}(S)$ is finite.

Proof Let S be a finite, consistent subset of finite elements of $|\underline{A}|$. We know, from Proposition 4.2.9 again, S can be written as

$$S = \{\overline{a_1}, \overline{a_2} \cdots, \overline{a_n}\}$$

with a_i's in A. By the axiom of finite closure there is some finite $R \supseteq \{a_i \mid 1 \leq i \leq n\}$ such that

$$R \supseteq X \neq \emptyset \Longrightarrow \exists Y \subseteq R \,.\, X \dashv\vdash Y.$$

By Proposition 4.2.10, $\{\overline{a} \mid a \in R\}$ is a MUB–closed set. Therefore

$$U^{\star}(S) \subseteq \{\overline{a} \mid a \in R\}$$

and $U^{\star}(S)$ is finite. ∎

Theorem 4.2.1 If $\underline{A}$ is a strongly finite information system then $|\underline{A}|$ is an **SFP** object.

Proof By Proposition 4.2.10, Proposition 4.2.11 and Theorem 2.5.1 ∎

On the other hand, an **SFP** object determines a strongly finite information system in the way described below.

Definition 4.2.5 Let D be an **SFP** object. Define

$$IS(D) = (\mathbf{P\Omega}(D), \vdash)$$

where $\mathbf{P\Omega}(D)$ is the set $\{d{\uparrow} \mid d \in D^0 \setminus \{\perp_D\}\}$ and

$$X \vdash Y \text{ iff } \bigcap X \subseteq \bigcup Y.$$

Note that the notation $\uparrow$ was introduced in Chapter 2 for finite elements and sets of finite elements, with different interpretations. Notice how the idea of open sets as properties about computation is reflected in our definition: the propositions of the information system $IS(D)$ consists of the prime open sets of the Scott topology on D.

Proposition 4.2.12 Let D be an **SFP** objects. Then $IS(D)$ is an SFIS.

Proof It is routine to show that $IS\,(D)$ is a GIS. When $\bigcap X = \bigcup Y$ we have both $\bigcap X \subseteq \bigcup Y$ and $b \subseteq \bigcup Y \subseteq a$ for any $a \in X$, $b \in Y$, *i.e.*, $X \dashv\vdash Y$. Hence the axiom of finite closure follows from the quasi-conjunctive closedness of **SFP** objects. ∎

Definition 4.2.6 Two SFISs $\underline{A} = (A_{\underline{A}}, \vdash_{\underline{A}})$ and $\underline{B} = (B_{\underline{B}}, \vdash_{\underline{B}})$ are said to be isomorphic if there is a bijection $\beta : (A/{=_{\underline{A}}}) \to (B/{=_{\underline{B}}})$ such that

$$X \vdash_{\underline{A}} Y \text{ iff } \beta X \vdash_{\underline{B}} \beta Y,$$

where $\beta Z = \bigcup_{c \in C}\{\, a \mid a =_{\underline{A}} \beta(c)\,\}$, $A/{=_{\underline{A}}}$ is the quotient set and $a =_{\underline{A}} b$ is an abbreviation for $\{a\} \vdash_{\underline{A}} \{b\}$ and $\{b\} \vdash_{\underline{A}} \{a\}$.

It is obvious that if $\underline{A}$ and $\underline{B}$ are isomorphic then $|\,\underline{A}\,|$ and $|\,\underline{B}\,|$ are isomorphic.

Proposition 4.2.13 For any SFIS $\underline{A}$, $IS\,(\,|\,\underline{A}\,|\,)$ is isomorphic to $\underline{A}$, and for any **SFP** object D, $|\,IS\,(D)\,|$ is isomorphic to D.

Proof We give the isomorphism pairs. It is routine to check that they are indeed isomorphisms. The first pair is (θ_1, ϕ_1), where

$\theta_1 : D \to |\,IS\,(D)\,|$ is given by $e \longmapsto \{\, d\!\uparrow \mid d \sqsubseteq e \;\&\; d \in D^0 \setminus \{\perp_D\}\,\}$,

$\phi_1 : |\,IS\,(D)\,| \to D$ is given by $x \longmapsto \bigsqcup\{\, e \mid e\!\uparrow \in x\,\}$,

and the second pair is (θ_2, ϕ_2), where

$\theta_2 : \underline{A} \to IS\,(\,|\,\underline{A}\,|\,)$ is given by $a \longmapsto \overline{a}\!\uparrow$,

$\phi_2 : IS\,(\,|\,\underline{A}\,|\,) \to \underline{A}$ is given by $\overline{a}\!\uparrow \longmapsto a$.

∎

For information systems in [Sc82] and [LaWi84], if one starts from $\underline{A}$ and gets back an information system from $|\,\underline{A}\,|$, $\underline{A}$ and $IS\,(\,|\,\underline{A}\,|\,)$ need not be isomorphic in our sense. Consider, for example, the information system $(\{0, 1\}, \mathit{Con}, \vdash)$, where Con is th consistency predicate such that $\{0, 1\} \in \mathit{Con}$ and $\vdash$ is the trivial entailment $\{0, 1\} \vdash 0$ and $\{0, 1\} \vdash 1$. The strongly finite information systems here are canonical, in the sense that the propositions exactly correspond to the non-bottom finite elements of the **SFP** object so that $\underline{A}$ and $IS\,(\,|\,\underline{A}\,|\,)$ are isomorphic.

4.3 A Category of Information Systems

We introduce morphisms on SFISs called approximable mappings. This makes them a category. Approximable mappings show how information systems are related to one

another and they correspond to continuous functions between the associated **SFP** objects. The way approximable mappings defined is slightly different from the traditional one. The canonical character of the strongly finite information system makes it possible to specify approximable mappings as relations on propositions rather than on consistent sets.

Definition 4.3.1 Let $\underline{A} = (A, \vdash_{\underline{A}})$, $\underline{B} = (B, \vdash_{\underline{B}})$ be SFISs. An approximable mapping from $\underline{A}$ to $\underline{B}$ is a relation $R \subseteq A \times B$ which satisfies

1. $\{a\} \vdash_{\underline{A}} \{a'\}$ & $a' R b'$ & $\{b'\} \vdash_{\underline{B}} \{b\} \Rightarrow a R b$
2. $(\forall i \in I.\ a_i R b_i$ & $\{a_i \mid i \in I\} \dashv\vdash_{\underline{A}} X$ & $\{b_i \mid i \in I\} \dashv\vdash_{\underline{B}} Y)$
   $\Longrightarrow \forall a'' \in X \exists b'' \in Y.\ a'' R b''$

Proposition 4.3.1 Strongly finite informations systems with approximable mappings form a category, written **SFIS**.

Proof Identities are given by $a\, Id_{\underline{A}}\, b$ iff $\{a\} \vdash_{\underline{A}} \{b\}$. We check that approximable mappings compose. Other axioms for a category can be checked similarly.

Let $\underline{A}$, $\underline{B}$ and $\underline{C}$ be SFIS s and $R : \underline{A} \to \underline{B}$ and $S : \underline{B} \to \underline{C}$ be approximable mappings. Let $R \circ S$ be the relational composition. We show that $R \circ S$ satisfies the axioms for approximable mappings.

Axiom 1 in definition 4.3.1 obviously holds for $R \circ S$. To see that $R \circ S$ satisfies axiom 2 suppose, for a finite set I, $\forall i \in I.\ a_i\, R \circ S\, c_i$ and $\{a_i \mid i \in I\} \dashv\vdash_{\underline{A}} X$, $\{c_i \mid i \in I\} \dashv\vdash_{\underline{C}} Z$. There exist $u_i \in B$ such that $a_i R u_i$, $u_i R c_i$ for any $i \in I$. Let $\{u_i \mid i \in I\} \dashv\vdash_{\underline{B}} Y$. By axiom 2 $\forall p \in X \exists q \in Y.\ p R q$. For this q we have $q S c_i$ for all $i \in I$ by axiom 1, which implies the existence of some $r \in Z$ such that $q S r$, since $\{q\} \dashv\vdash \{q\}$. Therefore $p R \circ S r$. ∎

Proposition 4.3.2 Let R be an approximable mapping between SFISs $\underline{A}$ and $\underline{B}$. Define $f_R : |\underline{A}| \to |\underline{B}|$ by

$$f_R(x) = \{b \in B \mid \exists a \in x.\ a R b\}.$$

f_R is a continuous function in $|\underline{A}| \to |\underline{B}|$.

Proof Let $x \in |\underline{A}|$ and $R : \underline{A} \to \underline{B}$ be an approximable mapping. To show $f_R(x) \in |\underline{B}|$ let $Y \subseteq f_R(x)$ and $Y \vdash_{\underline{B}} Z$. For each $b \in Y$ there is some $a \in x$ such that $a R b$. Write X for such a collection of a's. Because $\underline{A}$ and $\underline{B}$ are strongly finite there are

X', Y' such that $X \dashv\vdash_{\underline{A}} X'$ and $Y \dashv\vdash_{\underline{B}} Y'$. $X \subseteq x$ and $X \vdash_{\underline{A}} X'$, implies $X' \cap x \neq \emptyset$. Let $u_0 \in X' \cap x$. By axiom 2 of definition 4.3.1 there is $v_0 \in Y'$ such that $u_0 \, R \, v_0$. Also we have $\{ v_0 \} \vdash_{\underline{B}} Z$, which implies $\{ v_0 \} \vdash_{\underline{B}} \{ c \}$ for some $c \in Z$. Therefore $c \in f_R(x)$.

The monotonicity of f_R is obvious. It also preserves sups of directed sets of $|\underline{A}|$; for assuming $b \in f_R(\bigcup P)$, where P is a directed subset of $|\underline{A}|$, there is some $a \in \bigcup P$ with $a \, R \, b$. Therefore there is some $y \in P$ such that $a \in y$, which means $b \in f_R(y)$. Hence

$$f_R(\bigcup P) \subseteq \bigcup \{ f_R(y) \mid y \in P \},$$

enough for the equality to hold. ∎

Theorem 4.3.1 **SFIS** is equivalent to the category **SFP**.

Proof Let F : **SFIS** $\rightarrow$ **SFP** be a functor such that

$$F(\underline{A}) = |\underline{A}|$$
$$F(R) = f_R.$$

We use one of MacLane's results in [Ma71] (pp 91). It is enough to show that F is full and faithful, and each **SFP** object D is isomorphic to $F(\underline{A})$ for some SFIS $\underline{A}$.

That each **SFP** object D is isomorphic to $F(\underline{A})$ for some SFIS $\underline{A}$ is routine. It remains to show that F is full and faithful. First we show that F is full.

Suppose $\underline{A}$ and $\underline{B}$ are SFISs and

$$f : F(\underline{A}) \rightarrow F(\underline{B})$$

a continuous function. Define a relation $R \subseteq A \times B$ by letting $a \, R \, b$ if $b \in f(\overline{a})$. We check that this relation is an approximable mapping form $\underline{A}$ to $\underline{B}$. The first condition in Definition 4.3.1 holds trivially. Let $\{ a_i \mid i \in I \}$ $\{ b_i \mid i \in I \}$ be finite sets such that for any $i \in I$, $a_i \, R \, b_i$. Suppose

$$\{ a_i \mid i \in I \} \dashv\vdash_{\underline{A}} X$$

and

$$\{ b_i \mid i \in I \} \dashv\vdash_{\underline{B}} Y.$$

For any $a \in X$, $\{ a \} \vdash^* \{ a_i \mid i \in I \}$. Thus we have $b_i \in f(\overline{a_i}) \subseteq f(\overline{a})$ for any $i \in I$. Now $\{ b_i \mid i \in I \} \vdash_{\underline{B}} Y$. Therefore $f(\overline{a}) \cap Y \neq \emptyset$. This means for some $b \in Y$, $b \in f(\overline{a})$, or $a \, R \, b$.

We now show that the continuous function f_R determined by R is actually equal to f. Let $x \in |\underline{A}|$. Suppose $b \in f_R(x)$. By definition there is some $a \in x$, $a \, R \, b$. That is,

$b \in f(\overline{a})$. Therefore $b \in f(x)$, by the monotonicity of f. Thus $f_R(x) \subseteq f(x)$. On the other hand, let $b \in f(x)$. By the continuity of f there is some $a \in x$ such that $b \in f(\overline{a})$. Hence $a\,R\,b$ and $b \in f_R(x)$. This means $f(x) \subseteq f_R(x)$. Hence $f = f_R$.

Secondly, we show that F is faithful. Suppose $R,\ S : \underline{A} \to \underline{B}$ are approximable mappings such that $f_R = f_S$. Let $a\,R\,b$. Then $b \in f_R(\overline{a}) = f_S(\overline{a})$. This means for some $a' \in \overline{a}$, $a'\,S\,b$. By the first condition in Definition 4.3.1 $a\,S\,b$ as S is an approximable mapping. Therefore, $R \subseteq S$. By symmetry, $S \subseteq R$ and hence $R = S$. ∎

Now we consider construction on SFIS. In this section we introduce constructions of lifting $(\ \ )_\uparrow$, sum $+$, product $\times$ and function space $\to$, with an emphasis on $\to$. The powerdomain constructions are given in the next section. Lifting, sum and product are more or less the same as those on information systems[LaWi84].

Definition 4.3.2 (Lifting) Let $\underline{A} = (A,\ \vdash)$ be an SFIS. Define the *lift* of $\underline{A}$ to be $\underline{A}_\uparrow = (A',\ \vdash')$ where

- $A' = (\{0\} \times A) \cup \{0\}$
- $X \vdash' Y \Leftrightarrow [\,0 \in Y \text{ or } \{c \mid (0,\ c) \in X\} \vdash_{\underline{A}} \{b \mid (0,\ b) \in Y\}\,]$

Lifting is an operation which given an SFIS produces a new one by joining a new proposition weaker than all the old ones.

Definition 4.3.3 (Sum) Let $\underline{A} = (A,\ \vdash_{\underline{A}})$ and $\underline{B} = (B,\ \vdash_{\underline{B}})$ be SFISs. Define their *sum*, $\underline{A} + \underline{B}$, to be $\underline{C} = (C,\ \vdash)$ where

- $C = \{0\} \times A \cup \{1\} \times B$
- $X \vdash Y \Leftrightarrow X = \{(0,\ a) \mid a \in X_0\}\ \&\ X_0 \vdash_{\underline{A}} \{r \mid (0,\ r) \in Y\}$ or

  $X = \{(1,\ b) \mid b \in X_1\}\ \&\ X_1 \vdash_{\underline{B}} \{s \mid (1,\ s) \in Y\}$ or

  $X \cap (\{0\} \times A) \neq \emptyset\ \&\ X \cap (\{1\} \times B) \neq \emptyset$

The effect of sum is to juxtaposing disjoint copies of two SFISs. We can obtain the separated sum $\oplus$ by letting $\underline{A} \oplus \underline{B} =^{\text{def}} \underline{A}_\uparrow + \underline{B}_\uparrow$.

Definition 4.3.4 (Product) Let $\underline{A} = (A,\ \vdash_{\underline{A}})$ and $\underline{B} = (B,\ \vdash_{\underline{B}})$ be SFISs. Define

their *product*, $\underline{A} \times \underline{B}$, to be $\underline{C} = (C, \vdash)$ where

- $C = \{(a, *) \mid a \in A\} \cup \{(*, b) \mid b \in B\} \cup \{(a, b) \mid a \in A \;\&\; b \in B\}$

- $X \vdash Y \Leftrightarrow \exists c \in Y.\, (\pi_0 X = \{*\} \Rightarrow \pi_0 c = *)\;\&$
  $(\pi_1 X = \{*\} \Rightarrow \pi_1 c = *)\;\&$
  $\pi_0 X \neq \{*\} \Rightarrow (\pi_0 c = * \text{ or } [\pi_0 X \setminus \{*\}] \vdash_{\underline{A}} \pi_0 c)\;\&$
  $\pi_1 X \neq \{*\} \Rightarrow (\pi_1 c = * \text{ or } [\pi_1 X \setminus \{*\}] \vdash_{\underline{B}} \pi_1 c)$

The symbol $*$ here acts like a proposition which is always true. Note that to have enough propositions for the product the disjoint union of A and B or the set of pairs $A \times B$ are not sufficient; The use of disjoint union makes $\{(0, a), (1, b)\}$ a non $\dashv\vdash$ −closed set while the use of $A \times B$ misses out those points corresponding to $(\emptyset, y)$ and $(x, \emptyset)$, which are in $|\underline{A}| \times |\underline{B}|$. The fact that we get enough propositions by introducing $*$ is shown in Theorem 4.3.2.

Definition 4.3.5 For a SFIS $\underline{A} = (A, \vdash)$, $X \subseteq^{fin} A$ is said to be $\dashv\vdash_{\underline{A}}$ −closed if

$$(X' \subseteq X \;\&\; X' \neq \emptyset) \Longrightarrow \exists X'' \subseteq X\,.\, X' \dashv\vdash_{\underline{A}} X''.$$

Thus the axiom of finite closure says every finite set of a SFIS has a finite super set which is $\dashv\vdash_{\underline{A}}$ −closed.

Theorem 4.3.2 If $\underline{A}$, $\underline{B}$ are SFISs then so are $\underline{A}_{\uparrow}$, $\underline{A}+\underline{B}$, and $\underline{A}\times\underline{B}$. Furthermore,

$$x \in |\underline{A}|_{\uparrow} \Longleftrightarrow (x = \emptyset \text{ or } \exists y \in |\underline{A}|\,.\, x = \{0\} \cup (\{0\} \times y)),$$

$$x \in |\underline{A} + \underline{B}| \Longleftrightarrow (\exists x_0 \in |\underline{A}|\,.\, x_0 = \{a \mid (0, a) \in x\}) \text{or}$$
$$(\exists x_1 \in |\underline{B}|\,.\, x_1 = \{b \mid (1, b) \in x\}),$$

$$x \in |\underline{A} \times \underline{B}| \Longleftrightarrow (a, b) \in x \Rightarrow (a, *), (*, b) \in x\;\&$$
$$\{a \in A \mid \exists b \in B.\, (a, b) \in x\} \in |\underline{A}|\;\&$$
$$\{b \in B \mid \exists a \in A.\, (a, b) \in x\} \in |\underline{B}|\,.$$

Proof It is routine to show that $\underline{A}_{\uparrow}$, $\underline{A} + \underline{B}$, and $\underline{A} \times \underline{B}$ are GISs. To verify the finite closure axiom we need to produce $\dashv\vdash$ −closed sets of propositions. The rules given below indicate how to get them.

$$P \;\dashv\vdash_{\underline{A}} \text{−closed} \;\&\; \{a \mid (0, a) \in W\} \subseteq P$$

$$\Longrightarrow W \subseteq \Sigma_0,$$

where $\Sigma_0 = \{(0, a) \mid a \in P\}\;\&\;\{(0, a) \mid a \in P\}$ and it is $\dashv\vdash_{\underline{A}_{\uparrow}}$ −closed;

$$P, Q \;\dashv\vdash \text{−closed} \;\&\; \{a \mid (0, a) \in W\} \subseteq P \;\&\; \{a \mid (1, a) \in W\} \subseteq Q$$

$$\Longrightarrow W \subseteq \Sigma_1,$$

where $\Sigma_1 = \{ (0, a) \mid a \in P \} \cup \{ (1, a) \mid a \in Q \}$, which is $\dashv\vdash_{\underline{A}+\underline{B}}$ −closed;

$$\left.\begin{array}{l} P, Q \dashv\vdash \text{-closed} \\ \{ a \in A \mid (a, *) \text{ or } (a, b) \in W \} \subseteq P \\ \{ b \in B \mid (*, b) \text{ or } (a, b) \in W \} \subseteq Q \end{array}\right\} \Longrightarrow W \subseteq \Sigma_2,$$

where $\Sigma_2 = \{ (a, *) \mid a \in P \} \cup \{ (*, b) \mid b \in Q \} \cup \{ (a, b) \mid a \in P \ \& \ b \in Q \}$, which is $\dashv\vdash_{\underline{A}\times\underline{B}}$ −closed.

The first two rules are obviously valid. To check the third rule let $W' \subseteq \Sigma_2$ We show that there is some $W'' \subseteq \Sigma_2$ such that $W' \dashv\vdash_{\underline{A}\times\underline{B}} W''$. This is trivial when there is no elements of the form $(a, *)$ or (a, b) in W', and similarly for the case when there is no elements of the form $(*, b)$ or (a, b) in W'. So suppose

$$\{ a \in A \mid (a, *) \in W' \text{ or } \exists b \in B. (a, b) \in W' \} \neq \emptyset$$

and

$$\{ b \in B \mid (*, b) \in W' \text{ or } \exists a \in A. (a, b) \in W' \} \neq \emptyset.$$

Since P and Q are $\dashv\vdash$ −closed, there are $P' \subseteq P$, $Q' \subseteq Q$ such that

$$\{ a \in A \mid (a, *) \in W' \text{ or } \exists b \in B. (a, b) \in W' \} \dashv\vdash_{\underline{A}} P'$$

and

$$\{ b \in B \mid (*, b) \in W' \text{ or } \exists a \in A. (a, b) \in W' \} \dashv\vdash_{\underline{B}} Q'.$$

Let $W'' = \{ (a, b) \mid a \in P' \ \& \ b \in Q' \}$. Clearly $W' \dashv\vdash_{\underline{A}\times\underline{B}} W''$.

It is routine to show the second part of the theorem. ∎

From this theorem it is easy to see that there is an isomorphism between $|\underline{A} + \underline{B}|$, $|\underline{A}| + |\underline{B}|$; and $|\underline{A} \times \underline{B}|$, $|\underline{A}| \times |\underline{B}|$. Thus it justifies our definition of product and sum, since **SFIS** is equivalent to **SFP**.

Definition 4.3.7 Let $\underline{A} = (A, \vdash_{\underline{A}})$ and $\underline{B} = (B, \vdash_{\underline{B}})$ be SFISs. An element m of Fin $(A \times B)$ is said to be a *molecule* if

- $\pi_0 m$ is $\dashv\vdash_{\underline{A}}$ −closed,
- $(\alpha, \beta \in m \ \& \ \pi_0\alpha \vdash_{\underline{A}} \pi_0\beta) \Rightarrow \pi_1\alpha \vdash_{\underline{B}} \pi_1\beta$.

A molecule corresponds to a finite element in function space by Theorem 2.10.1.

Definition 4.3.8 (*Function space*) Let $\underline{A} = (A, \vdash_{\underline{A}})$ and $\underline{B} = (B, \vdash_{\underline{B}})$ be SFISs. Their function space, $\underline{A} \to \underline{B}$, is the information system $\underline{C} = (C, \vdash,)$ where

- $C = \{ m \mid m \text{ is a molecule in Fin } (A \times B) \}$
- $X \vdash Y \Leftrightarrow (\forall m' \in X. \{ m \} \vdash \{ m' \}) \Rightarrow \exists m'' \in Y. \{ m \} \vdash \{ m'' \}$

where $\{ m \} \vdash \{ m' \} \Leftrightarrow \forall \alpha' \in m' \exists \alpha \in m. \{ \pi_0 \alpha' \} \vdash_{\underline{A}} \{ \pi_0 \alpha \} \ \& \ \{ \pi_1 \alpha \} \vdash_{\underline{B}} \{ \pi_1 \alpha' \}$

There is a general guideline according to which we can test whether a definition of entailment is correct or not. The notation $X \vdash Y$ should mean

$$\bigcap \{ \overline{a}\uparrow \mid a \in X \} \subseteq \bigcup \{ \overline{b}\uparrow \mid b \in Y \},$$

a relation between compact open sets. In other words, for any point x above all $\overline{a}$ with $a \in X$, x should be above some point $\overline{b}$, $b \in Y$. Since the domains concerned are algebraic, this x can be chosen from the set of finite elements, which, by proposition 4.2.4, are of the form $\overline{a}$ with a a proposition.

Let us accept that it is reasonable to have, for m, $m' \in C$,

$$\{ m \} \vdash \{ m' \} \Leftrightarrow$$

$$\forall \alpha' \in m' \exists \alpha \in m. \{ \pi_0 \alpha' \} \vdash_{\underline{A}} \{ \pi_0 \alpha \} \ \& \ \{ \pi_1 \alpha \} \vdash_{\underline{B}} \{ \pi_1 \alpha' \}$$

Therefore,

$$(\forall m' \in X. \{ m \} \vdash \{ m' \}) \Rightarrow \exists m'' \in Y. \{ m \} \vdash \{ m'' \}$$

is just a direct translation of the corresponding topological situation which we want to describe. It has, however, the unwelcome feature that it contains a universal quantifier over propositions in C. Although a more local way to give the entailment is possible (after we see the proof of proposition 4.3.3), it will not be very simple, and I don't know any other better way to do it at the moment.

The construction of function space is a bit complicated. Unlike Scott's information system where one can use (X, Y) as propositions for the function space, we have to use a more complicated form of propositions. The reason for not being able to use a simpler form of propositions for the function space construction can be illustrated by the following example.

Example 4.3.1 Consider a SFIS $\underline{A} = (A, \vdash)$, where

$$A = \{a, b, c, d\},$$

$\vdash$ is given by

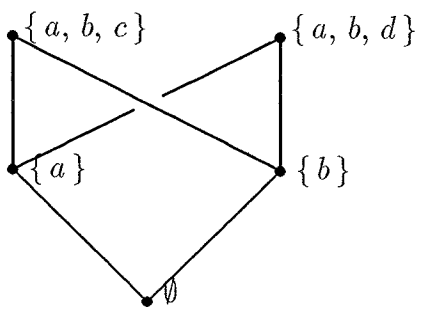
$$\{a, b\} \vdash \{c, d\},$$

$$\{c\} \vdash \{a\}\ ,\ \{c\} \vdash \{b\},$$

$$\{d\} \vdash \{a\}\ ,\ \{d\} \vdash \{b\},$$

$$\{a, b, c, d\} \vdash \emptyset.$$

This SFIS represents the **SFP** object pictured as



Now consider the function space construction of $\underline{A}$ to itself. If one were to use propositions of the form (u, v) with $u, v \in A$ and $\{(u, v)\} \vdash \{(u', v')\}$ iff $\{u'\} \vdash \{u\}$ and $\{v\} \vdash \{v'\}$ (which seems reasonable), we would only have an entailment

$$\{(a, a), (b, b)\} \vdash \{(c, c), (c, d), (d, c), (d, d)\}$$

but without any Y such that

$$\{(a, a), (b, b)\} \dashv\vdash Y,$$

i.e., the resulting information system would not be strongly finite.

We have the following propositions to show that the function space construction in definition 4.3.4 works.

Proposition 4.3.3 If $\underline{A}$ and $\underline{B}$ are strongly finite then $\underline{A} \to \underline{B}$ is strongly finite.

Proof First $\underline{A} \rightarrow \underline{B}$ is a GIS. This can be shown by checking axioms 1 to 6 in definition 4.2.1. As an example we check axiom 5. Suppose $X \vdash Y$, $Y' \subseteq Y$, and $\forall b \in Y'. \{ b \} \vdash Z$. Since $X \vdash Y$, for any m such that $\forall m' \in X. \{ m \} \vdash \{ m' \}$ there is some m'' in Y for which $\{ m \} \vdash \{ m'' \}$. If this m'' is in $Y \setminus Y'$ then we are done, otherwise m'' must be in Y'. However $\{ m'' \} \vdash Z$, which means $\{ m'' \} \vdash \{ n \}$ for some $n \in Z$. Clearly $\{ m \} \vdash \{ n \}$. Thus $X \vdash (Y \setminus Y') \cup Z$.

To show that $\underline{A} \rightarrow \underline{B}$ satisfies the finite closure axiom in definition 4.2.4 let $X \subseteq^{fin} C$. Since $\underline{A}$ and $\underline{B}$ are strongly finite we know that there are P, Q, $\dashv\vdash_{\underline{A}}$ and $\dashv\vdash_{\underline{B}}$ closed, respectively, with $P \supseteq \bigcup\{ \pi_0 m \mid m \in X \}$ and $Q \supseteq \bigcup\{ \pi_1 m \mid m \in X \}$. We claim that the set

$$\Sigma = \{ m \in C \mid \pi_0 m \subseteq P \ \& \ \pi_1 m \subseteq Q \}$$

is $\dashv\vdash$ –closed and contains X. Σ is finite because P and Q are. X is obviously a subset of Σ. Given $Y' \subseteq \Sigma$. If there is no $m \in C$ such that $\{ m \} \vdash \{ m' \}$ for all $m' \in Y'$, then it is clear that we have $Y' \dashv\vdash \emptyset$. So suppose for some $m \in C$, $\{ m \} \vdash \{ m' \}$ for all $m' \in Y'$. We perform the following steps.

Step 1. Form a list ℓ with $\bigcup Y'$ as its head.

Step 2. Repeat the following until for each member L of the list ℓ, $\pi_0 L$ is $\dashv\vdash$ –closed. For each L in the list, for $L' \subseteq L$, let $s \subseteq P$ be such that $\pi_0 L' \dashv\vdash_{\underline{A}} s$. If $s \not\subseteq \pi_0 L$ then add, for each f,

$$L \cup \{ (a, f(a)) \mid a \in (s \setminus \pi_0 L) \}$$

to ℓ, where f is a function from $(s \setminus \pi_0 L)$ to t, with $t \subseteq Q$ satisfying $\pi_1 L' \dashv\vdash_{\underline{B}} t$. Remove L from the list.

Step 3. Repeat this step until every member L in list ℓ has the property that

$$\forall (a, b), (a', b') \in L. \{ a \} \vdash_{\underline{A}} \{ a' \} \Rightarrow \{ b \} \vdash_{\underline{B}} \{ b' \}.$$

For any element L of the list, for any (a, b), $(a', b') \in L$ with $\{ a \} \vdash \{ a' \}$, let $t \subseteq Q$ be such that $\{ b, b' \} \dashv\vdash_{\underline{B}} t$. Add, for each $b_i \in t$,

$$(L \setminus \{ (a, b) \}) \cup \{ (a, b_i) \}$$

to ℓ. Remove L from the list.

This algorithm is guaranteed to stop in finitely many steps because P and Q are finite. We now check that at each step of the algorithm the following two properties hold:

$$(\forall m' \in Y'.\ \{m\} \vdash \{m'\})$$

$$\Longrightarrow \exists L \in \ell.\ (\forall \alpha \in L \exists \beta \in m.\ \{\pi_0\alpha\} \vdash \{\pi_0\beta\}\ \&\ \{\pi_1\beta\} \vdash \{\pi_1\alpha\})$$

and

$$L \in \ell\ \&\ m' \in Y \Rightarrow \forall \alpha \in m' \exists \beta \in L.\ (\{\pi_0\alpha\} \vdash \{\pi_0\beta\}\ \&\ \{\pi_1\beta\} \vdash \{\pi_1\alpha\}).$$

As an illustraction we show that Step 2 preserves the first property. Following the notation used in Step 2 it is enough to show that if

$$\forall \alpha \in L \exists \beta \in m.\ \{\pi_0\alpha\} \vdash \{\pi_0\beta\}\ \&\ \{\pi_1\beta\} \vdash \{\pi_1\alpha\}$$

then for some f,

$$\forall a \in (s \setminus \pi_0 L) \exists \beta \in m.\ \{a\} \vdash \{\pi_0\beta\}\ \&\ \{\pi_1\beta\} \vdash \{f(a)\}.$$

This f is specified as follows. Given $a \in (s \setminus \pi_0 L)$, we have $\{a\} \vdash^* \pi_0 L'$. Clearly, then, $\{a\} \vdash^* \pi_0 L'_m$, where L'_m is a collection of the associated β's in m such that

$$\forall \alpha \in L' \exists \beta \in L'_m.\ \{\pi_0\alpha\} \vdash \{\pi_0\beta\}\ \&\ \{\pi_1\beta\} \vdash \{\pi_1\alpha\}.$$

Let $M \subseteq \bigcup m$ and $\pi_0 L'_m \dashv\vdash \pi_0 M$. There must be some $\beta \in M$ such that $\{a\} \vdash \{\pi_0\beta\}$ ($\vdash^* \pi_0 L'_m$). Since m is a molecule, $\{\pi_1\beta\} \vdash^* \pi_1 L'_m$. Obviously $\pi_1 L'_m \vdash \{e\}$ for all $e \in \pi_1 L'$. Hence $\{\pi_1\beta\} \vdash t$, which implies $\{\pi_1\beta\} \vdash \{b\}$ for some $b \in t$. Assign b to $f(a)$. This shows the existence of f. Similarly one can prove that the first property is preserved by Step 3 and the second property is preserved by Step 2 and 3.

Therefore $Y' \dashv\vdash Z$, where Z is the collection of molecules got after the algorithm stops, which is clearly a subset of Σ. The role which the two properties stated above play is just to guarantee the $\dashv\vdash$ relation. ∎

We can extract from the proof a rule which tells us how to get $\dashv\vdash$ $-$closed sets in the function space if we know how to get them for the domain and the codomain:

$$P,\ Q\ \dashv\vdash -\text{closed}\ \&\ \bigcup\{\pi_0 m \mid m \in F\} \subseteq P\ \&\ \bigcup\{\pi_1 m \mid m \in F\} \subseteq Q$$
$$\Longrightarrow \{m \in C \mid \pi_0 \subseteq P\ \&\ \pi_1 m \subseteq Q\} \text{ is } \dashv\vdash -\text{closed}$$

As SFISs represent **SFP** objects, the proposition below shows that the function space of **SFP** objects is still **SFP**.

Proposition 4.3.4 If $\underline{A}$ and $\underline{B}$ are information systems then $|\,\underline{A} \to \underline{B}\,|$ is isomorphic to $|\,\underline{A}\,| \to |\,\underline{B}\,|$.

Proof Use Theorem 2.10.1. ∎

4.4 Powerdomain Constructions

This section proposes the Hoare, the Smyth, and the Plotkin powerdomain constructions on SFISs.

Definition 4.4.1 Let $\underline{A}$ be a strongly finite information system. The Hoare powerdomain of $\underline{A}$ is the information system $P_H\underline{A} = (C, \vdash)$ where

- $C = \mathrm{Fin}(A) \setminus \{\emptyset\}$
- $X \vdash Y$ iff $\exists \beta \in Y.\, (\forall b \in \beta\, \exists a \in \bigcup X.\, \{a\} \vdash_{\underline{A}} \{b\})$

According to the definition, $\{\alpha\} \vdash_{P_H\underline{A}} \{\beta\}$ iff $\forall b \in \beta \exists a \in \alpha.\, \{a\} \vdash_{\underline{A}} \{b\}$. Therefore $\vdash_{P_H\underline{A}}$ on C is a reverse of the preorder $\sqsubseteq_{\approx_1}$ introduced in Chapter 2 on the finite sets of finite elements of $|\,\underline{A}\,|$.

We want $X \dashv\vdash Y$ to characterise, the situation

$$\bigcap\{\overline{\alpha}\uparrow \mid \alpha \in X\} \subseteq \bigcup\{\overline{\beta}\uparrow \mid \beta \in Y\}.$$

This is equivalent to saying that whenever $\{\alpha_0\} \vdash \{\alpha\}$ for all $\alpha \in X$, $\{\alpha_0\} \vdash \{\beta_0\}$ for some $\beta_0 \in Y$. Write $\vdash'$ for the associated entailment. Clearly $X \vdash' Y$ is equivalent to $X \vdash_{P_H\underline{A}} Y$ since we have $X \dashv\vdash_{P_H\underline{A}} \{\bigcup X\}$.

We can think of $\alpha \in C$ as a logical formula $\bigwedge\{\Diamond a \mid a \in \alpha\}$, where intuitively a set S of processes satisfies $\Diamond a$ if there exists $p \in S$, p satisfies a. Then we have, for example,

$$\left(\bigwedge\{\Diamond a \mid a \in \alpha\}\right) \wedge \left(\bigwedge\{\Diamond b \mid b \in \beta\}\right) \Leftrightarrow \bigwedge\{\Diamond c \mid c \in \alpha \cup \beta\},$$

in the sense that a set of processes satisfies the proposition on the left hand side iff it satisfies the proposition on the right hand side.

Proposition 4.4.1 If $\underline{A}$ is a strongly finite information system then so is $P_H\underline{A}$.

Proof It is clearly a GIS. Suppose $X \subseteq^{fin} C$. Let $\beta = \bigcup X$. Then $\beta \in C$, $X \vdash \{\beta\}$, and $\forall \alpha \in X\ \{\beta\} \vdash \{\alpha\}$ according to the definition of entailment. Therefore $X \dashv\vdash \{\beta\}$, from which it is easy to see that the finite closure axiom holds. ∎

It is easy to see that if X is a finite set of propositions of $P_H\underline{A}$, then $\{ H \mid H \subseteq \bigcup X \}$ is always a $\dashv\vdash$ –closed set which contains X.

The definition of entailment for $P_H\underline{A}$ implies that, as explained earlier, if $X \vdash Y$ then there is some $\beta \in Y$ such that $X \vdash \{\beta\}$. Thus $P_H\underline{A}$ is actually *definite*. This indicates that $|\, P_H\underline{A} \,|$ is always a Scott domain.

Proposition 4.4.2 $|\, P_H\underline{A} \,|$ is isomorphic to $\mathcal{P}_H \; |\, \underline{A} \,|$.

Proof It is enough to establish an order preserving one-one correspondence between the finite elements of $|\, P_H\underline{A} \,|$ and $\mathcal{P}_H \; |\, \underline{A} \,|$. Define

$$\theta : |\, P_H\underline{A} \,| \to \mathcal{P}_H \; |\, \underline{A} \,| \qquad\qquad \overline{\alpha} \longmapsto Cl_H(\{ \overline{a} \mid a \in \alpha \})$$

with $\alpha \in C$ and

$$\eta : \mathcal{P}_H \; |\, \underline{A} \,| \to |\, P_H\underline{A} \,| \qquad\qquad Cl_H(\{ \overline{a} \mid a \in X \}) \longmapsto \overline{X},$$

where $X \subseteq^{fin} A$. Suppose $\overline{\alpha} \subseteq \overline{\beta}$. Then $\alpha \in \overline{\beta}$, or $\{\beta\} \vdash \{\alpha\}$. But by definition we have $\forall a \in \alpha \exists b \in \beta.\, \{b\} \vdash_{\underline{A}} \{a\}$. Hence $\forall \overline{a} \in \{ \overline{a'} \mid a' \in \alpha \} \exists \overline{b} \in \{ \overline{b'} \mid b' \in \beta \}$ such that $\overline{a} \subseteq \overline{b}$. Therefore $\{ \overline{a'} \mid a \in \alpha \} \sqsubseteq_{\sim_1} \{ \overline{b'} \mid b' \in \beta \}$, which implies that θ is order preserving. It is easy to see that θ is one-one. That $\eta \circ \theta = \mathrm{id}_{|P_H\underline{A}|}$ and $\theta \circ \eta = \mathrm{id}_{\mathcal{P}_H |\underline{A}|}$ are also obvious. ∎

Definition 4.4.2 Let $\underline{A}$ be a strongly finite information system. The Smyth power-domain of $\underline{A}$ is the information system $P_S\underline{A} = (C, \vdash)$ where[2]

- $C = \mathrm{Fin}(A) \setminus \{\emptyset\}$
- $X \vdash Y$ iff $\exists \beta \in Y. \forall \alpha_0 \in C. (\forall \alpha \in X. \alpha_0 \cap \alpha \neq \emptyset) \Rightarrow \alpha_0 \vdash_{\underline{A}} \beta$

When $X = \{\alpha\}$, $Y = \{\beta\}$ where $\alpha, \beta \in C$, the entailment $\{\alpha\} \vdash \{\beta\}$ means $\forall W \subseteq^{fin} A.\, (W \cap \alpha \neq \emptyset \Longrightarrow \forall w \in W \exists b \in \beta.\, \{w\} \vdash_{\underline{A}} \{b\})$. In other words, $\{\alpha\} \vdash \{\beta\}$ iff $\forall a \in \alpha \exists b \in \beta.\, \{a\} \vdash \{b\}$. Therefore $\vdash_{P_S\underline{A}}$ on C is a reverse of the preorder $\sqsubseteq_{\sim_0}$ introduced in chapter 2 on the non-empty, finite sets of finite elements of $|\, \underline{A} \,|$.

Having agreed on how the entailment should be on singletons, we show that $X \vdash Y$ is equivalent to

$$\forall \alpha_0 \in C.\, (\forall \alpha \in X. \{\alpha_0\} \vdash \{\alpha\}) \Longrightarrow \exists \beta \in Y. \{\alpha_0\} \vdash \{\beta\},$$

[2]In an extended version of [Sc82] Scott call a finite set α_0 with the property $(\forall \alpha \in X. \alpha_0 \cap \alpha \neq \emptyset)$ a choice set.

which we write as $X \vdash' Y$, a description of the situation $\bigcap\{\,\overline{\alpha}\uparrow \mid \alpha \in X\,\} \subseteq \bigcup\{\,\overline{\beta}\uparrow \mid \beta \in Y\,\}$.

$X \vdash Y \Longrightarrow X \vdash' Y$: Suppose $\alpha_0 \in C$ and $\forall \alpha \in X.\, \{\,\alpha_0\,\} \vdash \{\,\alpha\,\}$. Let $\beta_0 \in Y$ be such that

$$(\,\forall \alpha \in X.\, \alpha' \cap \alpha \neq \emptyset\,) \Rightarrow \alpha' \vdash_{\underline{A}} \beta_0.$$

Let $a_0 \in \alpha_0$. For any $\alpha \in X$, there is $b_\alpha \in \alpha$ for which $\{\,a_0\,\} \vdash_{\underline{A}} \{\,b_\alpha\,\}$. Clearly $\{\,b_\alpha \mid \alpha \in X\,\} \cap \alpha \neq \emptyset$ for all $\alpha \in X$. Therefore, $\{\,b_\alpha \mid \alpha \in X\,\} \vdash_{\underline{A}} \beta_0$. However, $\{\,a_0\,\} \vdash^*_{\underline{A}} \{\,b_\alpha \mid \alpha \in X\,\}$. Thus $\{\,a_0\,\} \vdash_{\underline{A}} \beta_0$ which implies $\{\,a_0\,\} \vdash_{\underline{A}} \{\,b_0\,\}$ for some $b_0 \in \beta_0$ and this is true for any a_0 in α_0, which means $\{\,\alpha_0\,\} \vdash \{\,\beta_0\,\}$.

$X \vdash' Y \Longrightarrow X \vdash Y$: Suppose $X \vdash' Y$. Rewrite X as $\{\,\alpha_i \mid i \in I\,\}$, where I is finite. For any $f \in \{\,g : I \to \bigcup X \mid \forall i \in I.\, g(i) \in \alpha_i\,\}$, let $\{\,f(i) \mid i \in I\,\} \dashv\vdash_{\underline{A}} Z_f$. Clearly for any c in $\bigcup_f Z_f$, for any $i \in I$, $\{\,c\,\} \vdash_{\underline{A}} \alpha_i$. Therefore for any i, $\{\,\bigcup_f Z_f\,\} \vdash \{\,\alpha_i\,\}$, and hence $\{\,\bigcup_f Z_f\,\} \vdash \{\,\beta_0\,\}$ for some $\beta_0 \in Y$, since $X \vdash' Y$. Now let $\alpha_0 \in C$ be such that for any α in X, $\alpha_0 \cap \alpha \neq \emptyset$. For each i, select an $a_i \in \alpha_0 \cap \alpha_i$. The collection of such a_i's corresponds to a function h such that $h(i) = a_i$. If $\alpha_0 \vdash_{\underline{A}} \emptyset$ then there is nothing to prove; otherwise we have $\{\,h(i) \mid i \in I\,\} \dashv\vdash_{\underline{A}} Z_h$ and $E_h \neq \emptyset$. $\{\,\bigcup_f Z_f\,\} \vdash \{\,\beta_0\,\}$ implies $\{\,c\,\} \vdash_{\underline{A}} \beta_0$ for all $c \in Z_h$. Hence $\alpha_0 \vdash_{\underline{A}} \beta_0$.

From the above explanation it is easy to see that we could have used the following definition of entailment for the Smyth powerdomain,

$$X \vdash Y \text{ iff } \exists \beta \in Y. \forall \alpha_0 \subseteq \bigcup X.\, (\,\forall \alpha \in X.\, \mid \alpha_0 \cap \alpha \mid = 1\,) \Rightarrow \alpha_0 \vdash_{\underline{A}} \beta$$

which is better, since it avoids the use of universal quantifier over C. Here $\mid S \mid$ stands for the number of elements of S.

It is suitable to think of $\alpha \in C$ as $\Box \bigvee \alpha$, with the interpretation that a set of processes S satisfies $\Box \bigvee \alpha$ iff each process in S satisfies $\bigvee \alpha$. We have, under this interpretation, $\Box \bigvee X \Rightarrow \Box \bigvee Y$ iff $\bigvee X \Rightarrow \bigvee Y$, iff $\forall a \in X\ \exists b \in Y.\ a \Rightarrow b$ and

$$(\,\Box \bigvee X_1\,) \wedge (\,\Box \bigvee X_2\,) \wedge \cdots (\,\Box \bigvee X_n\,)$$
$$\Leftrightarrow \Box \left[\,(\,\bigvee X_1\,) \wedge (\,\bigvee X_2\,) \wedge \cdots (\,\bigvee X_n\,)\,\right].$$

Proposition 4.4.3 If $\underline{A}$ is a strongly finite information system then so is $P_S\underline{A}$.

Proof The only non-trivial part is finite closure. Let V be a finite set of propositions of $P_S\underline{A}$. $\bigcup V$ is a finite set of propositions of $\underline{A}$. Therefore there is a finite set P of $\underline{A}$,

$\dashv\vdash_{\underline{A}}$ −closed and contains $\bigcup V$. Then $\{\alpha \mid \alpha \subseteq P\}$ is a $\dashv\vdash$ −closed set of $P_S\underline{A}$. In fact, let $X \subseteq \{\alpha \mid \alpha \subseteq P\}$. Rewrite X as $\{\alpha_i \mid i \in I\}$, where I is finite. For any $f \in \{g : I \to \bigcup X \mid \forall i \in I.\, g(i) \in \alpha_i\}$, let $\{f(i) \mid i \in I\} \dashv\vdash_{\underline{A}} Z_f$. Clearly for any c in $\bigcup_f Z_f$, for any $i \in I$, $\{c\} \vdash_{\underline{A}} \alpha_i$. Therefore for any i, $\{\bigcup_f Z_f\} \vdash \{\alpha_i\}$. According to our definition of entailment for $P_S\underline{A}$, $X \dashv\vdash \{\bigcup_f Z_f\}$. ∎

From the proof we can see that, $P_S\underline{A}$, too, is definite. Similar to proposition 4.4.2 we have the following proposition, whose proof follows the same pattern, hence omitted.

Proposition 4.4.4 $|\, P_S\underline{A}\,|$ is isomorphic to $\mathcal{P}_S\ |\,\underline{A}\,|$.

Definition 4.4.3 Let $\underline{A}$ be an SFIS. The Plotkin powerdomain of $\underline{A}$ is the information system $P_P\underline{A} = (C, \vdash)$, with

- $C = \mathrm{Fin}(A) \setminus \{\emptyset\}$
- $X \vdash Y$ iff $\forall\beta.\, (\forall\alpha \in X.\, \{\beta\} \vdash \{\alpha\}) \Longrightarrow \exists\beta' \in Y.\, \{\beta\} \vdash \{\beta'\}$

  where $\{\alpha\} \vdash \{\beta\}$ iff $\{\alpha\} \vdash_{P_H\underline{A}} \{\beta\}$ & $\{\alpha\} \vdash_{P_S\underline{A}} \{\beta\}$

Like the construction of function space, we have used a universal quantifier in defining the entailment for the Plotkin powerdomain. This universal quantifier can be, however, avoided by a slightly complicated definition use the algorithm introduced in proposition 4.4.5, though we do not have time and space to go into details of it.

Proposition 4.4.5 If $\underline{A}$ is an SFIS then so is $P_P\underline{A}$.

Proof $P_P\underline{A}$ is clearly a GIS. Let $P_P\underline{A} = (C, \vdash)$.

$$\begin{aligned} V \subseteq^{fin} C &\Longrightarrow \bigcup V \subseteq^{fin} A \\ &\Longrightarrow \exists P \subseteq^{fin} A.\ \bigcup V \subseteq P \ \&\ P \text{ is } \dashv\vdash_{\underline{A}} \text{-closed} \\ &\Longrightarrow \{\alpha \mid \alpha \subseteq P\} \text{ is } \dashv\vdash_{P_P\underline{A}} \text{-closed.} \end{aligned}$$

We check the last step in the above implication. Let $X \subseteq \{\alpha \mid \alpha \subseteq P\}$. We have the following algorithm which finds a subset $X' \subseteq \{\alpha \mid \alpha \subseteq P\}$ such that $X \dashv\vdash_{P_P\underline{A}} X'$.

Step 1 Form a list ℓ with its head being

$$(\bigcup_f Z_f,\ \bigcup X),$$

where $\bigcup_f Z_f$ is the set constructed by the procedure described in the proof of Proposition 4.4.3.

Step 2 Add to ℓ the element

$$(\, S,\ S \cup (\bigcup X)\,)$$

for each $S \subseteq \bigcup_f Z_f$, and remove the head $(\, \bigcup_f Z_f,\ \bigcup X\,)$ from ℓ.

Step 3 Do this step for ℓ until each member of it is of the form $(\, S,\, T\,)$ where $S \supseteq T$: Pick up an element $(\, S,\, T\,)$ in ℓ for which $S \not\supseteq T$(After step 2 we must have $S \subseteq T$, obviously). Let a be in T but not in S. For each $s \in S$, add to ℓ the element

$$(\, S \cup S',\ \{\, s' \,\} \cup (T \setminus \{\, a \,\})\,)$$

for each $s' \in S'$ if S' is not empty, where $\{\, s,\, a \,\} \dashv\vdash_{\underline{A}} S'$ and $S' \subseteq P$. If S' is empty then skip. Remove the current element $(\, S,\, T\,)$ from ℓ.

Step 4 Replace each element $(\, S,\, T\,)$ of ℓ with $S \neq T$ by the elements $(\, T,\, T\,)$, and $(\, U_a,\, U_a\,)$ for each $a \in (\, S \setminus T\,)$ and $U_a \subseteq (\, S \setminus \{\, a \,\})$.

The algorithm must terminate. To show this it is enough to check that Step 3 terminates. But

$$|\ (s' \cup (T \setminus \{\, a \,\})) \setminus (S \cup S')\ | = |\ (T \setminus \{\, a \,\}) \setminus (S \cup S')\ | < |\ T \setminus S\ |$$

since $s' \in S'$ and $a \in T \setminus S$. The conclusion is then clear.

We claim that after the algorithm stops, we have

$$X \dashv\vdash_{P_P \underline{A}} \{\, U \mid (\, U,\, U\,) \text{ is a member of the list } \ell \},$$

where each U is clearly a subset of P. What the algorithm is doing at each step is clear; but it needs explanation why it fulfils our goal. Here we have the corresponding results to some of the lemmas given in Chapter 5 (The rest of the proof is best read together with the revalent materials in Chapter 5). Each member $(\, S,\, T\,)$ of the list ℓ corresponds to a logical formula $(\, \Box \bigvee S\,) \wedge \bigwedge \{\, \Diamond b \mid b \in T \,\}$, i.e. the first element S takes care of the order of the Smyth powerdomain while the second element T takes care of the order of the Hoare powerdomain. At each stage, the whole list corresponds to a disjunction of all its members, and we need only make sure that at each step the algorithm maintains

the equivalence of the big formula the list represents. More precisely, note those lemmas are still valid if we replace the 'prime assertions' by the propositions in A of $\underline{A}$. The equivalence should be rephased as $\ell \Longleftrightarrow \ell'$ iff for any $\alpha \in C$,

$$\exists (S, T) \in \ell. \{\alpha\} \vdash_{P_S\underline{A}} \{S\} \;\&\; \{\alpha\} \vdash_{P_H\underline{A}} \{T\}$$

$$\Longleftrightarrow \exists (S, T) \in \ell'. \{\alpha\} \vdash_{P_S\underline{A}} \{S\} \;\&\; \{\alpha\} \vdash_{P_H\underline{A}} \{T\}$$

where we borrowed the notation $(S, T) \in \ell$ to mean (S, T) is an element in the list.

The following sequence of observations, which are routine to check, finish the proof of Proposition 4.4.5.

Observation 1 $\forall \alpha \in X. \{\beta\} \vdash_{P_P\underline{A}} \{\alpha\}$ iff $\{\beta\} \vdash_{P_S\underline{A}} \{\bigcup_f Z_f\}$ and $\{\beta\} \vdash_{P_H\underline{A}} \{\bigcup X\}$, where $\bigcup_f Z_f$, using the same notation, is the set introduced in the proof of Proposition 4.4.3.

Observation 2 $\{\beta\} \vdash_{P_S\underline{A}} \{S\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{T\}$ iff $\exists S' \subseteq S$, $\{\beta\} \vdash_{P_S\underline{A}} \{S'\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{S' \cup T\}$.

Observation 3 Suppose $S \subseteq T$ and $a \in T \setminus S$. Then $\{\beta\} \vdash_{P_S\underline{A}} \{S\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{T\}$ iff $\exists s' \in S'$, $\{\beta\} \vdash_{P_S\underline{A}} \{S \cup S'\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{\{s'\} \cup (T \setminus \{a\})\}$. Here we reused all the notations used in Step 3 of the algorithm.

Observation 4 Suppose $S \supseteq T$ but $S \neq T$. Then $\{\beta\} \vdash_{P_S\underline{A}} \{S\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{T\}$ iff for some U $\{\beta\} \vdash_{P_S\underline{A}} \{U\} \;\&\; \{\beta\} \vdash_{P_H\underline{A}} \{U\}$, where $U = T$ or $U \subseteq (S \setminus \{a\})$ for some $a \in S \setminus T$. ∎

Similar to Proposition 4.4.2 we have

Proposition 4.4.6 $|\, P_P\underline{A} \,|$ is isomorphic to $\mathcal{P}_P \;|\, \underline{A} \,|$.

What we have given in this section are the various constructions on SFISs. The most tricky one is the Plotkin powerdomain, by which the phenomenon of non-consistent completeness is introduced. However, the function space construction seems equally tricky. The sum, product and lifting are simple, and so are the Hoare and the Smyth powerdomain. These two powerdomain constructions have the pleasing property that they 'knock down' any ω−algebraic cpo into a Scott domain.

4.5 CPO of SFISs

Following the idea of [LaWi84] we introduce a complete partial order of SFISs. We show that all the constructions induce continuous functions on the big cpo. Within this cpo it is possible to solve equations on SFISs using fixed point theory. The order on SFISs captures an intuitive notion that one information system can be viewed as a subsystem of another.

Definition 4.5.1 Let $\underline{A} = (A, \vdash_{\underline{A}})$, $\underline{B} = (B, \vdash_{\underline{B}})$ be SFISs. $\underline{A} \trianglelefteq \underline{B}$ if

- $A \subseteq B$
- $X \vdash_{\underline{A}} Y \iff X \cup Y \subseteq A \;\&\; X \vdash_{\underline{B}} Y$

When $\underline{A} \trianglelefteq \underline{B}$ we call $\underline{A}$ a subsystem of $\underline{B}$.

Proposition 4.5.1 Let $\underline{A}$ and $\underline{B}$ be SFISs. If $\underline{A} \trianglelefteq \underline{B}$ then there is an embedding-projection pair between $|\underline{A}|$ and $|\underline{B}|$.

Proof Define

$$\theta : |\underline{A}| \to |\underline{B}| \qquad x \longmapsto \{ b \mid \exists a \in x.\ \{ a \} \vdash_{\underline{B}} \{ b \} \}$$

and

$$\phi : |\underline{B}| \to |\underline{A}| \qquad y \longmapsto y \cap A.$$

We show that θ, ϕ form an embedding-projection pair, *i.e.*, $\phi \circ \theta = \mathrm{id}_{|\underline{A}|}$, $\theta \circ \phi \sqsubseteq \mathrm{id}_{|\underline{B}|}$ and θ, ϕ are continuous.

Let $x \in |\underline{A}|$. We check $\theta(x) \in |\underline{B}|$. Suppose $Z \subseteq \theta(x)$ and $Z \vdash_{\underline{B}} H$. By definition $\forall c \in Z \exists a \in x.\ \{ a \} \vdash_{\underline{B}} \{ c \}$. Let X be the collection of such a's. Clearly $X \subseteq x$. $X \vdash^{*} Z$, which implies $X \vdash_{\underline{B}} H$. By the finite closure axiom there is some $X' \subseteq A$ such that $X \dashv\vdash_{\underline{A}} X'$. As $X \subseteq x$ and $X \vdash_{\underline{A}} X'$, there is some $a' \in x \cap X'$. But $\underline{A}$ is a subsystem of $\underline{B}$. We have $X \dashv\vdash_{\underline{B}} X'$, and hence $\{ a' \} \vdash \{ a \}$ for each $a \in X$. Thus $\{ a' \} \vdash_{\underline{B}} H$ and therefore $\{ a' \} \vdash_{\underline{B}} \{ b \}$ for some $b \in H$. That is, $b \in \theta(x)$.

Let $y \in |\underline{B}|$. We check $y \cap A \in |\underline{A}|$. Suppose $Y \subseteq y \cap A$ and $Y \vdash_{\underline{A}} Z$. Then $\exists c \in Z.\ c \in y$. But $Z \subseteq A$, so $c \in y \cap A$.

The proof that θ, ϕ form an embedding-projection pair is then straightforward. ∎

As the collection of SFISs do not form a set but rather a class they cannot form a complete partial order(cpo) in the ordinary sense. We could say that they form a *large*

cpo $\mathbf{CPO}_{SFIS}$. However the standard theory of fixed points of continuous functions still works for $\mathbf{CPO}_{SFIS}$, and that is all we need.

Theorem 4.5.1 The relation $\trianglelefteq$ is a partial order with the least element $\underline{\perp} = (\emptyset, \emptyset)$. If $\underline{A_0} \trianglelefteq \underline{A_1} \trianglelefteq \cdots \trianglelefteq \underline{A_i} \trianglelefteq \cdots$ is an increasing chain of SFISs where $\underline{A_i} = (A_i, \vdash_i)$, then their least upper bound is

$$\bigcup_i \underline{A_i} = \left(\bigcup_i A_i, \bigcup_i \vdash_i \right).$$

Proof It is routine to check that

$$\bigcup_i \underline{A_i} = \left(\bigcup_i A_i, \bigcup_i \vdash_i \right)$$

is a SFIS . For each i, $\underline{A_i} \trianglelefteq \bigcup_k \underline{A_k}$ because of the following:

1. $A_i \subseteq \bigcup_k A_k$.

2. If $X \cup Y \subseteq A_i$ and $X \vdash_{\bigcup \underline{A_k}} Y$ then $X \vdash_j Y$ for some $j \geq i$ because $\vdash_{\bigcup \underline{A_k}} = \bigcup_k \vdash_k$. Therefore $X \vdash_i Y$.

It is also the least upper bound of the chain. Suppose $\underline{B}$ is an upper bound of the chain. Then for each i, $A_i \subseteq B$. Thus $\bigcup A_i \subseteq B$.

$$X \vdash_{\bigcup \underline{A_k}} Y \iff X \cup Y \subseteq \bigcup A_k \;\&\; \exists i.\, X \vdash_i Y$$

$$\iff X \cup Y \subseteq \bigcup A_k \;\&\; X \vdash_{\underline{B}} Y$$

Therefore, $\bigcup \underline{A_i} \trianglelefteq \underline{B}$. ∎

The subsystem relation $\trianglelefteq$ can be easily extended to $n-$tuples of information systems coordinatewisely. More precisely we require

$$(\underline{A_1}, \underline{A_2}, \cdots \underline{A_n}) \trianglelefteq (\underline{B_1}, \underline{B_2}, \cdots \underline{B_n})$$

iff for each $1 \leq i \leq n$, $\underline{A_i} \trianglelefteq \underline{B_i}$. For convenience write $\vec{\underline{A}}$ for $(\underline{A_1}, \underline{A_2}, \cdots \underline{A_n})$.

The least upper bound of an $\omega-$chain of $n-$tuples of information systems is then just the $n-$tuple of information systems consisting of the least upper bounds on each component, *i.e.* if

$$\vec{\underline{A}}_1 \trianglelefteq \vec{\underline{A}}_2 \cdots \trianglelefteq \vec{\underline{A}}_i \trianglelefteq \cdots$$

then

$$\pi_j \left(\bigsqcup_i \vec{\underline{A}}_i \right) = \bigcup_i \pi_j \left(\vec{\underline{A}}_i \right).$$

An operation F form $n-$tuples of information systems to $m-$tuples of information systems is said to be continuous iff it is monotonic, *i.e.* $\vec{\underline{A}} \trianglelefteq \vec{\underline{B}}$ implies $F(\vec{\underline{A}}) \trianglelefteq F(\vec{\underline{B}})$ and preserves $\omega-$increasing chains of information systems, *i.e.*

$$\vec{\underline{A}}_1 \trianglelefteq \vec{\underline{A}}_2 \cdots \trianglelefteq \vec{\underline{A}}_i \trianglelefteq \cdots$$

implies

$$\bigcup_i F(\vec{\underline{A}}_i) = F(\bigcup_i \vec{\underline{A}}_i).$$

It is well known that for functions on tuples of cpos they are continuous iff by changing (any) one argument while fixing others the induced function is continuous.

Larsen and Winskel [LaWi84] have a useful lemma which concludes that an operation F is continuous iff it is monotonic with respect to $\trianglelefteq$ and continuous on proposition sets, *i.e.*, , for any $\omega-$chain

$$\vec{\underline{A}}_1 \trianglelefteq \vec{\underline{A}}_2 \cdots \trianglelefteq \vec{\underline{A}}_i \trianglelefteq \cdots,$$

each proposition of $F(\bigcup_i \vec{\underline{A}}_i)$ is a proposition of $\bigcup_i F(\vec{\underline{A}}_i)$.

Theorem 4.5.2 $(\ )_\uparrow$, $+$, $\times$, $\rightarrow$, P_H, P_S, and P_P are all continuous.

Proof We illustrate the proof for $\rightarrow$. The proof for the other cases follow the same pattern, hence ommitted.

We have to show that $\rightarrow$ is a continuous operation from pairs of information systems to information systems. As Proposition 4.4.5 indicates, $\rightarrow$ is a well defined operation on SFISs.

$\rightarrow$ is monotonic in its first argument. Suppose $\underline{A} \trianglelefteq \underline{A'}$ and $\underline{B}$ are information systems. Write

$$\underline{C} = (C, \vdash) = \underline{A} \rightarrow \underline{B}$$

and

$$\underline{C'} = (C', \vdash') = \underline{A'} \rightarrow \underline{B}.$$

We check 1 and 2 in Definition 4.5.1 to show that $\underline{C} \trianglelefteq \underline{C'}$.

1. Suppose $X \in C$. It is easy to see that $X \in C'$ by Definition 4.4.3.

2. Clearly $X \vdash_{\underline{C}} Y$ implies $X \vdash_{\underline{C'}} Y$. Assume $X \subseteq C$, $Y \subseteq C$ and $X \vdash_{\underline{C'}} Y$. Because in this case each entailment subscribed by $\underline{A'}$ about the first components of elements of X is an entailment subscribed by $\underline{A}$, using the assumption that $\underline{A} \trianglelefteq \underline{A'}$. we have $X \vdash_{\underline{C}} Y$.

Now we show that $\rightarrow$ is continuous on proposition sets. Let

$$\underline{A_0} \trianglelefteq \underline{A_1} \trianglelefteq \cdots \trianglelefteq \underline{A_i} \trianglelefteq \cdots$$

be a chain of SFISs. Let X be a proposition of $(\bigcup_i \underline{A_i}) \rightarrow \underline{B}$. Then $\pi_1 X \subseteq^{\text{fin}} \bigcup_i A_i$. Hence $\pi_1 X \subseteq A_j$ for some j, which means X is a proposition of $\underline{A_j} \rightarrow \underline{B}$. Thus X is a proposition of $\bigcup_i(\underline{A_i} \rightarrow \underline{B})$. By the previous mentioned lemma of Larsen and Winskel, we deduce that $\rightarrow$ is continuous in its first argument. Similarly we can prove that $\rightarrow$ is continuous in its second argument, and hence it is continuous. ∎

Chapter 5

A Logic of SFP

This chapter introduces a logic of **SFP**. A meta-language for denotational semantics is introduced with general type constructions like sum, product, function space, the three powerdomains, and recursively defined types. For each type there is a language of open set assertions. Proof systems are given; they use inequational formulae to axiomatise entailment and non-entailment of assertions. Soundness and completeness results are obtained. As an application of the logic, the style of assertions of Brookes proof system is shown to be determined by the logic of Plotkin's domain of resumptions.

This chapter has the following sections. Section 5.1 introduces a meta-language for denotational semantics with general type constructions like sum, product, function space, the three powerdomains, and recursively defined types. For each type there is an assertion language, constructed from the language of the type constituents. Section 5.2 proposes proof systems for the typed assertion languages. The proof rules axiomatise the entailment relation $\leq$ between assertions as well as the non-entailment $\not\leq$ relation. They are built up form those of the components of the type. Section 5.3 interprets assertions as open sets of domains and proves the soundness of the proof systems. Section 5.4 gives completeness and expressiveness theorems. The last section presents an application of the framework, showing that the style of assertions of Brookes[Br85] is determined by the logic of Plotkin's domain of resumptions[Pl76].

5.1 Typed Assertion Languages

Our framework consists of four parts: a meta-language for denotational semantics, typed assertion languages, structural and logical rules and meta-predicates. This section is concerned with the meta-language for denotational semantics and typed assertion languages.

The meta-language for denotational semantics is usually introduced as a language of

type expressions as follows:

$$\sigma ::= \mathbf{1} \mid \sigma \times \tau \mid \sigma \to \tau \mid \sigma + \tau \mid \sigma_\perp \mid \mathcal{P}_S[\sigma] \mid \mathcal{P}_H[\sigma] \mid \mathcal{P}_P[\sigma] \mid t \mid rec\, t.\sigma$$

where t is a type variable and σ, τ ranges over type expressions.

Note we could have avoided using **1** as a ground type because it can be expressed by the recursively defined type $rec\ t.t$. Every closed type expression is interpreted as an **SFP** object, with **1** as the one-point domain, and $\times$ as the Cartesian product, $+$ as the coalesced sum, $(\ )\perp$ as lifting, $\to$ as the function space, $\mathcal{P}_S$, $\mathcal{P}_H$ and $\mathcal{P}_P$ as the Smyth, Hoare and Plotkin powerdomains, and $rec\ t.\sigma$ the initial solution of the associated domain equation. Write $\mathcal{D}(\sigma)$ for the domain corresponding to σ.

Using this meta-language we can give denotational semantics to a large class of programming languages. For a programming language L, we specify a type expression σ and denote a program as an element in $\mathcal{D}(\sigma)$. Here we are not concerned with the problem how the type expression is selected for a particular programming language.

For each type σ we introduce an assertion language $\mathcal{A}_\sigma$. The assertions of $\mathcal{A}_\sigma$ are constructed according to the way σ is built up from its subtypes. We use the notations for type constructions again in the assertion language to make a clear correspondence between types and typed assertions, just as we did for type constructions and constructions on domains.

Assertion Formation Rules

$$(\mathbf{t},\mathbf{f}) \qquad \mathbf{t},\mathbf{f} : \sigma$$

$$(\wedge - \vee) \qquad \frac{\varphi, \psi : \sigma}{\varphi \wedge \psi : \sigma \qquad \varphi \vee \psi : \sigma}$$

$$(\times) \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \times \psi : \sigma \times \tau}$$

$$(\rightarrow) \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \rightarrow \psi : \sigma \rightarrow \tau}$$

$$(+) \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\dot{inl}\,\varphi : \sigma + \tau \qquad \dot{inr}\,\psi : \sigma + \tau}$$

$$(\perp) \qquad \frac{\varphi : \sigma}{\varphi_\perp : \sigma_\perp}$$

$$(H) \qquad \frac{\varphi : \sigma}{\Diamond\varphi : \mathcal{P}_H[\sigma]}$$

$$(S) \qquad \frac{\varphi : \sigma}{\Box\varphi : \mathcal{P}_S[\sigma]}$$

$$(P) \qquad \frac{\varphi : \sigma}{\Box\varphi : \mathcal{P}_P[\sigma],\ \Diamond\varphi : \mathcal{P}_P[\sigma]}$$

$$(rec) \qquad \frac{\varphi : \sigma[rec\ t.\sigma/t]}{\varphi : rec\ t.\sigma}$$

Rule (**t**,**f**) says that **t** and **f** are assertions of every type, standing for the logical 'true' and 'false'. Rule $(\wedge - \vee)$ means that for assertions of the same type, we can form their logical conjunction and disjunction. The rest of the rules tell us how assertions are built up according to the structure of the type. For example, according to $(\times)$, if A is an assertion of type σ and B is an assertion of type τ, then $A \times B$ is an assertion of type $\sigma \times \tau$. We do not need extra structures for recursively defined types. This is because, as shown in Chapter 4, that it is possible to get equality in solving recursive domain equations.

In this way, given any type expression σ, there is a corresponding assertion language $\mathcal{A}_\sigma$.

5.2 Proof Systems

We now propose a proof system for each $\mathcal{A}_\sigma$. For the axiomatization of the logic we introduce $\leq_\sigma$ between assertions of $\mathcal{A}_\sigma$. The relation $\leq_\sigma$ stands for logical entailment. It is intended that if $\varphi \leq_\sigma \psi$ then every object (program) satisfying φ also satisfies ψ. This intuition will be made precise later when we give interpretations to assertions. The logical equivalence, $=_\sigma$, stands for both $\leq_\sigma$ and $\geq_\sigma$. Subscripts are usually omitted. When $\varphi, \psi \in \mathcal{A}_\sigma$, $\varphi \leq_\sigma \psi$ and $\varphi =_\sigma \psi$ are called *positive formulae.*

We also introduce $\not\leq_\sigma$ between assertions of $\mathcal{A}_\sigma$. $\varphi \not\leq \psi$ reads 'φ does not entail ψ.' Apart from some technical reasons (the definition of prime assertions for function space), our motivation to axiomatise $\not\leq$ is based on three observations. First, descriptions of domains usually involve a consistency predicate *Con*. In the context of domain logic, what $Con(\varphi)$ means is just $\varphi \not\leq \mathbf{f}$. So particular forms of $\not\leq$ have already been used. Secondly, when implementing domain logic by machines, one might try to input $\varphi \leq \psi$ and wait for an answer. Of course we would like the machine not only have the ability to say 'yes' when $\varphi \leq \psi$ is valid, but also 'no' when $\varphi \leq \psi$ is not. The reasoning involved in producing 'no' can be formulated by an axiomatisation of $\not\leq$, for the same sake as that 'yes' is formulated by an axiomatisation of $\leq$. Thirdly, theoretical results on effectively given domains show that $\varphi \leq \psi$ is decidable, which implies the feasibility of producing such a kind of negative information.

We call $\varphi \not\leq_\sigma \psi$ a *negative formula.* Both positive and negative formulae are *formulae.*

The axioms of the proof system are of the form $\varphi \leq_\sigma \psi$ or $\varphi \not\leq_\sigma \psi$ and rules

$$\frac{A_1 \quad A_2 \quad \cdots A_n}{B}$$

with A_i's and B formulae. Such a rule means, intuitively, that if A_i's are valid formulae then so is B.

By convention let $\bigvee \emptyset \equiv \mathbf{f}$ and $\bigwedge \emptyset \equiv \mathbf{t}$.

For each $\mathcal{A}_\sigma$ we have the following logical axioms and rules (with subscripts omitted):

Logical Axioms and Rules

$(\mathbf{t})$ $\quad \varphi \leq \mathbf{t}$ $\qquad$ $(\mathbf{f})$ $\quad \mathbf{f} \leq \varphi$

(Ref) $\quad \varphi \leq \varphi$

(Trans) $\quad \dfrac{\varphi \leq \varphi' \quad \varphi' \leq \varphi''}{\varphi \leq \varphi''}$

$(\leq - =)$ $\quad \dfrac{\varphi \leq \psi \quad \psi \leq \varphi}{\varphi = \psi}$

$(= - \leq)$ $\quad \dfrac{\varphi = \varphi'}{\varphi \leq \varphi'} \qquad \dfrac{\varphi = \varphi'}{\varphi' \leq \varphi}$

$(\wedge - \leq)$ $\quad \varphi \wedge \varphi' \leq \varphi \qquad \varphi \wedge \varphi' \leq \varphi'$

$(\leq - \wedge)$ $\quad \dfrac{\varphi \leq \varphi' \quad \varphi \leq \varphi''}{\varphi \leq \varphi' \wedge \varphi''}$

$(\vee - \leq)$ $\quad \dfrac{\varphi \leq \varphi' \quad \psi \leq \varphi'}{\varphi \vee \psi \leq \varphi'}$

$(\leq - \vee)$ $\quad \varphi \leq \varphi' \vee \varphi \qquad \varphi' \leq \varphi' \vee \varphi$

$(\wedge - \vee)$ $\quad \varphi \wedge (\varphi_1 \vee \varphi_2) \leq (\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2)$

$(\mathbf{t}, \mathbf{f})$ $\quad \mathbf{t} \not\leq \mathbf{f}$

$(\neg - \wedge)$ $\quad \dfrac{\varphi \not\leq \psi}{\varphi \not\leq \psi \wedge \psi'}$

$(\vee - \neg)$ $\quad \dfrac{\varphi \not\leq \psi}{\varphi \vee \varphi' \not\leq \psi}$

$(\wedge - \neg)$ $\quad \dfrac{\varphi \wedge \psi \not\leq \psi'}{\varphi \not\leq \psi'} \qquad \dfrac{\varphi \wedge \psi \not\leq \psi'}{\psi \not\leq \psi'}$

$(\neg - \vee)$ $\quad \dfrac{\varphi \not\leq \psi_1 \vee \psi_2}{\varphi \not\leq \psi_1} \qquad \dfrac{\varphi \not\leq \psi_1 \vee \psi_2}{\varphi \not\leq \psi_2}$

$(= - \neg)$ $\quad \dfrac{\varphi = \varphi' \quad \varphi' \not\leq \psi' \quad \psi' = \psi}{\varphi \not\leq \psi}$

There are type-specific rules which provide relationships between axioms of different types. There are also axioms that tell us how logical constructions interact with type constructions.

For product we have

Product

$(\times - \mathrm{t})$ $\quad \mathrm{t}_\sigma \times \mathrm{t}_\tau =_{\sigma\times\tau} \mathrm{t}_{\sigma\times\tau}$

$(\times - \mathrm{f})$ $\quad \mathrm{f}_\sigma \times \varphi = \mathrm{f}_{\sigma\times\tau} \quad \varphi \times \mathrm{f}_\tau =_{\sigma\times\tau} \mathrm{f}_{\sigma\times\tau}$

$(\times - \leq)$ $\quad \dfrac{\psi \leq_\sigma \psi' \quad \varphi \leq_\tau \varphi'}{\psi \times \varphi \leq_{\sigma\times\tau} \psi' \times \varphi'}$

$(\times - \vee)$ $\quad (\psi_1 \vee \psi_2) \times (\varphi_1 \vee \varphi_2) =_{\sigma\times\tau}$
$(\psi_1 \times \varphi_1) \vee (\psi_1 \times \varphi_2) \vee (\psi_2 \times \varphi_1) \vee (\psi_2 \times \varphi_2)$

$(\times - \wedge)$ $\quad (\varphi_1 \times \varphi_2) \wedge (\psi_1 \times \psi_2) =_{\sigma\times\tau} (\varphi_1 \wedge \psi_1) \times (\varphi_2 \wedge \psi_2)$

$(\neg - \times)$ $\quad \dfrac{\varphi \not\leq \varphi'}{\varphi \times \psi \not\leq \varphi' \times \psi'} \qquad \dfrac{\psi \not\leq \psi'}{\varphi \times \psi \not\leq \varphi' \times \psi'}$

These axioms and rules are self-evident. Similarly, there are axioms and rules for other type constructions. For simplicity subscripts are omitted.

Sum

$(+ - \mathrm{t})$ $\quad \mathit{inl}\,\mathrm{t} = \mathrm{t} \quad \mathit{inr}\,\mathrm{t} = \mathrm{t}$

$(+ - \mathrm{f})$ $\quad \mathit{inl}\,\mathrm{f} = \mathrm{f} \quad \mathit{inr}\,\mathrm{f} = \mathrm{f}$

$(\mathit{inl} - \leq)$ $\quad \dfrac{\varphi \leq \psi}{\mathit{inl}\,\varphi \leq \mathit{inl}\,\psi}$

$(\mathit{inr} - \leq)$ $\quad \dfrac{\varphi \leq \psi}{\mathit{inr}\,\varphi \leq \mathit{inr}\,\psi}$

$(\mathit{inl} - \wedge)$ $\quad \mathit{inl}\,(\varphi \wedge \psi) = (\mathit{inl}\,\varphi) \wedge (\mathit{inl}\,\psi)$

$(\mathit{inr} - \wedge)$ $\quad \mathit{inr}\,(\varphi \wedge \psi) = (\mathit{inr}\,\varphi) \wedge (\mathit{inr}\,\psi)$

$(\mathit{inl} - \vee)$ $\quad \mathit{inl}\,(\varphi \vee \psi) = (\mathit{inl}\,\varphi) \vee (\mathit{inl}\,\psi)$

$(\mathit{inr} - \vee)$ $\quad \mathit{inr}\,(\varphi \vee \psi) = (\mathit{inr}\,\varphi) \vee (\mathit{inr}\,\psi)$

$(\neg + \mathrm{f})$ $\quad \dfrac{\mathrm{t} \not\leq \varphi \qquad \mathrm{t} \not\leq \psi}{\mathit{inl}\varphi \wedge \mathit{inr}\psi \leq \mathrm{f}}$

$(\neg - +)$ $\quad \dfrac{\varphi \not\leq \psi}{\mathit{inl}\,\varphi \not\leq \mathit{inl}\psi} \qquad \dfrac{\varphi \not\leq \psi}{\mathit{inr}\,\varphi \not\leq \mathit{inr}\psi}$

We can combine axioms and rules for sum and lifting to get those for the separated sum.

Lifting

$(\bot - \mathbf{f})$ $\quad (\mathbf{f}_\sigma)_\bot = \mathbf{f}_{\sigma_\bot}$

$(\bot - \leq)$ $\quad \dfrac{\varphi \leq \psi}{\varphi_\bot \leq \psi_\bot}$

$(\bot - \wedge)$ $\quad (\varphi_1 \wedge \varphi_2)_\bot = (\varphi_1)_\bot \wedge (\varphi_2)_\bot$

$(\bot - \vee)$ $\quad (\varphi_1 \vee \varphi_2)_\bot = (\varphi_1)_\bot \vee (\varphi_2)_\bot$

$(\bot - \mathbf{t})$ $\quad \mathbf{t} \not\leq \varphi_\bot$

$(\neg - \bot)$ $\quad \dfrac{\varphi \not\leq \psi}{\varphi_\bot \not\leq \psi_\bot}$

Function Space

$(\rightarrow - \mathbf{t})$ $\quad \varphi \rightarrow \mathbf{t} = \mathbf{t} \qquad \mathbf{f} \rightarrow \varphi = \mathbf{t}$

$(\rightarrow - \leq)$ $\quad \dfrac{\psi' \leq \psi \quad \varphi \leq \varphi'}{\psi \rightarrow \varphi \leq \psi' \rightarrow \varphi'}$

$(\rightarrow - \wedge)$ $\quad \varphi \rightarrow (\psi_1 \wedge \psi_2) = (\varphi \rightarrow \psi_1) \wedge (\varphi \rightarrow \psi_2)$

$(\vee - \rightarrow)$ $\quad (\varphi_1 \vee \varphi_2) \rightarrow \psi = (\varphi_1 \rightarrow \psi) \wedge (\varphi_2 \rightarrow \psi)$

There is a rule for negative formulae of the function space which will be introduced later.

The Modality $\Box$

$(\Box - \mathbf{t})$ $\quad \Box\mathbf{f} = \mathbf{f}$

$(\Box - \mathbf{f})$ $\quad \Box\mathbf{t} = \mathbf{t}$

$(\Box - \leq)$ $\quad \dfrac{\varphi \leq \psi}{\Box\varphi \leq \Box\psi}$

$(\Box - \wedge)$ $\quad \Box(\varphi_1 \wedge \varphi_2) = (\Box\varphi_1) \wedge (\Box\varphi_2)$

$(\neg - \Box)$ $\quad \dfrac{\varphi \not\leq \psi}{\Box\varphi \not\leq \Box\psi}$

Note the axiom $(\Box - \mathbf{f})$ corresponding to the choice that the empty set is excluded

from the Smyth powerdomain.

The Modality $\Diamond$

$$(\Diamond - \mathbf{t}) \qquad \Diamond \mathbf{f} = \mathbf{f}$$

$$(\Diamond - \mathbf{f}) \qquad \Diamond \mathbf{t} = \mathbf{t}$$

$$(\Diamond - \leq) \qquad \frac{\varphi \leq \psi}{\Diamond\varphi \leq \Diamond\psi}$$

$$(\Diamond - \vee) \qquad \Diamond(\varphi_1 \vee \varphi_2) = (\Diamond\varphi_1) \vee (\Diamond\varphi_2)$$

$$(\neg - \Diamond) \qquad \frac{\varphi \not\leq \psi}{\Diamond\varphi \not\leq \Diamond\psi}$$

We remark that the axioms and rules just presented for $\Box$ and $\Diamond$ are for the Smyth powerdomain and the Hoare powerdomain, respectively. They are axioms for the Plotkin powerdomain, too. However, the following two axioms are for the Plotkin powerdomain only.

$\Diamond - \Box$:

$$(\Diamond) \qquad \Diamond\varphi \wedge \Box\psi \leq \Diamond(\varphi \wedge \psi)$$

$$(\Box) \qquad \Box(\varphi \vee \psi) \leq \Diamond\varphi \vee \Box\psi$$

$$(\neg\Diamond\Box) \qquad \frac{\bigwedge_{i\in I} \Diamond\varphi_i \not\leq \bigwedge_{j\in J} \Diamond\psi_j}{(\Box \bigvee_{i\in I} \varphi_i) \wedge \bigwedge_{i\in I} \Diamond\varphi_i \not\leq (\Box \bigvee_{j\in J} \psi_j) \wedge \bigwedge_{j\in J} \Diamond\psi_j}$$

$$\frac{\Box \bigvee_{i\in I} \varphi_i \not\leq \Box \bigvee_{j\in J} \psi_j}{(\Box \bigvee_{i\in I} \varphi_i) \wedge \bigwedge_{i\in I} \Diamond\varphi_i \not\leq (\Box \bigvee_{j\in J} \psi_j) \wedge \bigwedge_{j\in J} \Diamond\psi_j}$$

I believe all the axioms and rules are independent of each other. In other words, I suspect no axiom or rule is derivable form the rest. However, we can derive $(\times - \leq)$, $(\dot{in}l - \leq)$, $(\dot{in}r - \leq)$, $(\to - \leq)$, $(\bot - \leq)$, $(\Box - \leq)$ and $(\Diamond - \leq)$ from the other axioms and rules (including logical rules) if we introduce

$$(Contx) \qquad \frac{\varphi =_\sigma \psi}{C[\varphi] =_\tau C[\psi]}$$

where $C[\ ]$ is any context with assertions, logical and type constructions such as $[\ ]$, $\varphi \times [\ ] \vee u \times v$, and $([\ ] \to \varphi) \wedge ([\ ] \to \psi)$. $(Contx)$ means substitutions of equivalent assertions in any context preserve equivalence.

$(\rightarrow - \leq)$ can be derived in the following way.

$$(\varphi \leq \varphi' \ \& \ \psi' \leq \psi) \Longrightarrow (\varphi = \varphi \wedge \varphi' \ \& \ \psi = \psi \vee \psi').$$

Hence

$$\begin{aligned}
\psi \rightarrow \varphi &= \psi \rightarrow \varphi \wedge \varphi' && (Contx) \\
&= (\psi \rightarrow \varphi) \wedge (\psi \rightarrow \varphi') && (\rightarrow - \wedge) \\
&\leq \psi \rightarrow \varphi' && \text{logical} \\
&= \psi \vee \psi' \rightarrow \varphi' && (Contx) \\
&= (\psi' \rightarrow \varphi') \wedge (\psi \rightarrow \varphi') && (\vee - \rightarrow) \\
&\leq \psi' \rightarrow \varphi' && \text{logical}
\end{aligned}$$

where $A \leq B = C \leq D \cdots$ is an abbreviation for $A \leq B \ \& \ B = C \ \& C \leq D \cdots$.

Remark. In an assertion, type constructions $\times$, $\rightarrow$, inl, inr, $(\quad)_\perp$, $\Diamond$, and $\Box$ are given priority over logical operations. For example, when write $\mathit{inl}\,\varphi \vee \mathit{inr}\,\psi$ we mean $(\mathit{inl}\,\varphi) \vee (\mathit{inr}\,\psi)$.

5.3 Soundness

In this section we give an interpretation for assertions and prove that the proof systems are sound with respect to this interpretation.

For each closed type expression σ we define an *interpretation function*

$$[\![\ ]\!]_\sigma : \mathcal{A}_\sigma \rightarrow \mathbf{K\Omega}(\mathcal{D}(\sigma))$$

in the following structured way.

For each closed type expression σ, we define

$$[\![\mathbf{t}]\!]_\sigma = \mathcal{D}(\sigma)$$

$$[\![\mathbf{f}]\!]_\sigma = \emptyset$$

$$[\![\varphi \vee \psi]\!]_\sigma = [\![\varphi]\!]_\sigma \cup [\![\psi]\!]_\sigma$$

$$[\![\varphi \wedge \psi]\!]_\sigma = [\![\varphi]\!]_\sigma \cap [\![\psi]\!]_\sigma$$

With respect to type constructions we define

$$[\![\varphi \times \psi]\!]_{\sigma\times\tau} = \{\langle u, v\rangle \mid u \in [\![\varphi]\!]_\sigma \;\&\; v \in [\![\psi]\!]_\tau\}$$

$$[\![\dot{inl}\,\varphi]\!]_{\sigma+\tau} = \{\langle 0, u\rangle \mid u \in [\![\varphi]\!]_\sigma \setminus \{\perp_{\mathcal{D}(\sigma)}\}\}\cup$$

$$\{x \in \mathcal{D}(\sigma+\tau) \mid \perp_{\mathcal{D}(\sigma)} \in [\![\varphi]\!]_\sigma\}$$

$$[\![\dot{inr}\,\varphi]\!]_{\sigma+\tau} = \{\langle 1, u\rangle \mid u \in [\![\varphi]\!]_\tau \setminus \{\perp_{\mathcal{D}(\tau)}\}\}\cup$$

$$\{x \in \mathcal{D}(\sigma+\tau) \mid \perp_{\mathcal{D}(\tau)} \in [\![\varphi]\!]_\tau\}$$

$$[\![\varphi \to \psi]\!]_{\sigma\to\tau} = \{f \in [\mathcal{D}(\sigma) \to \mathcal{D}(\tau)] \mid [\![\varphi]\!]_\sigma \subseteq f^{-1}([\![\psi]\!]_\tau)\}$$

$$[\![(\varphi)_\perp]\!]_{(\sigma)_\perp} = \{\langle 0, u\rangle \mid u \in [\![\varphi]\!]_\sigma\}$$

$$[\![\Box\varphi]\!]_{\mathcal{P}_S(\sigma)} = \{U \in \mathcal{P}_S(\mathcal{D}\sigma) \mid U \subseteq [\![\varphi]\!]_\sigma\}$$

$$[\![\Box\varphi]\!]_{\mathcal{P}_P(\sigma)} = \{U \in \mathcal{P}_P(\mathcal{D}\sigma) \mid U \subseteq [\![\varphi]\!]_\sigma\}$$

$$[\![\Diamond\varphi]\!]_{\mathcal{P}_H(\sigma)} = \{L \in \mathcal{P}_H(\mathcal{D}\sigma) \mid L \cap [\![\varphi]\!]_\sigma \neq \emptyset\}$$

$$[\![\Diamond\varphi]\!]_{\mathcal{P}_P(\sigma)} = \{L \in \mathcal{P}_P(\mathcal{D}\sigma) \mid L \cap [\![\varphi]\!]_\sigma \neq \emptyset\}$$

$$[\![\varphi]\!]_{rect.\sigma} = \{\epsilon_\sigma(u) \mid u \in [\![\varphi]\!]_{\sigma[(rect.\sigma)\setminus t]}\}$$

where $\epsilon_\sigma : [\,\mathcal{D}(\,\sigma[\,(rect.\,\sigma)\setminus t\,]\,) \to \mathcal{D}(\,rect.\,\sigma\,)\,]$ is the isomorphism arising form the initial solution to the domain equation associated with type $rec\ t.\ \sigma$.

Definition 5.3.1 $\forall\varphi, \psi \in \mathcal{A}_\sigma$, write $\models_\sigma \varphi \leq_\sigma \psi$ if $[\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma$, for negative formulae, $\models_\sigma \varphi \not\leq_\sigma \psi$ if $[\![\varphi]\!]_\sigma \not\subseteq [\![\psi]\!]_\sigma$. Call a logical formula $\varphi \leq_\sigma \psi$ *valid* if $\models_\sigma \varphi \leq_\sigma \psi$; $\varphi \not\leq_\sigma \psi$ *valid* if $\models_\sigma \varphi \not\leq_\sigma \psi$.

Definition 5.3.2 $\forall\varphi, \psi \in \mathcal{A}_\sigma$, write $\vdash_\sigma \varphi \leq_\sigma \psi$ if $\varphi \leq_\sigma \psi$ is an axiom or it can be derived from axioms and rules given in Section 5.2 (together with those presented in Section 5.4 later) and similarly $\vdash_\sigma \varphi \not\leq_\sigma \psi$ if formula $\varphi \not\leq \psi$ can be derived from the proof system.

Definition 5.3.3 The proof system is called *sound* if $\vdash \varphi \leq_\sigma \psi$ implies $\models \varphi \leq_\sigma \psi$ and $\vdash \varphi \not\leq_\sigma \psi$ implies $\models \varphi \not\leq_\sigma \psi$. It is complete if $\models \varphi \leq_\sigma \psi$ implies $\vdash \varphi \leq_\sigma \psi$ and $\models \varphi \not\leq_\sigma \psi$ implies $\vdash \varphi \not\leq_\sigma \psi$. An axiom is *valid* if it is a valid formula. A rule is sound if it produces valid formulae from valid formulae.

It is clear that a proof system is sound iff all its axioms are valid and rules sound. Proposition 5.3.1 establishes the soundness of the proof system.

Proposition 5.3.1

- The logical axioms are valid and logical rules sound.
- The axioms for product, sum, function space, lifting are valid.
- The rules for product, sum, function space, lifting are sound.
- The axioms associated with $\Box$, $\Diamond$ are valid and rules sound.

Proof The proof is routine. We show the validness of the positive part of $\Box - \Diamond$ axioms. Let A, B be compact open sets of D, an **SFP** object. Clearly

$$T \in \{U \in \mathcal{P}_P(D) \mid U \cap A \neq \emptyset\} \cap \{V \in \mathcal{P}_P(D) \mid V \subseteq B\}$$
$$\Longrightarrow T \in \{U \in \mathcal{P}_P(D) \mid U \cap A \cap B \neq \emptyset\}.$$

Hence

$$\{U \in \mathcal{P}_P(D) \mid U \cap A \neq \emptyset\} \cap \{V \in \mathcal{P}_P(D) \mid V \subseteq B\}$$
$$\subseteq \{U \in \mathcal{P}_P(D) \mid U \cap A \cap B \neq \emptyset\}$$

So $(\Diamond)$ is valid. $(\Box)$ is valid because

$$T \in \{U \in \mathcal{P}_P(D) \mid U \subseteq A \cup B\}$$
$$\Longrightarrow T \in \{U \in \mathcal{P}_P(D) \mid U \subseteq B\} \cup \{U \in \mathcal{P}_P(D) \mid U \cap A \neq \emptyset\}.$$

The rest of the proof can be similarly carried out by inspecting each axiom and rule. ∎

Remark. In many proofs of this section and next section we will not check the case for recursively defined types, because all the time we are dealing with finite sets of assertions, and these assertions can always be considered as of some finite type.

5.4 Completeness

The proof system given in Section 5.2 is not complete since there are valid formulae which cannot be derived, as shown by the following example. Motivated from this example, we are going to equip the proof system with a meta-predicate and a meta-function so as to make it complete.

Example 5.4.1 Consider the type $\mathbf{1} \to [(\mathbf{1}_\perp) + (\mathbf{1}_\perp)]$ and the formula

$$\mathbf{t} \to (\dot{inr}(\mathbf{t}_\perp) \vee \dot{inl}(\mathbf{t}_\perp)) = [\mathbf{t} \to \dot{inr}(\mathbf{t}_\perp)] \vee [\mathbf{t} \to \dot{inl}(\mathbf{t}_\perp)]$$

of this type. This valid formula cannot be derived from the proof system either, because it is not an axiom and there is no rule about assertions of the form $\varphi \to (\alpha \vee \beta)$.

Although $\varphi \to \bigvee_{i\in I} \alpha_i = \bigvee_{i\in I} \varphi \to \alpha_i$ is not a valid axiom, it becomes valid when we restrict φ to some particular form, those assertions which are *prime*, corresponding to (non-empty) prime open sets. By introducing a predicate **P** on assertions and some axiom with **P** as side condition, we will be able to derive the valid formula in Example 5.4.1.

Definition 5.4.1 $\{\varphi_i \mid i \in I\}$, a finite set of assertions of type σ, is said to be *quasi*$-\wedge$ closed if for all non-null $J \subseteq I$ there is $K \subseteq I$ such that

$$\vdash \bigwedge_{j\in J} \varphi_j = \bigvee_{k\in K} \varphi_k.$$

From Theorem 2.5.1 we understand that quasi$-\wedge$ closed sets of assertions will be important in specifying the prime assertions of function space. For this reason we introduce a (recursive) function $\Pi^\star$ on finite sets of assertions. It is intended that if $\Pi^\star(A) = B$ then B is quasi$-\wedge$ closed and $A \subseteq B$. Of course $\Pi^\star$ just produces *one* such set for each A and it need not be a *minimal* quasi$-\wedge$ closed one.

Prime Assertions

(t) $\mathbf{P}(\mathbf{t})$

$(\times)$ $\dfrac{\mathbf{P}(\varphi) \quad \mathbf{P}(\psi)}{\mathbf{P}(\varphi \times \psi)}$

$(\to)$ $\dfrac{\Pi^*(\{\varphi_i \mid i \in I\}) = \{\varphi_i \mid i \in I\} \,\&\, \forall i, j \in I.(\mathbf{P}(\varphi_i) \,\&\, \mathbf{P}(\psi_i) \,\&\, [\varphi_i \not\leq \varphi_j \text{ or } \psi_i \leq \psi_j])}{\mathbf{P}(\bigwedge_{i \in I} \varphi_i \to \psi_i)}$

$(+)$ $\dfrac{\mathbf{P}(\varphi)}{\mathbf{P}(\mathit{inl}\,\varphi)} \qquad \dfrac{\mathbf{P}(\psi)}{\mathbf{P}(\mathit{inr}\,\psi)}$

$(\bot)$ $\dfrac{\mathbf{P}(\varphi)}{\mathbf{P}(\varphi_\bot)}$

(H) $\dfrac{\mathbf{P}(\varphi_i) \quad 1 \leq i \leq n}{\mathbf{P}(\bigwedge_{1 \leq i \leq n} \Diamond \varphi_i)}$

(S) $\dfrac{\mathbf{P}(\varphi_i) \quad 1 \leq i \leq n}{\mathbf{P}(\Box(\bigvee_{1 \leq i \leq n} \varphi_i))}$

(P) $\dfrac{\mathbf{P}(\varphi_i) \quad 1 \leq i \leq n}{\mathbf{P}([\Box \bigvee_{1 \leq i \leq n} \varphi_i] \wedge \bigwedge_{1 \leq i \leq n} \Diamond \varphi_i)}$

When $\mathbf{P}(\varphi)$ we say φ is a prime assertion. According to the definition, for example, $\mathit{inl}\,\varphi \wedge \mathit{inr}\,\psi$ is not a prime assertion even φ and ψ are. $\Box\varphi \wedge \Box\psi$ is not a prime assertion either. It is worth noticing that whether an assertion is prime or not is purely a syntactical question. Note that even when $[\![\varphi]\!]$ is a prime open set, it is not necessary true that $\mathbf{P}(\varphi)$. On the other hand, however, to pick up enough prime open sets we must have

$$\forall \varphi. [\![\varphi]\!] \text{ prime open} \implies \exists \psi. \varphi = \psi \,\&\, \mathbf{P}(\psi).$$

This is a consequence of the completeness theorem. $\mathbf{P}$ and Π^* are given mutual recursively.

All the assertions appearing in the following $\Pi^\star$ rules are assumed to be prime.

$\Pi^\star$ Rules

$(\mathbf{t})$ $\quad \Pi^\star_\sigma(\{\mathbf{t}\}) = \{\mathbf{t}\}$

$(\bot)$ $\quad \dfrac{\Pi^\star_\sigma(\{\varphi \mid (\varphi)_\bot \in A\}) = P}{\Pi^\star_{\sigma_\bot} A = \{(\varphi)_\bot \mid \varphi \in P\}}$

$(\times)$ $\quad \dfrac{\Pi^\star_\sigma(\{\varphi \mid \exists\psi.\, \varphi \times \psi \in A\}) = P \qquad \Pi^\star_\tau(\{\psi \mid \exists\varphi.\, \varphi \times \psi \in A\})) = Q}{\Pi^\star_{\sigma\times\tau}(A) = \{\varphi \times \psi \mid \varphi \in P \ \&\ \psi \in Q\}}$

$(+)$ $\quad \dfrac{\Pi^\star_\sigma(\{\varphi \mid \mathit{inl}\,\varphi \in A\}) = U \qquad \Pi^\star_\tau(\{\varphi \mid \mathit{inr}\,\varphi \in A\}) = V}{\Pi^\star_{\sigma+\tau}(A) = \{\mathit{inl}\,\varphi \mid \varphi \in U\} \cup \{\mathit{inr}\,\varphi \mid \varphi \in V\}}$

$(\to)$ $\quad \dfrac{\Pi^\star_\sigma(\bigcup_{1\le i\le n} \pi_0\alpha_i) = A' \qquad \Pi^\star_\tau(\bigcup_{1\le i\le n} \pi_1\alpha_i) = B'}{\Pi^\star_{\sigma\to\tau}\{\alpha_i \mid 1 \le i \le n\} = \{\beta \mid \mathbf{P}(\beta)\ \&\ \pi_0\beta \subseteq A'\ \&\ \pi_1\beta \subseteq B'\}}$

$(\Diamond)$ $\quad \Pi^\star_{\mathcal{P}_H(\sigma)}(A) = \{\bigwedge \Phi \mid \Phi \subseteq A\}$

$(\Box)$ $\quad \dfrac{\Pi^\star_\sigma(\bigcup\{\{\varphi_i \mid 1 \le i \le n\} \mid \Box \bigvee_{1\le i\le n} \varphi_i \in A\}) = T}{\Pi^\star_{\mathcal{P}_S(\sigma)}(A) = \{\Box\bigvee\Phi \mid \Phi \subseteq T\}}$

$(\Diamond - \Box)$ $\quad \dfrac{\Pi^\star_\sigma\{\varphi \mid \exists\alpha \in A.\, (\alpha \equiv \Box\bigvee\Phi \wedge \bigwedge_{\psi\in\Phi} \Diamond\psi\ \&\ \varphi \in \Phi)\} = S}{\Pi^\star_{\mathcal{P}_P(\sigma)}(A) = \{\Box\bigvee\Phi \wedge \bigwedge_{\psi\in\Phi} \Diamond\psi \mid \Phi \subseteq S\}}$

Now we are able to complete the proof system by introducing several rules which make use of the meta-predicate and meta-function.

$(\to -\vee)$ $\quad \dfrac{\mathbf{P}(\varphi)}{\varphi \to \bigvee_{i\in I}\alpha_i = \bigvee_{i\in I}\varphi \to \alpha_i}$

$(\neg - \to)$ $\quad \dfrac{\mathbf{P}(\bigwedge_{i\in I}\varphi_i \to \psi_i) \quad \mathbf{P}(\alpha) \quad \mathbf{P}(\beta) \quad \bigwedge_{\varphi_i \le \alpha}\psi_i \not\le \beta}{\bigwedge_{i\in I}\varphi_i \to \psi_i \not\le \alpha \to \beta}$

$(\neg - \mathbf{P})$ $\quad \dfrac{\mathbf{P}(\varphi) \qquad \forall i \in I.\, \varphi \not\le \psi_i}{\varphi \not\le \bigvee_{i\in I}\psi_i}$

Notation. If for an assertion φ, $[\![\varphi]\!]$ is a prime open set then there is a finite element a such that $[\![\varphi]\!] = a\uparrow$. Write $\check{\varphi}$ for such an a.

Notation. If for an assertion φ, $[\![\varphi]\!]$ is a prime open set then there is a finite element a such that $[\![\varphi]\!] = a\uparrow$. Write $\check{\varphi}$ for such an a.

Theorem 5.4.1 For any type expression σ and assertion $\varphi : \sigma$, $\mathbf{P}(\varphi)$ implies that $[\![\varphi]\!]_\sigma$ is a (non-empty) prime open set.

Let A be a finite set of prime assertions. If $\Pi^*(A) = B$ is derivable from the Π^* rules, then $A \subseteq B$ and B is quasi-$\wedge$ closed.

Proof The first part can be shown by structured induction. For illustration let us check the rule for function space. Other cases are straightforward. Suppose the assumptions for rule $(\rightarrow)$ of Prime Assertions hold. Clearly, then, $\bigsqcup_{i\in I}[\check{\varphi_i}, \check{\psi_i}]$ is a step function (Definition 2.5.2). It is easy to see that this step function is the least function in the set $\bigcap_{i\in I}[\![\varphi_i,]\!] \rightarrow [\![\psi_i,]\!]$. Hence $[\![\bigwedge_{i\in I}\varphi_i \rightarrow \psi_i]\!]$ is prime open.

The proof of the second part is also done by structured induction. We know that **SFP** is equivalent to **SFIS**. So all the cases in the induction follow from Theorem 4.3.1, Proposition 4.3.3, Proposition 4.4.1, Proposition 4.4.3, and Proposition 4.4.5. ∎

It is obvious that the proof rules $(\rightarrow -\vee)$, $(\neg - \rightarrow)$, and $\neg - \mathbf{P}$) using $\mathbf{P}$ or Π^* are sound. Therefore the whole proof system we have now is still sound.

Definition 5.4.2 Write $\mathcal{P}_\sigma$ for the proof system associated with type σ. $\mathcal{P}_\sigma$ is called *prime complete* if it has property p_0, *prime normal* if it has property p_1, and *complete* if it has property p_2, where

(p_0) $\quad \forall\varphi, \psi : \sigma.\ (\mathbf{P}(\varphi)\ \&\ \mathbf{P}(\psi)\ \&\ [\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma \Longrightarrow \vdash \varphi \leq \psi)\ \&$

$\quad (\mathbf{P}(\varphi)\ \&\ \mathbf{P}(\psi)\ \&\ [\![\varphi]\!]_\sigma \not\subseteq [\![\psi]\!]_\sigma \Longrightarrow \vdash \varphi \not\leq \psi)$

(p_1) $\quad \varphi : \sigma \Longrightarrow \exists\{\varphi_i \mid i \in I\}.\ \forall i \in I.\ \mathbf{P}(\varphi_i)\ \&\ \vdash \varphi = \bigvee_{i\in I}\varphi_i$

(p_2) $\quad \forall\varphi, \psi : \sigma.\ ([\![\varphi]\!] \subseteq [\![\psi]\!] \Longrightarrow \vdash \varphi \leq \psi)\ \&\ ([\![\varphi]\!] \not\subseteq [\![\psi]\!] \Longrightarrow \vdash \varphi \not\leq \psi)$

In (p_1), $\bigvee_{i\in I}\varphi_i$ is called a *prime normal form* of φ.

Clearly $\mathcal{P}_1$ has property p_0, p_1, and p_2. Our goal in this section is to prove the completeness of the proof system. This is achieved by showing that each type construction preserves property (p_0), (p_1), and (p_2), by several propositions. Note that for each case

the proof of (p_2) is routine: It follows from (p_0) and (p_1) directly. For the negative formulae, for example, we can use (p_1) to reduce assertions to their prime normal forms, $\vdash \varphi = \bigvee_{i\in I} \varphi_i$ and $\vdash \psi = \bigvee_{j\in J} \psi_j$. We then have

$$\begin{aligned}
\models \varphi \not\leq \psi &\Longrightarrow [\![\textstyle\bigvee_{i\in I} \varphi_i]\!] \not\subseteq [\![\textstyle\bigvee_{j\in J} \psi_j]\!] && \text{(Definition)} \\
&\Longrightarrow \exists i . \forall j . [\![\varphi_i]\!] \not\subseteq [\![\psi_j]\!] \\
&\Longrightarrow \exists i \forall j . \vdash \varphi_i \not\leq \psi_j && (p_0) \\
&\Longrightarrow \exists i . \vdash \varphi_i \not\leq \textstyle\bigvee_{j\in J} \psi_j && (\neg - \mathbf{P}) \\
&\Longrightarrow \vdash \textstyle\bigvee_{i\in I} \varphi_i \not\leq \bigvee_{j\in J} \psi_j && (\vee - \neg)
\end{aligned}$$

Similarly for positive formulae.

The following formula, obviously derivable from logical axioms and rules, will be frequently used for exchanging position of $\bigvee$ and $\bigwedge$.

$$(Ex) \qquad \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} \varphi_{i,j} = \bigvee_{f\in K} \bigwedge_{i=1}^{n} \varphi_{i,f(i)}$$

with

$$K = \{ f : \overline{n} \to \overline{\max\{ n_i \mid 1 \leq i \leq n \}} \mid \forall i . f(i) \in \overline{n_i} \}$$

where $\overline{s} = \{ 1, 2, \cdots, s \}$.

We abbreviate $\vdash A = B = C \cdots$ for $\vdash A = B \ \& \ \vdash B = C \ \& \ \vdash C = \cdots$.

Proposition 5.4.1 Product preserves property p_0, p_1, and p_2.

Proof (p_0).

$$\begin{aligned}
[\![\varphi \times \psi]\!]_{\sigma\times\tau} \subseteq [\![\varphi' \times \psi']\!]_{\sigma\times\tau} &\Longrightarrow [\![\varphi]\!]_\sigma \subseteq [\![\varphi']\!]_\sigma \ \& \ [\![\psi]\!]_\sigma \subseteq [\![\psi']\!]_\tau \\
&\Longrightarrow \vdash \varphi \leq \varphi' \ \& \ \vdash \psi \leq \psi' \\
&\Longrightarrow \vdash \varphi \times \psi \leq \varphi' \times \psi' \\
[\![\varphi \times \psi]\!]_{\sigma\times\tau} \not\subseteq [\![\varphi' \times \psi']\!]_{\sigma\times\tau} &\Longrightarrow [\![\varphi]\!]_\sigma \not\subseteq [\![\varphi']\!]_\sigma \text{ or } [\![\psi]\!]_\sigma \not\subseteq [\![\psi']\!]_\tau \\
&\Longrightarrow \vdash \varphi \not\leq \varphi' \text{ or } \vdash \psi \not\leq \psi' \\
&\Longrightarrow \vdash \varphi \times \psi \not\leq \varphi' \times \psi'
\end{aligned}$$

(p_1). Suppose $\alpha : \sigma \times \tau$ is an assertion.

$$\begin{aligned}
\vdash \alpha &= \bigvee_{1\le i\le n} \bigwedge_{1\le j\le m} \varphi_{ij} \times \psi_{ij} && (Ex)\\
&= \bigvee_{1\le k\le p} \varphi_k \times \psi_k && (\times - \wedge)\\
&= \bigvee_{1\le k\le q} (\bigvee_i \varphi_{ik}) \times (\bigvee_j \varphi_{jk}) && \text{asumption, } \varphi_{ik},\ \varphi_{jk} \text{ are prime}\\
&= \bigvee_k \bigvee_i \bigvee_j \varphi_{ik} \times \psi_{jk}, && (\times - \vee)
\end{aligned}$$

a finite disjunction of prime assertions. ∎

Proposition 5.4.2 Sum preserves property p_0, p_1, and p_2.

Proof (p_0)

$$\begin{aligned}
[\![\dot{inl}\,\varphi]\!]_{\sigma+\tau} \subseteq [\![\dot{inl}\,\varphi']\!]_{\sigma+\tau} &\Longrightarrow [\![\varphi]\!]_\sigma \subseteq [\![\varphi']\!]_\sigma\\
&\Longrightarrow \vdash \varphi \le \varphi'\\
&\Longrightarrow \vdash \dot{inl}\,\varphi \le \dot{inl}\,\varphi'
\end{aligned}$$

Similarly for assertions of the form $\dot{inr}\,\varphi$ and the negative formulae. Note that the cases for $[\![\dot{inl}\,\varphi]\!]_{\sigma+\tau} \subseteq [\![\dot{inr}\,\varphi']\!]_{\sigma+\tau}$ and $[\![\dot{inr}\,\varphi]\!]_{\sigma+\tau} \subseteq [\![\dot{inl}\,\varphi']\!]_{\sigma+\tau}$ are trivial.

(p_1) Suppose $\alpha : \sigma + \tau$ is an assertion.

$$\begin{aligned}
\vdash \alpha &= \bigvee_{1\le i\le n} (\bigwedge_{1\le j\le m} \dot{inl}\,\varphi_{ij} \wedge \bigwedge_{1\le k\le w} \dot{inr}\,\psi_{ik}) && (Ex), (\dot{inl} - \vee), (\dot{inr} - \vee)\\
&= \bigvee_{1\le i\le n} (\dot{inl} \bigwedge_{1\le j\le m} \varphi_{ij} \wedge \dot{inr} \bigwedge_{1\le k\le w} \psi_{ik}) && (\dot{inl} - \wedge), (\dot{inr} - \wedge)\\
&= \bigvee_{1\le i\le n} (\dot{inl} \bigvee_p \varphi'_{ip} \wedge \dot{inr} \bigvee_q \psi'_{iq}) && \text{assumption, } \varphi'_{ip},\ \psi'_{iq} \text{ are prime}\\
&= \bigvee_{1\le i\le n} \bigvee_p \bigvee_q (\dot{inl}\,\varphi'_{ip} \wedge \dot{inr}\,\psi'_{iq}) && (\dot{inl} - \vee),\ (\dot{inr} - \vee)
\end{aligned}$$

The conclusion then follows from the rule $(\neg + \mathbf{f})$ for sum. ∎

Proposition 5.4.3 Lifting preserves property p_0, p_1, and p_2.

Proof Trivial.

Proposition 5.4.4 Function space preserves property p_0, p_1, and p_2.

Proof (p_0). For positive formulae, we have

$$[\![\bigwedge_{1\le i\le n} \varphi_i \to \psi_i]\!]_{\sigma\to\tau} \subseteq [\![\bigwedge_{1\le j\le m} \varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \bigcap_{1\le i\le n} [\![\varphi_i \to \psi_i]\!]_{\sigma\to\tau} \subseteq \bigcap_{1\le j\le m} [\![\varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \forall j.\ \bigcap_{1\le i\le n} [\![\varphi_i \to \psi_i]\!]_{\sigma\to\tau} \subseteq [\![\varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \forall j.\ \bigcap_{1\le i\le n} [\,\check{\varphi}_i\uparrow \to \check{\psi}_i\uparrow\,] \subseteq [\,\check{\varphi}'_j\uparrow \to \check{\psi}'_j\uparrow\,]$$

$$\forall j.\ \bigsqcup_{1\le i\le n} [\,\check{\varphi}_i,\ \check{\psi}_i\,] \sqsupseteq [\,\check{\varphi}'_j,\ \check{\psi}'_j\,]$$

$$\Longrightarrow \forall j\ \exists i.\ \check{\psi}_i \sqsupseteq \check{\psi}'_j\ \&\ \check{\varphi}_i \sqsubseteq \check{\varphi}'_i$$

$$\Longrightarrow \forall j\ \exists i.\ \vdash \psi_i \le \psi'_j\ \&\ \vdash \varphi_i \ge \varphi'_j \qquad (\text{ by assumption })$$

$$\Longrightarrow \forall j\ \exists i.\ \vdash \varphi_i \to \psi_i \le \varphi'_j \to \psi'_j \qquad (\to - \le)$$

$$\Longrightarrow \forall j.\ \vdash \bigwedge_{1\le i\le n} \varphi_i \to \psi_i\ \le \varphi'_j \to \psi'_j$$

$$\Longrightarrow \vdash \bigwedge_{1\le i\le n} \varphi_i \to \psi_i \le \bigwedge_{1\le j\le m} \varphi'_j \to \psi'_j$$

For negative formulae, we have

$$[\![\bigwedge_{1\le i\le n} \varphi_i \to \psi_i]\!]_{\sigma\to\tau} \not\subseteq [\![\bigwedge_{1\le j\le m} \varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \bigcap_{1\le i\le n} [\![\varphi_i \to \psi_i]\!]_{\sigma\to\tau} \not\subseteq \bigcap_{1\le j\le m} [\![\varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \exists j.\ \bigcap_{1\le i\le n} [\![\varphi_i \to \psi_i]\!]_{\sigma\to\tau} \not\subseteq [\![\varphi'_j \to \psi'_j]\!]_{\sigma\to\tau}$$

$$\Longrightarrow \bigcap_{1\le i\le n} [\check{\varphi}_i\uparrow \to \check{\psi}_i\uparrow] \not\subseteq [\check{\varphi}'_j\uparrow \to \check{\psi}'_j\uparrow]$$

$$\Longrightarrow \bigsqcup_{1\le i\le n} [\check{\varphi}_i, \check{\psi}_i] \not\supseteq [\check{\varphi}'_j, \check{\psi}'_j]$$

$$\Longrightarrow \bigsqcup_{\check{\varphi}_i \sqsupseteq \check{\varphi}'_j} \check{\psi}_i \not\supseteq \check{\psi}'_j$$

$$\Longrightarrow \vdash \bigwedge_{\vdash \varphi_i \le \varphi'_j} \psi_i \not\le \psi'_j$$

$$\Longrightarrow \vdash \bigwedge_{1\le i\le n} \varphi_i \to \psi_i \not\le \varphi'_j \to \psi'_j \qquad (\neg - \to)$$

$$\Longrightarrow \vdash \bigwedge_{1\le i\le n} \varphi_i \to \psi_i \not\le \bigwedge_{1\le j\le m} \varphi'_j \to \psi'_j \qquad (\text{ logical rule })$$

(p_1). By using rule $(\to -\vee)$, $(\to -\wedge)$, $(\vee - \to)$ as well as (Ex), we can put each $\theta : \sigma \to \tau$ in the following form:

$$\vdash \theta = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} (\varphi_{i,j} \to \psi_{i,j})$$

where $\varphi_{i,j}$, $\psi_{i,j}$ are prime assertions of σ and τ, respectively. It is then enough to show that each assertion of the form $\bigwedge_{1\le i\le n} (\varphi_i \to \psi_i)$ is equivalent to a finite disjunction of prime assertions. First an observation.

Observation: Suppose $\vdash \varphi_i \leq \varphi_j$. We have

$$\begin{aligned}
&\vdash (\varphi_i \to \psi_i) \wedge (\varphi_j \to \psi_j) \\
&= (\varphi_i \to \psi_i) \wedge (\varphi_j \to \psi_j) \wedge (\varphi_i \wedge \varphi_j \to \psi_i) \\
&= (\varphi_i \to \psi_i) \wedge (\varphi_j \to \psi_j) \wedge (\varphi_i \wedge \varphi_j \to \psi_i) \wedge (\varphi_i \wedge \varphi_j \to \psi_j) \\
&= (\varphi_i \to \psi_i) \wedge (\varphi_j \to \psi_j) \wedge (\varphi_i \wedge \varphi_j \to \psi_i \wedge \psi_j) \\
&= (\varphi_j \to \psi_j) \wedge (\varphi_i \to \psi_i \wedge \psi_j) \\
&= (\varphi_j \to \psi_j) \wedge (\varphi_i \to \bigvee_k \psi'_k) \qquad \text{where } \vdash \psi_i \wedge \psi_j = \bigvee_k \psi'_k,\ \ \psi'_k \text{ prime} \\
&= \bigvee_k [(\varphi_i \to \psi'_k) \wedge (\varphi_j \to \psi_j)]
\end{aligned}$$

where $\vdash \psi'_k \leq \psi_j$ for each k. Based on this observation we can assume $\bigwedge_{1 \leq i \leq n} (\varphi_i \to \psi_i)$ to be such that $\vdash \varphi_i \leq \varphi_j \Longrightarrow \vdash \psi_i \leq \psi_j$.

Let

$$\Pi^*\{\, \varphi_i \mid 1 \leq i \leq n \,\} = \{\, \alpha_p \mid p \in P \,\}$$

and

$$\Pi^*\{\, \psi_j \mid 1 \leq j \leq m \,\} = \{\, \beta_q \mid q \in Q \,\}.$$

We have

$$\begin{aligned}
&\vdash \bigwedge_{1 \leq i \leq n} (\varphi_i \to \psi_i) \\
&= \bigwedge_{p \in P} (\alpha_p \to \bigwedge_{\vdash \alpha_p \leq \varphi_i} \psi_i) \\
&= \bigwedge_{p \in P} (\alpha_p \to \bigvee \beta_{pi}) \\
&\qquad\qquad (\vdash \bigwedge_{\vdash \alpha_p \leq \varphi_i} \psi_i = \bigvee \beta_{pi} \ \ \& \ \ \beta_{pi} \in \{\, \beta_q \mid q \in Q \,\}) \\
&= \bigwedge_{p \in P} \bigvee_i (\alpha_p \to \beta_{pi}) \\
&=^{\star} \bigvee_k (\bigwedge_{p \in P} \alpha_p \to \beta_{pk})
\end{aligned}$$

$\star$: where each $\bigwedge_{p \in P} \alpha_p \to \beta_{pk}$ is a prime assertion of $\sigma \to \tau$, by virtue of the observation above and the formula (Ex). ∎

For Smyth and Hoare powerdomains (p_1) is routine to prove. (p_0) for negative formulae is also simple. Hence for the following two propositions we only show (p_0) for positive formulae.

Proposition 5.4.5 Smyth powerdomain preserves property p_0, p_1, and p_2.

Proof

$$[\![\Box \bigvee_{1\le i\le n} \varphi_i]\!]_{\mathcal{P}_S\sigma} \subseteq [\![\Box \bigvee_{1\le j\le m} \psi_j]\!]_{\mathcal{P}_S\sigma}$$

$$\Longrightarrow \{ U \in \mathcal{P}_S(\mathcal{D}\sigma) \mid U \subseteq [\![\bigvee_{1\le i\le n} \varphi_i]\!]_\sigma \} \subseteq \{ U \in \mathcal{P}_S(\mathcal{D}\sigma) \mid U \subseteq [\![\bigvee_{1\le j\le m} \psi_j]\!]_\sigma \}$$

$$\Longrightarrow \{ \check{\varphi}_i \mid 1 \le i \le n \} \sqsupseteq_0 \{ \check{\psi}_j \mid 1 \le j \le m \}$$

$$\Longrightarrow \forall i\ \exists j.\ \check{\varphi}_i \sqsupseteq \check{\psi}_j$$

$$\Longrightarrow \forall i\ \exists j.\ [\![\varphi_i]\!]_\sigma \subseteq [\![\psi_j]\!]_\sigma$$

$$\Longrightarrow \forall i\ \exists j.\ \vdash \varphi_i \le \psi_j \qquad \text{assumption}$$

$$\Longrightarrow \forall i.\ \vdash \varphi_i \le \bigvee_{1\le j\le m} \psi_j$$

$$\Longrightarrow \vdash \bigvee_{1\le i\le n} \varphi_i \le \bigvee_{1\le j\le m} \psi_j$$

$$\Longrightarrow \vdash \Box \bigvee_{1\le i\le n} \varphi_i \le \Box \bigvee_{1\le j\le m} \psi_j$$

∎

Proposition 5.4.6 Hoare powerdomain preserves property p_0, p_1, and p_2.

Proof

$$[\![\bigwedge_{1\le i\le n} \Diamond\varphi_i]\!]_{\mathcal{P}_H\sigma} \subseteq [\![\bigwedge_{1\le j\le m} \Diamond\psi_j]\!]_{\mathcal{P}_H\sigma}$$

$$\Longrightarrow \bigcap_{1\le i\le n} [\![\Diamond\varphi_i]\!]_{\mathcal{P}_H\sigma} \subseteq \bigcap_{1\le j\le m} [\![\Diamond\psi_j]\!]_{\mathcal{P}_H\sigma}$$

$$\Longrightarrow \forall j.\ \bigcap_{1\le i\le n} [\![\Diamond\varphi_i]\!]_{\mathcal{P}_H\sigma} \subseteq [\![\Diamond\psi_j]\!]_{\mathcal{P}_H\sigma}$$

$$\Longrightarrow \forall j.\ \{\, \check{\varphi}_i \mid 1 \le i \le n \,\} \sqsupseteq_1 \{\, \check{\psi}_j \,\}$$

$$(\text{ as } Cl_H\{\, \check{\varphi}_i \mid 1 \le i \le n \,\} \in \bigcap_{1\le i\le n} [\![\Diamond\varphi_i]\!]_{\mathcal{P}_H\sigma})$$

$$\Longrightarrow \forall j\ \exists i.\ \check{\varphi}_i \sqsupseteq \check{\psi}_j$$

$$\Longrightarrow \forall j\ \exists i.\ \vdash \varphi_i \le \psi_j \qquad \text{by assumption}$$

$$\Longrightarrow \forall j\ \exists i.\ \vdash \Diamond\varphi_i \le \Diamond\psi_j$$

$$\Longrightarrow \forall j.\ \vdash \bigwedge_{1\le i\le n} \Diamond\varphi_i \le \Diamond\psi_j$$

$$\Longrightarrow \vdash \bigwedge_{1\le i\le n} \Diamond\varphi_i \le \bigwedge_{1\le j\le m} \Diamond\psi_j$$

∎

That the Plotkin powerdomain preserves (p_1) is not so direct to prove. By exchanging the position of $\bigvee$ and $\bigwedge$ we can reduce an assertion into a disjunction of conjunction of assertions $\Box \bigvee_i \varphi_i \wedge \bigwedge_j \Diamond\psi_j$, where φ_i's and ψ_j's are prime. We need to show that $\Box \bigvee_i \varphi_i \wedge \bigwedge_j \Diamond\psi_j$ is equivalent to a disjunction of prime assertions. Our idea is to show that, first, any $\Box \bigvee_i \varphi_i$ is equivalent to a disjunction of prime assertions. We then show that if θ is prime, then $\theta \wedge \Diamond\psi$ is equivalent to a disjunction of prime assertions. Applying this result as many time as necessary, we can 'absorb' any $\Diamond\psi$ in $\theta \wedge \Diamond\psi$ if it is not already a prime. The following lemmas show some results which are a bit stronger than we need. Note we assume that $\mathcal{P}_\sigma$ has property (p_0), (p_1), and (p_2) in the lemmas given below.

Lemma 5.4.1 Let $\varphi_i : \sigma$, $1 \le i \le n$ be prime assertions. $\Box \bigvee_{1\le i\le n} \varphi_i$ is provable to be equivalent to a finite disjunction of prime assertions.

Proof By induction on n.

$n = 1$.

$$\begin{aligned}
\vdash \Box\varphi_1 &= \Box(\varphi_1 \vee \mathbf{f}) \\
&= \Box(\varphi_1 \vee \mathbf{f}) \wedge (\Diamond\varphi_1 \vee \Box\mathbf{f}) \qquad (\Box) \\
&= [\Box(\varphi_1 \vee \mathbf{f}) \wedge \Diamond\varphi_1] \vee [\Box(\varphi_1 \vee \mathbf{f}) \wedge \Box\mathbf{f}] \\
&= \Box\varphi_1 \wedge \Diamond\varphi_1.
\end{aligned}$$

Induction step.

$$\begin{aligned}
\vdash \Box(\bigvee_{i=1}^{n} \varphi_i) &= \Box(\bigvee_{i=1}^{n} \varphi_i) \wedge (\Diamond\varphi_1 \vee \Box(\bigvee_{2\le i\le n} \varphi_i)) \qquad (\Box) \\
&= (\Diamond\varphi_1 \wedge \Box \bigvee_{i=1}^{n} \varphi_i) \vee \Box(\bigvee_{2\le i\le n} \varphi_i) \\
&= (\Diamond\varphi_1 \wedge \Diamond\varphi_2 \wedge \Box \bigvee_{i=1}^{n} \varphi_i) \vee \Box(\bigvee_{i\neq 1} \varphi_i) \vee \Box(\bigvee_{i\neq 2} \varphi_i) \\
&\cdots \\
&= (\Box \bigvee_{i=1}^{n} \varphi_i \wedge \bigwedge_{1\le i\le n} \Diamond\varphi_i) \vee \bigvee_{1\le i\le n} (\Box \bigvee_{j\neq i} \varphi_j)
\end{aligned}$$

By induction hypothesis each $\Box \bigvee_{j\neq i} \varphi_j$ has a prime normal form, hence so does $\Box \bigvee_{i=1}^{n} \varphi_i$. ∎

We can draw from the proof the following formula.

$$(\mathbf{P}-\Box) \qquad \Box \bigvee_{i\in I} \varphi_i = \bigvee_{J\subseteq I} (\Box \bigvee_{j\in J} \varphi_j \wedge \bigwedge_{j\in J} \Diamond\varphi_j)$$

By the same method used in the proof of Lemma 5.4.1 we can similarly prove

Lemma 5.4.2 Let $\varphi_i : \sigma$, $1 \le i \le n$ be prime assertions.

$$\Box \bigvee_{1\le i\le n} \varphi_i \wedge \bigwedge_{j\in J} \Diamond\varphi_j$$

is provable to be equivalent to a finite disjunction of prime assertions, where

$$J \subseteq \{1, 2, \cdots, n\}.$$

Lemma 5.4.3 Every assertion of the form

$$\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i$$

is equivalent to a finite disjunction of prime assertions, where φ_i's are not necessary prime.

Proof By assumption, for each i, there are ψ_{ij}, prime assertions such that

$$\vdash \varphi_i = \bigvee_{1 \le j \le n_i} \psi_{i,j}.$$

We have

$$\begin{aligned}
\vdash \Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i &= \Box\,(\bigvee_{i=1}^{n}\bigvee_{j=1}^{n_i} \psi_{i,j}) \wedge \bigwedge_{i=1}^{n} \Diamond(\bigvee_{j=1}^{n_i} \psi_{i,j}) \\
&= \Box(\bigvee_{i=1}^{n}\bigvee_{j=1}^{n_i} \psi_{i,j}) \wedge \bigwedge_{i=1}^{n}\bigvee_{j=1}^{n_i} \Diamond\psi_{i,j} \\
&= \bigvee_{f\in K} (\Box(\bigvee_{i=1}^{n}\bigvee_{j=1}^{n_i} \psi_{i,j}) \wedge \bigwedge_{i=1}^{n} \Diamond\psi_{i,f(i)})
\end{aligned}$$

where $K = \{f : \overline{n} \to \overline{\max\{n_i \mid 1 \le i \le n\}} \mid \forall i.\, f(i) \in \overline{n_i}\}$.

From Lemma 5.4.2 we know that each term in $\bigvee_{f\in K}$ is equivalent to a finite disjunction of prime assertions. ∎

Lemma 5.4.4 For any prime assertion

$$\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i,$$

$$(\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i) \wedge \Diamond\psi$$

is equivalent to a finite disjunction of prime assertions.

Proof By $(\Diamond)$ we see that

$$\begin{aligned}
&\vdash (\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i) \wedge \Diamond\psi = (\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i) \wedge \Diamond(\psi \wedge (\bigvee_{i=1}^{n} \varphi_i)) \\
&= \bigvee_{j=1}^{n} \Box((\bigvee_{i=1}^{n} \varphi_i) \vee (\varphi_j \wedge \psi)) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i \wedge \Diamond(\psi \wedge \varphi_j).
\end{aligned}$$

By Lemma 5.4.3 the above assertion can be equivalently transformed into a finite disjunction of prime assertions. ∎

Lemma 5.4.5 For any prime assertion

$$\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i,$$

$$(\Box(\bigvee_{i=1}^{n} \varphi_i) \wedge \bigwedge_{i=1}^{n} \Diamond\varphi_i) \wedge \Box\psi$$

is equivalent to a finite disjunction of prime assertions.

Proof

$$\vdash (\Box(\bigvee_{i=1}^{n}\varphi_i)\wedge\bigwedge_{i=1}^{n}\Diamond\varphi_i)\wedge\Box\psi = \Box((\bigvee_{i=1}^{n}\varphi_i)\wedge\psi)\wedge\bigwedge_{i=1}^{n}(\Diamond\varphi_i\wedge\Box\psi)$$
$$= \Box\bigvee_{i=1}^{n}(\varphi_i\wedge\psi)\wedge\bigwedge_{i=1}^{n}\Diamond(\varphi_i\wedge\psi)$$

By Lemma 5.4.4, the above assertion can be equivalently transformed into a finite disjunction of prime assertions. ∎

Now we have

Proposition 5.4.7 Plotkin powerdomain preserves property p_0, p_1, and p_2.

Proof Property (p_1) follows from Lemma 5.4.1 to 5.4.5. The proof of (p_0) for negative formulae is similar to that for positive formulae, which is given below.

$$[\![(\Box\bigvee_{1\le i\le n}\varphi_i)\wedge\bigwedge_{1\le i\le n}\Diamond\varphi_i]\!]_{\mathcal{P}_{P\sigma}} \subseteq [\![(\Box\bigvee_{1\le j\le m}\psi_j)\wedge\bigwedge_{1\le j\le m}\Diamond\psi_j]\!]_{\mathcal{P}_{P\sigma}}$$

$$\Longrightarrow [\![(\Box\bigvee_{1\le i\le n}\varphi_i)\wedge\bigwedge_{1\le i\le n}\Diamond\varphi_i]\!]_{\mathcal{P}_{P\sigma}} \subseteq [\![\Box\bigvee_{1\le j\le m}\psi_j]\!]_{\mathcal{P}_{P\sigma}}\cap\bigcap_{1\le j\le m}[\![\Diamond\psi_j]\!]_{\mathcal{P}_{P\sigma}}$$

$$\Longrightarrow \{\check{\varphi}_i \mid 1\le i\le n\}\sqsupseteq_{\approx_0}\{\check{\psi}_j \mid 1\le j\le m\}\ \&$$

$$\{\check{\varphi}_i \mid 1\le i\le n\}\sqsupseteq_{\approx_1}\{\check{\psi}_j \mid 1\le j\le m\}$$

$$(\text{ since } Cl_P(\{\check{\varphi}_i \mid 1\le i\le n\})\in[\![(\Box\bigvee_{1\le i\le n}\varphi_i)\wedge\bigwedge_{1\le i\le n}\Diamond\varphi_i]\!]_{\mathcal{P}_{P\sigma}})$$

$$\Longrightarrow \forall i\,\exists j.\ \vdash\varphi_i\le\psi_j\ \&\ \forall j\,\exists i.\ \vdash\varphi_i\le\psi_j$$

$$\Longrightarrow \vdash\Box\bigvee_{1\le i\le n}\varphi_i\le\Box\bigvee_{1\le j\le m}\psi_j\ \&\ \vdash\bigwedge_{1\le i\le n}\Diamond\varphi_i\le\bigwedge_{1\le j\le m}\Diamond\psi_j$$

$$\Longrightarrow \vdash\Box\bigvee_{1\le i\le n}\varphi_i\wedge\bigwedge_{1\le i\le n}\Diamond\varphi_i\le\Box\bigvee_{1\le j\le m}\psi_j\wedge\bigwedge_{1\le j\le m}\Diamond\psi_j$$

∎

The completeness of the proof system now follows from propositions 5.4.1 to 5.4.7 by a structured induction on type structures, with each proposition taking care of one case. Note that the case for recursively defined types reduces to finite types since we are only dealing with a finite set of assertions at each time, and these assertions can well be regarded as of some proper finite type. So we have

Theorem 5.4.2 Our proof system is complete.

For expressiveness of our assertion language we have

Theorem 5.4.3 For each σ,

$$[\![\ ]\!]_\sigma : (\mathcal{A}_\sigma/=,\ \leq_\sigma) \to (\mathbf{K\Omega}\mathcal{D}(\sigma),\ \subseteq)$$

is an isomorphism, where $\mathbf{K\Omega}(D)$ is the set of compact open sets of D.

Proof By the Completeness Theorem it is enough to show that

$$\forall P \in \mathbf{P\Omega}(\mathcal{D}(\sigma))\ \ \exists \varphi \in \mathcal{A}_\sigma.\ \ \mathbf{P}(\varphi)\ \&\ [\![\varphi]\!]_\sigma = P,$$

which can be routinely done by a structured induction. ∎

Corollary 5.4.1 For each type expression σ and for any pair φ, ψ of type σ, either $\vdash_\sigma \varphi \leq_\sigma \psi$ or $\vdash_\sigma \varphi \not\leq_\sigma \psi$.

Example 5.4.4 In Abramsky's framework for Scott domains, the conjunction of prime assertions is again prime. In this example we show that, in our framework, the conjunction of two prime assertions need not be equivalent to a single prime assertion. Consider

$$\mathcal{P}_P((\mathbf{1}_\perp + \mathbf{1}_\perp) \times (\mathbf{1}_\perp + \mathbf{1}_\perp)),$$

or $\mathcal{P}_P(2\mathbf{1}_\perp)^2$, as an abbreviation. Write

$$p \equiv \mathit{inl}\,\mathbf{t}_\perp : \mathbf{1}_\perp + \mathbf{1}_\perp,$$

$$q \equiv \mathit{inr}\,\mathbf{t}_\perp : \mathbf{1}_\perp + \mathbf{1}_\perp.$$

Obviously $\vdash p \wedge q \leq \mathbf{f}$ and

$$\Box(p \times \mathbf{t} \vee q \times \mathbf{t}) \wedge \Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t}$$

and

$$\Box(\mathbf{t} \times p \vee \mathbf{t} \times q) \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q$$

are prime assertions of $\mathcal{P}_P(2\mathbf{1}_\perp)^2$.

We have

$$\vdash \quad \Box(p \times \mathbf{t} \vee q \times \mathbf{t}) \wedge \Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t} \ \wedge \ \Box(\mathbf{t} \times p \vee \mathbf{t} \times q) \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q$$

$$= \quad \Box(p \times p \vee p \times q \vee q \times p \vee q \times q) \wedge \Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t} \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q$$

$$= \quad (\bigvee_i \alpha_i) \wedge \Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t} \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q$$

by formula ($\mathbf{P} - \Box$), α_i's are prime

$$= \quad \Box(p \times q \vee q \times p) \wedge \Diamond p \times q \wedge \Diamond q \times p$$

$$\vee \ \Box(p \times p \vee q \times q) \wedge \Diamond p \times p \wedge \Diamond q \times q$$

$$\vee \ \Box(p \times q \vee q \times p \vee p \times p) \wedge \Diamond p \times q \wedge \Diamond q \times p \wedge \Diamond p \times p$$

$$\vee \ \Box(p \times q \vee q \times p \vee q \times q) \wedge \Diamond p \times q \wedge \Diamond q \times p \wedge \Diamond q \times q$$

$$\vee \ \Box(p \times q \vee q \times q \vee p \times p) \wedge \Diamond p \times q \wedge \Diamond q \times q \wedge \Diamond p \times p$$

$$\vee \ \Box(q \times p \vee q \times q \vee p \times p) \wedge \Diamond q \times p \wedge \Diamond q \times q \wedge \Diamond p \times p$$

$$\vee \ \Box(q \times p \vee p \times q \vee q \times q \vee p \times p) \wedge \Diamond q \times p \wedge \Diamond p \times q \wedge \Diamond q \times q \wedge \Diamond p \times p$$

Remarks.

1. By formula ($\mathbf{P} - \Box$), all possible primes formed from subset of $\{ \varphi_i \mid i \in I \}$ should be in the disjunction. We ommited some of them in the last formula. But the equivalence still holds, because we have, for example,

$$\vdash \Box(p \times p \vee p \times q) \wedge \Diamond q \times \mathbf{t} \leq \Diamond(p \times p \vee p \times q) \wedge q \times \mathbf{t} \leq \mathbf{f},$$

that is, all the omitted assertions are provable to be equivalent to $\mathbf{f}$ when put in conjunction with

$$\Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t} \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q.$$

2. For those α_i's which remain in the last derivation,

$$\vdash \alpha \wedge (\Diamond p \times \mathbf{t} \wedge \Diamond q \times \mathbf{t} \wedge \Diamond \mathbf{t} \times p \wedge \Diamond \mathbf{t} \times q) = \alpha$$

because of obvious reasons such as

$$\vdash \Diamond p \times q \wedge \Diamond p \times \mathbf{t} = \Diamond p \times q.$$

3. The prime assertions in the last formula are pairwise inconsistent. We have, for example,

$$\begin{aligned}
&\vdash \Box(p \times q \vee q \times q \vee p \times p) \wedge \Diamond p \times q \wedge \Diamond q \times q \wedge \Diamond p \times p \\
&\qquad\qquad \wedge \ \Box(q \times p \vee q \times q \vee p \times p) \wedge \Diamond q \times p \wedge \Diamond q \times q \wedge \Diamond p \times p \\
&\leq \Box(p \times q \vee q \times q \vee p \times p) \wedge \Diamond q \times p \\
&\leq \Diamond q \times p \ \wedge \ (p \times q \vee q \times q \vee p \times p) \\
&\leq \mathbf{f}
\end{aligned}$$

4. We really need a machine to carry out this example!

5.5 An Application

As an application, we show how our framework can be used to derive the assertions used in Brookes' proof system [Br85]. In [Br85] Brookes proposed a proof system for a parallel programming language with the syntax of commands specified by

$$\Gamma ::= Skip \mid I := E \mid \Gamma_1; \Gamma_2 \mid \Gamma_1 \mid\mid \Gamma_2 \mid await\ B\ then\ \Gamma$$

$$\mid if\ B\ then\ \Gamma_1\ else \Gamma_2 \mid while\ B\ do\ \Gamma$$

He gave an operational semantics for this programming language by a set of labeled transition rules of the form $\langle \Gamma, \sigma \rangle \xrightarrow{\alpha} \langle \Gamma', \sigma' \rangle$ with the standard interpretation, where Γ's are commands and σ's are states. A proof system was proposed using assertions of the form

$$\varphi ::= P \Sigma_{i=1}^{n} \alpha_i P_i \varphi_i.$$

We show that the assertions Brookes used can be derived from the domain for the denotational semantics of the parallel programming language, *i.e.* , the domain of resumptions

$$R \cong S \to \mathcal{P}_P[(S \times R) + S]$$

proposed by Plotkin [Pl76]. S is the domain of states. R corresponds to the type $r = rec\ t.(\sigma_0 \to \mathcal{P}_P[(\sigma_0 \times t) + \sigma_0])$, where σ_0 is the type for the domain of states. Note that the domain of integers is the solution to the domain equation

$$X \cong \mathbf{1}_\perp + X$$

corresponding to the type $rec\ x.\mathbf{1}_\perp + x$. Therefore, we may write $\sigma_0 = (\ rec\ x.\mathbf{1}_\perp + x)^N$, the product of N copies of the integer domain. According to the framework presented in the previous sections, assertions of type σ_0 are given by

$$P ::= a \mid P \wedge P \mid \quad P \vee P \mid \quad \mathbf{t} \quad \mid \quad \mathbf{f}$$

where a is some atomic assertions indicating what value is in each store for the current state. Assertions of type $(\sigma_0 \times r) + \sigma_0$ are given by

$$u ::= \dot{inr}(P) \mid \quad \dot{inl}(P \times \varphi) \mid \quad u \wedge u \mid \quad u \vee u \mid \quad \mathbf{t} \quad \mid \quad \mathbf{f}$$

assertions of type $P_P[(\sigma_0 \times r) + \sigma_0]$ by

$$w ::= \Box u \mid \Diamond u \mid w \wedge w \mid w \vee w \mid \quad \mathbf{t} \quad \mid \quad \mathbf{f}$$

Finally, assertions of type r are given by

$$\varphi ::= P \rightarrow w \mid \varphi_1 \wedge \varphi_2 \mid \quad \varphi_1 \vee \varphi_2 \mid \quad \mathbf{t} \quad \mid \quad \mathbf{f}.$$

Note that this is a recursive definition. Now we can see that the following form of assertion

$$P \rightarrow \Box(\bigvee_{i=1}^{n} \dot{inl}\ (P_i \times \varphi_i)) \wedge \bigwedge_{i=1}^{n} \Diamond \dot{inl}\ (P_i \times \varphi_i)$$

constructed from the above rules plays the same role as assertions

$$P \Sigma_{i=1}^{n} \alpha_i P_i \varphi_i$$

do in Brookes' proof system. In fact, our framework gives a definition of a satisfaction relation between a command and an assertion. Γ satisfies

$$P \rightarrow \Box(\bigvee_{i=1}^{n} \dot{inl}(P_i \times \varphi_i)) \wedge \bigwedge_{i=1}^{n} \Diamond \dot{inl}(P_i \times \varphi_i)$$

write

$$\Gamma \models P \rightarrow \Box(\bigvee_{i=1}^{n} \dot{inl}(P_i \times \varphi_i)) \wedge \bigwedge_{i=1}^{n} \Diamond \dot{inl}(P_i \times \varphi_i),$$

if

$$[\![\, \Gamma \,]\!] \in [\![\, P \rightarrow \Box(\bigvee_{i=1}^{n} \dot{inl}\ (P_i \times \varphi_i)) \wedge \bigwedge_{i=1}^{n} \Diamond \dot{inl}\ (P_i \times \varphi_i) \,]\!].$$

But this is the same as saying, in terms of operational semantics, that

$$\forall \sigma.(\sigma \models P) \text{ implies } \{\langle \Gamma', \sigma' \rangle \quad \mid \ \langle \Gamma, \sigma \rangle \rightarrow \langle \Gamma', \sigma' \rangle\} \models \Box(\bigvee_{i=1}^{n} \dot{inl}\ (P_i \times \varphi_i)),$$

i.e. ,

$$\forall\sigma.(\sigma \models P)\&\langle\Gamma,\sigma\rangle \rightarrow \langle\Gamma',\sigma'\rangle \text{ implies } \langle\Gamma',\sigma'\rangle \models (\bigvee_{i=1}^{n} \dot{inl}\,(P_i \times \varphi_i))$$

and, at the same time,

$$\forall\sigma.(\sigma \models P) \text{ implies } \{\langle\Gamma',\sigma'\rangle \quad | \quad \langle\Gamma,\sigma\rangle \rightarrow \langle\Gamma',\sigma'\rangle\} \models \bigwedge_{i=1}^{n} \Diamond\dot{inl}\,(P_i \times \varphi_i)$$

i.e. ,

$$\forall i\ \forall\sigma.\ (\sigma \models P) \text{ implies } \{\langle\Gamma',\sigma'\rangle \quad | \quad \langle\Gamma,\sigma\rangle \rightarrow \langle\Gamma',\sigma'\rangle\} \models \Diamond\dot{inl}\,(P_i \times \varphi_i).$$

In summary,

$$\Gamma \models P \rightarrow \Box(\bigvee_{i=1}^{n} \dot{inl}\,(P_i \times \varphi_i)) \wedge \bigwedge_{i=1}^{n} \Diamond\dot{inl}\,(P_i \times \varphi_i)$$

if

1) $\forall\sigma.\ (\sigma \models P)\&\ \langle\Gamma,\sigma\rangle \rightarrow \langle\Gamma',\sigma'\rangle$ implies $\exists i,\ 1 \leq i \leq n$ s.t. $\sigma' \models P_i \& \Gamma' \models \varphi_i$ and

2) $\forall i \forall\sigma.(\sigma \models P)$ implies $\exists\langle\Gamma',\sigma'\rangle$ such that $\langle\Gamma,\sigma\rangle \rightarrow \langle\Gamma',\sigma'\rangle\&\ \Gamma' \models \varphi_i \& \sigma' \models P_i$.

This is exactly a restatement of the definition for

$$\models \Gamma \textbf{ sat } \ P\Sigma_{i=1}^{n}\alpha_i P_i\varphi_i$$

in [Br85] without using labels. A final remark: the assertions in our framework are propositional, corresponding to the compact-open sets. Therefore, they are not exactly the same as those used by Brookes. Nevertheless, it is an important first step to be able to derive the assertions of the right propositional form from the domain for denotational semantics.

Chapter 6

Mu-Calculus of Domain Theory

In Chapter 5 we have shown that domains are related to a kind of domain logic with assertions interpreted as compact open sets. The expressive power of the logic is, therefore, quite weak. It cannot, for example, express 'the set of even numbers' of $\mathbf{N}_\perp$, the domain of natural numbers. To make the logic more expressive some extra structure on assertion is needed. One of the possible ways to get a richer assertion language is to introduce a fixed point operator.

For each type σ, we introduce a countable set of propositional variables. Under certain condition each assertion $\varphi(x)$ induces a continuous function

$$F : \Omega(\mathcal{D}(\sigma)) \to \Omega(\mathcal{D}(\sigma)).$$

By Tarski's theorem, such a function has a least fixed point. We introduce the μ-assertion $\mu x.\varphi(x)$, to express the least fixed point.

In section 6.1 an enriched assertion language is introduced, allowing μ-assertions, standing for the least fixed points. An interpretation is given for μ-assertions, making it precise what we mean by 'the least fixed point'. Section 6.2 provides proof rules for the new assertion language. Various derived rules and properties are given for the μ−calculus. In section 6.3 some other proof rules are introduced. Soundness, completeness and expressiveness are discussed in section 6.4, where we show that the proof system is sound in general, and complete for the domain of natural nmbers.

6.1 Recursively Defined Assertions

To focus our attention on the least fixed point of assertions we work on a language of type expressions without powerdomain constructions:

$$\sigma ::= \mathbf{1} \mid \sigma \times \tau \mid \sigma \to \tau \mid \sigma + \tau \mid \sigma_\perp \mid t \mid rec.\sigma$$

where t is a type variable and σ, τ ranges over type expressions. Clearly each type σ can be interpreted as a Scott domain $\mathcal{D}(\sigma)$, as specified in the previous chapter.

The assertions for each type are built up in the following way. For each type σ we have

$$(\mathbf{t},\mathbf{f}) \qquad \mathbf{t},\mathbf{f} : \sigma$$

$$(Var) \qquad x_0^\sigma,\ x_1^\sigma,\ \cdots :\ \sigma$$

$$(\wedge - \vee) \qquad \frac{\varphi,\ \psi : \sigma}{\varphi \wedge \psi : \sigma \qquad \varphi \vee \psi : \sigma}$$

$$(\mu) \qquad \frac{\varphi(\, x^\sigma\,) :\ \sigma}{\mu x^\sigma.\ \varphi\,(\, x^\sigma\,) : \sigma}$$

When writing $\varphi(x)$, we mean that 'x is *a possible* variable in φ'. Each type is associated with a countable number of free variables and these variables are assertions. If $\varphi(\, x^\sigma\,)$ is an assertion of type σ and x^σ is a free variable, we can form $\mu x^\sigma.\ \varphi\,(\, x^\sigma\,)$, for the least fixed point, intuitively. The assertion $\mu x^\sigma.\ \varphi\,(\, x^\sigma\,)$ is called a $\mu-$assertion. For a $\mu-$assertion of the form $\mu x.\, p \vee R(x)$ with p closed, we informally call $R(x)$ a *recursive unit* and p a *contributor.*

When no confusion arises, a variable x^σ is written as x.

With respect to type constructions we have

$$(\times) \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \times \psi : \sigma \times \tau}$$

$$(\rightarrow) \qquad \frac{\varphi \in \mathcal{A}_\sigma \qquad \psi : \tau}{\varphi \rightarrow \psi : \sigma \rightarrow \tau}$$

$$(+) \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\dot{\imath}nl\ \varphi : \sigma + \tau \qquad \dot{\imath}nr\ \psi : \sigma + \tau}$$

$$(\perp) \qquad \frac{\varphi : \sigma}{\varphi_\perp : \sigma_\perp}$$

$$(rec) \qquad \frac{\varphi : \sigma[\, rec\ t.\sigma/t]}{\varphi :\ rec\ t.\sigma}$$

Write $\mathcal{M}_\sigma = \{\, \varphi \mid \varphi :\ \sigma \,\}$.

Note that in $(\rightarrow)$, φ is required to be in $\mathcal{A}_\sigma$, the assertion language introduced in Chapter 5 to express compact open sets, to make $\varphi \rightarrow \psi$ an assertion for function space. There are two reasons for this restriction. First, if free variables are allowed in place of φ above, *e.g.* $x \rightarrow \mathbf{f}$, it results in the interpretation function not being monotonic,

as is necessary for the existence of fixed points. Secondly, note the interpretation for μ assertions need not be compact open sets again. Hence if the assertion φ on the left of $\varphi \to \psi$ is allowed to be a $\mu-$assertion, it can lead to $\varphi \to \psi$ not being open (Example 6.1.3), which we want to avoid at this stage.

Example 6.1.1 Consider the recursively defined type $rec.\, t_\perp \times t_\perp$.

$$x, y : rec.\, t_\perp \times t_\perp \Longrightarrow x_\perp \times y_\perp : (rec.\, t_\perp \times t_\perp)_\perp \times (rec.\, t_\perp \times t_\perp)_\perp.$$

By (rec),

$$x_\perp \times y_\perp : rec.\, t_\perp \times t_\perp.$$

Hence

$$\mu x.\, (\mu\, y.\, x_\perp \times y_\perp) : rec.\, t_\perp \times t_\perp.$$

Example 6.1.2

$$\mu x.\, [\dot{in}l(\mathbf{t}_\perp) \vee \dot{in}r\, (\dot{in}r\, (x))]$$

is a μ assertion of type $rec\, t.\, (\mathbf{1}_\perp + t)$ which will denote the set of even numbers.

Example 6.1.3 The following assertion shows if we did not require $\varphi \in \mathcal{A}_\sigma$ in $(\to)$ when formulating the assertion we would have allowed assertions such as

$$(\, \mu x.\, [\dot{in}l(\mathbf{t}_\perp) \vee \dot{in}r\, (x)]\,) \to \mathbf{t}_\perp$$

of type $(rec\, t.\, (\mathbf{1}_\perp + t)) \to \mathbf{1}_\perp$, which would, according the the interpretation given later, represent the set

$$\{\, f : \mathbf{N}_\perp \to \mathcal{O} \mid \omega \subseteq f^{-1}(\top)\,\}.$$

This set is not open since the limit (of a chain of functions) in this set does not imply that a finite approximation of the limit is also in the set. Here $\mathbf{N}_\perp$ is the cpo got by attaching a bottom element to the set of integers $\omega =^{\text{def}} \{0, 1, \cdots, n, \cdots\}$.

To give a semantic interpretation of the assertions we introduce *environments*, which are maps from type variables to compact open sets.

Definition 6.1.1 An environment ρ is a function

$$\rho : \bigcup_\sigma \{\, x^\sigma \mid x^\sigma : \sigma\,\} \to \bigcup_\sigma \{\, K \mid K \in \mathbf{K\Omega\mathcal{D}}(\sigma)\,\}$$

such that $\forall x^\sigma.\ \rho(x^\sigma) \in \mathbf{K\Omega\mathcal{D}}(\sigma)$. Write $\mathcal{E}nv$ for the set of environments.

The semantics of assertions are given by an inductive definition. For each closed type expression σ we define an *interpretation function*

$$[\![\ ]\!]_\sigma : \mathcal{M}_\sigma \to [\mathit{Env} \to \Omega(\mathcal{D}(\sigma))]$$

in the following structured way.

For each type σ, define

$$[\![\mathbf{t}]\!]_\sigma \rho = \mathcal{D}(\sigma)$$

$$[\![\mathbf{f}]\!]_\sigma \rho = \emptyset$$

$$[\![\varphi \vee \psi]\!]_\sigma \rho = [\![\varphi]\!]_\sigma \rho \cup [\![\psi]\!]_\sigma \rho$$

$$[\![\varphi \wedge \psi]\!]_\sigma \rho = [\![\varphi]\!]_\sigma \rho \cap [\![\psi]\!]_\sigma \rho$$

$$[\![x^\sigma]\!] \rho = \rho(x^\sigma)$$

$$[\![\mu x^\sigma . \varphi(x^\sigma)]\!]_\sigma \rho = \bigcup_{i \in \omega} [\![\varphi(x^\sigma)]\!]^i_\sigma \rho$$

where

$$[\![\varphi(x^\sigma)]\!]^0_\sigma \rho = \emptyset$$

and, in general,

$$[\![\varphi(x^\sigma)]\!]^i_\sigma \rho = [\![\varphi(x^\sigma)]\!]_\sigma \rho[x^\sigma \mapsto [\![\varphi(x^\sigma)]\!]^{i-1}_\sigma \rho].$$

Here

$$\rho[x^\sigma \mapsto K](y^\tau) = \begin{cases} \rho(y^\tau) & \text{if } x^\sigma \neq y^\tau \\ K & \text{if } x^\sigma = y^\tau \end{cases}$$

where $K \in \mathbf{K}\Omega(\mathcal{D}(\sigma))$.

With respect to type expressions we define

$$[\![\varphi \times \psi]\!]_{\sigma \times \tau} \rho = \{ \langle u, v \rangle \mid u \in [\![\varphi]\!]_\sigma \rho \ \&\ v \in [\![\psi]\!]_\tau \rho \}$$

$$[\![\mathit{inl}\, \varphi]\!]_{\sigma + \tau} \rho = \{ \langle 0, u \rangle \mid u \in [\![\varphi]\!]_\sigma \rho \setminus \{ \bot_{\mathcal{D}(\sigma)} \} \} \cup \{ x \in \mathcal{D}(\sigma + \tau) \mid \bot_{\mathcal{D}(\sigma)} \in [\![\varphi]\!]_\sigma \rho \}$$

$$[\![\mathit{inr}\, \varphi]\!]_{\sigma + \tau} \rho = \{ \langle 1, u \rangle \mid u \in [\![\varphi]\!]_\tau \rho \setminus \{ \bot_{\mathcal{D}(\tau)} \} \} \cup \{ x \in \mathcal{D}(\sigma + \tau) \mid \bot_{\mathcal{D}(\tau)} \in [\![\varphi]\!]_\tau \rho \}$$

$$[\![\varphi \to \psi]\!]_{\sigma \to \tau} \rho = \{ f \in \mathcal{D}(\sigma) \to \mathcal{D}(\tau) \mid [\![\varphi]\!]_\sigma \subseteq f^{-1}([\![\psi]\!]_\tau \rho) \}$$

$$[\![(\varphi)_\perp]\!]_{(\sigma)_\perp} \rho = \{ \langle 0, u \rangle \mid u \in [\![\varphi]\!]_\sigma \rho \}$$

$$[\![\varphi]\!]_{rec.\sigma} \rho = \{ \epsilon_\sigma(u) \mid u \in [\![\varphi]\!]_{\sigma[(rec.\sigma)\backslash t]} \rho \}$$

where, as before, $\epsilon_\sigma : \mathcal{D}(\sigma[(rect.\sigma)\backslash t]) \to \mathcal{D}(rect.\sigma)$ is the isomorphism arising form the initial solution to the domain equation associated with type $rect.\sigma$.

Often the associated type is implicit from the context, so type-subscripts are omitted. We often write $[\![\ ]\!]_\rho$ for the semantic interpretation.

We show that our definition has the desired effect, i.e., $\mu x.\ \varphi(x)$ is the least fixed point of a proper function related to $\varphi(x)$.

Definition 6.1.2 Let D be a Scott domain and $\Omega(D)$ be the set of Scott open sets of D. A function $F : \Omega(D) \to \Omega(D)$ is continuous if it is monotonic and for each chain

$$A_0 \subseteq A_1 \subseteq A_2 \cdots \subseteq A_i \cdots$$

in $\Omega(D)$ we have

$$F(\bigcup_{i\in\omega} A_i) = \bigcup_{i\in\omega} F(A_i).$$

Proposition 6.1.1 For each assertion $\varphi(x)$ of type σ and an environment ρ,

$$\lambda X.\, \Phi_\rho(X) : \Omega(\mathcal{D}(\sigma)) \longrightarrow \Omega(\mathcal{D}(\sigma))$$

is a continuous function, where $\Phi_\rho(A) = [\![\varphi(x)]\!]_{\rho[x\mapsto A]}$, for any $A \in \Omega(\mathcal{D}(\sigma))$.

For convenience we write $\lambda X.[\![\varphi(x)]\!]_{\rho[x\mapsto X]}$ for such an $\lambda X.\Phi_\rho(X)$.

Proof By structured induction on the assertions of a given type σ.

Every such $\lambda X.\Phi_\rho(X)$ is monotonic because free variables are not allowed on the right of $\to$ which is the only constructor we need to check. For each chain

$$A_0 \subseteq A_1 \subseteq A_2 \cdots \subseteq A_i \cdots$$

in $\Omega(\mathcal{D}(\sigma))$, we have the following equations.

$$\lambda X.[\![\ x^\sigma\ ]\!]_{\rho[x^\sigma\mapsto X]}(\bigcup_{i\in\omega} A_i) = \bigcup_{i\in\omega} A_i = \bigcup_{i\in\omega} \lambda X.[\![\ x^\sigma\ ]\!]_{\rho[x^\sigma\mapsto X]}(A_i)$$

$$\lambda X.[\![p]\!]_{\rho[x^\sigma\mapsto X]}(\bigcup_{i\in\omega} A_i) = \lambda X.[\![p]\!]_\rho = \bigcup_{i\in\omega} \lambda X.[\![\ p\ ]\!]_{\rho[x^\sigma\mapsto X]}(A_i)$$

where p is an assertion in which x^σ does not occur free.

$$\begin{aligned}\lambda X.[\![\ \varphi \vee \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) &= \lambda X.[\![\ \varphi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) \cup \lambda X.[\![\ \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) \\ &= \bigcup_{i\in\omega} \lambda X.[\![\varphi]\!]_{\rho[x^\sigma \mapsto X]}(A_i) \cup \bigcup_{i\in\omega} \lambda X.[\![\psi]\!]_{\rho[x^\sigma \mapsto X]}(A_i) \\ &= \bigcup_{i\in\omega} \lambda X.[\![\ \varphi \vee \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(A_i)\end{aligned}$$

$$\begin{aligned}\lambda X.[\![\ \varphi \wedge \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) &= \lambda X.[\![\ \varphi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) \cap \lambda X.[\![\ \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) \\ &= (\bigcup_{i\in\omega} \lambda X.[\![\varphi]\!]_{\rho[x^\sigma \mapsto X]}(A_i)) \cap (\bigcup_{i\in\omega} \lambda X.[\![\psi]\!]_{\rho[x^\sigma \mapsto X]}(A_i)) \\ &= \bigcup_{i\in\omega} \lambda X.[\![\ \varphi \wedge \psi\ ]\!]_{\rho[x^\sigma \mapsto X]}(A_i)\end{aligned}$$

$$\begin{aligned}\lambda X.[\![\ \mu t.\ \varphi(t,\ x)\ ]\!]_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) &= \bigcup_{j\in\omega} \lambda X.[\![\ \varphi\ ]\!]^j_{\rho[x^\sigma \mapsto X]}(\bigcup_{i\in\omega} A_i) \\ &= \bigcup_{j\in\omega}\bigcup_{i\in\omega} \lambda X.[\![\ \varphi\ ]\!]^j_{\rho[x^\sigma \mapsto X]}(A_i) \\ &= \bigcup_{i\in\omega}\bigcup_{j\in\omega} \lambda X.[\![\ \varphi\ ]\!]^j_{\rho[x^\sigma \mapsto X]}(A_i) \\ &= \bigcup_{i\in\omega} \lambda X.[\![\ \mu t.\ \varphi(t,\ x)\ ]\!]_{\rho[x^\sigma \mapsto X]}(A_i)\end{aligned}$$

∎

As a direct consequence of Proposition 6.1.1 we have

Proposition 6.1.2 For any assertion φ of type σ and any environment ρ,

$$[\![\ \mu t.\ \varphi\ ]\!]_\rho$$

is the least fixed point of

$$\lambda X.[\![\ \varphi\ ]\!]_{\rho[x^\sigma \mapsto X]}.$$

6.2 Results of a Mu-Calculus

The proof system for the fixed point calculus consists of the positive parts of relevant axioms and rules for lifting, sum, product, and function space introduced in Chapter 5. Although the formulation of the axioms and rules is still the same, they make broader sense as assertions can now have free variables in them and they can be μ assertions. We have, as an instance of the axiom

$$\dot{inl}\,(\,\varphi \vee \psi\,) = (\,\dot{inl}\varphi\,) \vee (\,\dot{inl}\psi\,)$$

an derived formula

$$\dot{inl}\,(\,x \vee y\,) = (\,\dot{inl}x\,) \vee (\,\dot{inl}y\,).$$

We only formulate $\leq$, the positive formulae for the μ-calculus.

The following axioms and rules are new, introduced for μ assertions.

Notation: In this section we write p or q for a closed assertion and $P(\,x\,)$, $Q\,(\,y\,)$ for assertions with possible variables as indicated in the bracket.

$$(\ \mu - \text{axiom}\ ) \qquad Q(\,\mu x.\ Q(\,x\,)\,) \leq \mu x.\ Q(\,x\,)$$

$$(\ \mu - \text{rule}\ ) \qquad \frac{Q(\,\varphi\,) \leq \varphi}{\mu x.\ Q(\,x\,) \leq \varphi}$$

The following rule, which gives us monotonicity, can be easily derived from the rest of the rules.

$$(\ \text{mono}\ ) \qquad \frac{\psi \leq \varphi}{Q(\,\psi\,) \leq Q(\,\varphi\,)}$$

Here $Q(x)$ is required to be an assertion when x is an assertion variable with the same type as ψ. This restriction rules out invalid instances such as

$$\frac{\psi \leq \varphi}{\psi \to y \leq \varphi \to y}.$$

For convenience when a formula $\varphi \leq \psi$ is derivable from the proof system consisting of rules for lifting, sum, product, function space, and the μ-axiom and μ-rule, we write $\vdash \varphi \leq \psi$. The explicit judgement $\vdash$ is missed out most of the time. This $\vdash$ notation is somewhat provisional because to get completeness we have to add some other rules, as we will see in section 4.

We would like to have a sound and complete proof system for the μ-calculus. The soundness part is easy to check. The proof for completeness usually divides into two steps: first one shows that each assertion is provably equivalent to some normal form, then one proves that the proof system is complete for assertions in the normal form. In the following we give theorems (derived proof rules) derived from the proof system to make the task of deducing the normal forms feasible.

Clearly there is no problem renaming bound variables. For example, we have

$$\vdash \mu x.\, P(x) = \mu y.\, P(y),$$

since $P(\mu y.\, P(y)) = \mu y.\, P(y)$, and one can apply the $\mu-$rule to get

$$\mu x.\, P(x) \leq \mu y.\, P(y).$$

Proposition 6.2.1 below shows that $\mu x.\, Q(\,x\,)$ is really a fixed point of Q.

Proposition 6.2.1

$$Q(\,\mu x.\, Q(\,x\,)\,) = \mu x.\, Q(\,x\,).$$

Proof

$$\begin{aligned}
Q(\,\mu x.\, Q(\,x\,)\,) &\leq \mu x.\, Q(\,x\,), && \mu - \text{axiom}\\
Q(\,Q(\,\mu x.\, Q(\,x\,)\,)\,) &\leq Q(\,\mu x.\, Q(\,x\,)\,), && \text{monotonicity}\\
\mu x.\, Q(\,x\,) &\leq Q(\,\mu x.\, Q(\,x\,)\,), && \mu - \text{rule}\\
\mu x.\, Q(\,x\,) &= Q(\,\mu x.\, Q(\,x\,)\,). && \mu - \text{axiom}
\end{aligned}$$

∎

Proposition 6.2.2

$$\mu x.\, [\,P \wedge Q(\,x\,)\,] \leq \mu x.\, Q(\,x\,),$$

where x does not occur free in P.

Proof

$$\begin{aligned}
Q(\,\mu x.\, Q(\,x\,)\,) &\leq \mu x.\, Q(\,x\,), && \mu - \text{axiom}\\
p \wedge Q(\,\mu x.\, Q(\,x\,)\,) &\leq \mu x.\, Q(\,x\,), && \text{logical rule}\\
\mu x.\, [\,p \wedge Q(\,x\,)\,] &\leq \mu x.\, Q(\,x\,). && \mu - \text{rule}
\end{aligned}$$

∎

There is the following more general derived rule which includes Proposition 6.2.2 as a special case:

$$\frac{P(x) \leq Q(x)}{\mu y.\ P(y) \leq \mu y.\ Q(y)}.$$

Instantiate x by $\mu y.\ Q(y)$ in the assumption we get

$$P(\mu y.\ Q(y)) \leq Q(\mu y.\ Q(y)).$$

But

$$Q(\mu y.\ Q(y)) \leq \mu y.\ Q(y)$$

by the μ−axiom; thus

$$\mu y.\ P(y) \leq \mu y.\ Q(y)$$

by the μ−rule.

Proposition 6.2.3 We have $\mu x.\ Q(x) \leq p$ implies $\mu x.\ Q(x) \leq \mu x.\ (p \wedge Q(x))$. In fact

$$\frac{\mu x.\ Q(x) \leq p}{\mu x.\ Q(x) \leq \mu x.\ (p \wedge Q(x))}$$

is a derived rule.

Proof

$\mu x.\ (p \wedge Q(x))$	$\leq \mu x.\ Q(x)$,	Proposition 6.2.2
$Q(\mu x.\ p \wedge Q(x))$	$\leq \mu x.\ Q(x)$,	monotonicity and $\mu -$ axiom
	$\leq p$,	assumption
$Q(\mu x.\ p \wedge Q(x))$	$\leq p \wedge Q(\mu x.\ p \wedge Q(x))$,	logical axiom
$Q(\mu x.\ p \wedge Q(x))$	$\leq \mu x.\ p \wedge Q(x)$,	$\mu -$ axiom
$\mu x.\ Q(x)$	$\leq \mu x.\ (p \wedge Q(x))$.	$\mu -$ rule

∎

Proposition 6.2.4 We have $Q(\mu x.\ p \wedge Q(x)) = p$ implies $\mu x.\ Q(x) = p$. In other words,

$$\frac{Q(\mu x.\ p \wedge Q(x)) = p}{\mu x.\ Q(x) = p}$$

is a derived rule.

Proof

$$
\begin{aligned}
Q(\mu x.\ p \wedge Q(x)) &= p, && \text{assumption}\\
Q(\mu x.\ p \wedge Q(x)) &= p \wedge Q(\mu x.\ p \wedge Q(x)), && \text{logical rule}\\
&= \mu x.\ (p \wedge Q(x)), && \text{Proposition 6.2.1}\\
\mu x.\ Q(x) &\leq \mu x.\ p \wedge Q(x), && \mu - \text{rule}\\
\mu x.\ Q(x) &= \mu x.\ p \wedge Q(x), && \text{Proposition 6.2.2}\\
Q(\mu x.\ Q(x)) &= Q(\mu x.\ p \wedge Q(x)), && \text{monotonicity}\\
\mu x.\ Q(x) &= p. && \text{by assumption}
\end{aligned}
$$

∎

Note the same proof goes through by changing $=$ to $\leq$, i.e., we have the derived rule

$$\frac{Q(\mu x.\ p \wedge Q(x)) \leq p}{\mu x.\ Q(x) \leq p}.$$

Call Q *distributive over* $\vee$ if

$$\vdash Q(x \vee y) = Q(x) \vee Q(y);$$

Call Q *distributive over* $\wedge$ if

$$\vdash Q(x \wedge y) = Q(x) \wedge Q(y).$$

Q is *distributive* if Q is distributive over both $\vee$ and $\wedge$ and $Q(\mathbf{f}) = \mathbf{f}$. Here x and y are assertion variables.

If $Q(x)$ is distributive over $\vee$ then so is $Q^n(x)$ for any natural number n, where $Q^i(x) = Q^{i-1}(Q(x))$. This can be easily checked by mathematical induction. Similarly if Q is distributive over $\wedge$ then so is Q^n for any natural number n; if Q is distributive then for any natural number n, Q^n is distributive. Note if R distributes over $\vee$ then so does $p \vee R$ and $p \wedge R$ for any closed assertion p; If R distributes over $\wedge$ then so does $p \vee R$ and $p \wedge R$ for any closed assertion p. Evidently, then, if R is distributive then so is $p \vee R$ and $p \vee R$ for any closed assertion p.

Assertions like $Q(x)$ are not necessarily always distributive over $\vee$; consider $p \rightarrow x$, for example. There is actually a large class of assertions of the form $\mu y.\ (\ P(y) \wedge x\ )$ which do not distribute over $\vee$. However we have

Lemma For any type σ, $Q(x) : \sigma$ always distributes over $\wedge$ provided $Q(x)$ does not contain any μ−assertion. If x is the only free variable in $P(x)$, then there exist p, $R(x)$, p', and $R'(x)$ such that

$$\vdash P(x) = p \vee R(x),$$

$$\vdash P(x) = p' \wedge R'(x),$$

where p and p' are closed, $R(\mathbf{f}) = \mathbf{f}$, and $R'(\mathbf{t}) = \mathbf{t}$. Again, $P(x)$ must not contain any μ−assertion.

Proof By inspecting the proof rules one can see that all the assertion constructors *inl*, *inr*, $(\quad)_\perp$, $\rightarrow$, $\times$ distribute over $\wedge$. Moreover, $\vee$, $\wedge$ preserve such distributivity. Hence any $Q(x)$ distributes over $\wedge$, by an easy structured induction.

To show $\vdash P(x) = p \vee R(x)$, we first use logical axioms to put $P(x)$ into a disjunctive normal form. Let p be the part of disjunction consisting of all the closed assertions. The rest are disjuncts in which x appear as free variable. Take this part as $R(x)$, we have $R(\mathbf{f}) = \mathbf{f}$. When dealing with the function space construction, we apply the axiom $\mathbf{f} \rightarrow x = \mathbf{t}$ whenever it is possible, so as to avoid x to be considered as free in $\mathbf{f} \rightarrow x$.

The proof that

$$\vdash P(x) = p' \wedge R'(x)$$

is similar, but in this case we use the conjunctive normal form. ∎

It is a bore to always indicate which rules are applied and in the remaining chapter we shall work more informally.

Proposition 6.2.5 says if the contributor part of a μ−assertion breaks into two parts then the μ− assertion is equivalent to two μ−assertions with the same recursive unit but different contributor parts of the original assertion as their contributors.

Proposition 6.2.5 If R distributes over $\vee$, then

$$\mu x.\,(\,p_1 \vee p_2 \vee R(\,x\,)\,) = [\,\mu x.\,p_1 \vee R(\,x\,)\,] \vee [\,\mu y.\,p_2 \vee R(\,y\,)\,].$$

Proof Clearly

$$(\,\mu x.\,p_1 \vee R(\,x\,)\,) \vee (\,\mu y.\,p_2 \vee R(\,x\,)\,) \leq \mu x.\,(\,p_1 \vee p_2 \vee R(\,x\,)\,).$$

On the other hand,

$$p_1 \vee p_2 \vee R(\mu x.\, p_1 \vee R(x) \vee \mu y.\, p_2 \vee R(y))$$
$$= [p_1 \vee R(\mu x.\, p_1 \vee R(x))] \vee [(p_2 \vee R(\mu y.\, p_2 \vee R(y)))]$$

by assumption. Hence

$$p_1 \vee p_2 \vee R(\mu x.\, p_1 \vee R(x) \vee \mu y.\, p_2 \vee R(y)) = \mu x.\, p_1 \vee R(x) \vee \mu y.\, p_2 \vee R(y),$$

which implies, by $\mu-$ rule,

$$\mu z.\, p_1 \vee p_2 \vee R(z) \leq \mu x.\, p_1 \vee R(x) \vee \mu y.\, p_2 \vee R(y).$$

∎

Proposition 6.2.6 means under certain assumption we can expand the contributor of a $\mu-$assertion by applying the recursive unit a number of times (n times, say) and get a new assertion which is equivalent to the original one provided we use a new recursive unit got by applying the old one to itself $n+1$ times.

Proposition 6.2.6 Suppose Q is distributive over $\vee$. Then $\forall n > 0$,

$$\vdash \mu x.\, \left[(\bigvee_{i=0}^{n} Q^i(p)) \vee Q^{n+1}(x)\right] = \mu x.\, p \vee Q(x).$$

Proof We prove for the case $n = 2$ (The general case can be done similarly use mathematical induction). We have

$$\begin{aligned} \mu x.\, p \vee Q(x) &= p \vee Q(p \vee Q(\mu x.\, p \vee Q(x))), \\ &= p \vee Q(p) \vee Q^2(\mu x.\, p \vee Q(x)). \end{aligned}$$

Therefore

$$\mu y.\; p \vee Q(p) \vee Q^2(y) \leq \mu x.\, p \vee Q(x).$$

On the other hand,

$$\begin{aligned} & p \vee Q(p) \vee Q^2[p \vee Q(\mu y.\, p \vee Q(p) \vee Q^2(y))] \\ &= p \vee Q(p) \vee Q^2(p) \vee Q^3[\mu y.\, p \vee Q(p) \vee Q^2(y)] \\ &= p \vee Q[p \vee Q(p) \vee Q^2(\mu y.\, p \vee Q(p) \vee Q^2(y))] \\ &= p \vee Q[\mu y.\, p \vee Q(p) \vee Q^2(y)]. \end{aligned}$$

Hence

$$\mu x.\ p \vee Q(p) \vee Q^2(x) \leq p \vee Q(\mu y.\ p \vee Q(p) \vee Q^2(y)).$$

Now we have

$$\begin{aligned} p \vee Q(\mu x.\ p \vee Q(p) \vee Q^2(x)) &\leq p \vee Q(p \vee Q(\mu y.\ p \vee Q(p) \vee Q^2(y))) \\ &\leq p \vee Q(p) \vee Q^2(\mu y.\ p \vee Q(p) \vee Q^2(y)) \\ &\leq \mu y.\ p \vee Q(p) \vee Q^2(y). \end{aligned}$$

Therefore

$$\mu x.\ p \vee Q(x) \leq \mu y.\ p \vee Q(p) \vee Q^2(y).$$

∎

The following proposition tells us that under certain conditions, the effect of applying the recursive unit to the whole μ–assertion is the same as only applying it to the contributor.

Proposition 6.2.7

$$\vdash Q(\mu x.\ p \vee Q^n(x)) = \mu x.\ Q(p) \vee Q^n(x)$$

for any $n > 0$, provided that Q is distributive over $\vee$.

Proof We have

$$\begin{aligned} p \vee Q^n(\mu x.\ p \vee Q^n(x)) &= \mu x.\ p \vee Q^n(x), \\ Q(p \vee Q^n(\mu x.\ p \vee Q^n(x))) &= Q(\mu x.\ p \vee Q^n(x)), \\ Q(p) \vee Q^n[Q(\mu x.\ p \vee Q^n(x))] &\leq Q(\mu x.\ p \vee Q^n(x)), \\ \mu x.\ Q(p) \vee Q^n(x) &\leq Q[\mu x.\ p \vee Q^n(x)]. \end{aligned}$$

On the other hand,

$$\begin{aligned} p \vee Q^n[p \vee Q^{n-1}(\mu x.\ Q(p) \vee Q^n(x))] &= p \vee Q^n(p) \vee Q^{2n-1}(\mu x.\ Q(p) \vee Q^n(x)), \\ &= p \vee Q^{n-1}[Q(p) \vee Q^n(\mu x.\ Q(p) \vee Q^n(x))], \\ &= p \vee Q^{n-1}(\mu x.\ Q(p) \vee Q^n(x)), \end{aligned}$$

so

$$\mu x.\ p \vee Q^n(x) \leq p \vee Q^{n-1}(\mu x.\ Q(p) \vee Q^n(x)).$$

Therefore

$$\begin{aligned} Q(\,\mu x.\; p \vee Q^n(\,x\,)\,) &\leq Q(\,p\,) \vee Q^n(\,\mu x.\; Q(\,p\,) \vee Q^n(\,x\,)\,), \\ &\leq \mu x.\; Q(\,p\,) \vee Q^n(\,x\,). \end{aligned}$$

∎

By repeated application of Proposition 6.2.7 we have, $\forall m \geq 0$ and $n > 0$,

$$Q^m(\,\mu x.\; p \vee Q^n(\,x\,)\,) = \mu x.\; Q^m(\,p\,) \vee Q^n(\,x\,)$$

provided that Q distributes over $\vee$.

For certain forms of recursive unit, if it has two parts then the μ−assertion is equivalent to a disjunction of two μ−assertions each of which only has one part of the recursive unit of the original assertion.

Proposition 6.2.8 If Q distributes over $\vee$, then

$$\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,) = \bigvee_{i=0}^{m-1} [\mu x.\, Q^{i \cdot n}(\,p\,) \vee Q^m(\,x\,)] \;\vee\; \bigvee_{j=0}^{n-1} [\mu y.\, Q^{j \cdot m}(\,p\,) \vee Q^n(\,y\,)].$$

Proof For any i with $0 \leq i \leq m-1$,

$$\begin{aligned} \mu x.\; Q^{i \cdot n}(\,p\,) \vee Q^m(\,x\,) &= Q^{i \cdot n}[\mu x.\; p \vee Q^m(\,x\,)] \\ &\leq Q^{(i-1) \cdot n}[p \vee Q^n(\,\mu x.\; p \vee Q^m(\,x\,)\,)] \\ &\leq Q^{(i-1) \cdot n}\,[p \vee Q^n(\,\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)\,) \\ &\qquad \vee Q^m(\,\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)\,)] \\ &= Q^{(i-1) \cdot n}[\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)] \\ &\leq Q^{(i-2) \cdot n} Q^n[\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)] \\ &\leq Q^{(i-2) \cdot n}\,[p \vee Q^m(\,\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)\,) \\ &\qquad \vee Q^n(\,\mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,)\,)] \\ &\;\;\vdots \\ &\leq \mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,). \end{aligned}$$

Therefore,

$$\bigvee_{i=0}^{m-1} \left[\mu x.\; Q^{i \cdot n}(\,p\,) \vee Q^m(\,x\,)\right] \leq \mu x.\; p \vee Q^m(\,x\,) \vee Q^n(\,x\,).$$

Similarly,

$$\bigvee_{j=0}^{n-1} \left[\mu y.\ Q^{j\cdot m}(p) \vee Q^n(y)\right] \leq \mu x.\ p \vee Q^m(x) \vee Q^n(x).$$

On the other hand, we need to show

$$p \vee Q^m(RHS) \vee Q^n(RHS) \leq RHS,$$

where RHS is an abbreviation for the assertion

$$\left[\mu x.\ \bigvee_{i=0}^{m-1} Q^{i\cdot n}(p) \vee Q^m(x)\right] \vee \left[\mu y.\ \bigvee_{j=0}^{n-1} Q^{j\cdot m}(p) \vee Q^n(y)\right]$$

which is, by Proposition 6.2.5, equivalent to the right hand side of $=$ in the original formula we want to prove.

We show $Q^m(RHS) \leq RHS$ in the following. The proof for $Q^n(RHS) \leq RHS$ is similar.

$$\begin{aligned}
&Q^m\left[\bigvee_{i=0}^{m-1}[\mu x.\ Q^{i\cdot n}(p) \vee Q^m(x)]\right] \vee \left[\bigvee_{j=0}^{n-1}[\mu y.\ Q^{j\cdot m}(p) \vee Q^n(y)]\right] \\
&\quad = \bigvee_{i=0}^{m-1} Q^m[\mu x.\ Q^{i\cdot n}(p) \vee Q^m(x)] \vee \bigvee_{j=0}^{n-1} Q^m[\mu y.\ Q_{i\cdot n}(p) \vee Q^m(y)] \\
&\quad \leq \bigvee_{i=0}^{m-1} \left[Q^{i\cdot n}(p) \vee Q^m[\mu x.\ Q^{i\cdot n}(p) \vee Q^m(x)]\right] \\
&\qquad \vee \bigvee_{j=1}^{n-1} \left[\mu y.\ (Q^{j\cdot m}(p) \vee Q^n(y))\right] \vee \mu y.\ [Q^{n\cdot m}(p) \vee Q^n(y)] \\
&\quad \leq \bigvee_{i=0}^{m-1} \left[\mu x.\ Q^{i\cdot n}(p) \vee Q^m(x)\right] \\
&\qquad \vee \bigvee_{j=0}^{n-1} \left[\mu y.\ Q^{j\cdot m}(p) \vee Q^n(y)\right] \vee \mu y.\ [Q^{n\cdot m}(p) \vee Q^n(y)].
\end{aligned}$$

However,

$$\begin{aligned}
Q^{m\cdot n}(p) \vee Q^n(\mu y.\, p \vee Q^n(y)) &\leq Q^{m\cdot n}(p) \vee p \vee Q^n(\mu y.\, p \vee Q^n(y)) \\
&\leq Q^{m\cdot n}(p) \vee \mu y.\, p \vee Q^n(y) \\
&\leq \mu y.\, p \vee Q^n(y).
\end{aligned}$$

Therefore,

$$\mu y.\, [Q^{m\cdot n}(p) \vee Q^n(y)] \leq \mu y.\, p \vee Q^n(y).$$

Hence

$$Q^m(RHS) \leq RHS.$$

Now applying $\mu-$ rule we have

$$\mu x.\, p \vee Q^m(\,x\,) \vee Q^n(\,x\,) \leq RHS.$$

∎

Generalising Proposition 6.2.8, we have

$$\begin{aligned}
\mu x.\, p \vee Q^{n_1}(\,x\,) \quad & \vee Q^{n_2}(x) \cdots \vee Q^{n_k}(\,x\,) \\
= \quad & \mu x.\, \left[\bigvee_{i=0}^{n_1-1} \bigvee_{j=0}^{k} Q^{i\cdot n_j}(\,p\,)\right] \vee Q^{n_1}(\,x\,) \;\vee \\
& \mu x.\, \left[\bigvee_{i=0}^{n_2-1} \bigvee_{j=0}^{k} Q^{i\cdot n_j}(\,p\,)\right] \vee Q^{n_2}(\,x\,) \;\vee \\
& \qquad\qquad \vdots \\
& \mu x.\, \left[\bigvee_{i=0}^{n_k-1} \bigvee_{j=0}^{k} Q^{i\cdot n_j}(\,p\,)\right] \vee Q^{n_k}(\,x\,).
\end{aligned}$$

We remark that sometimes it is not necessary to work on assertions of a particular form like $\mu x.\, p \vee P(x)$; We might have worked on $\mu x.\, Q(x)$ directly as well. However no generality is lost in this way because we can always, as a special case, let p be $\mathbf{f}$. This special case is worth noticing, for example, from Proposition 6.2.6, 6.2.7, and 6.2.8 we obtain:

$$\mu x.\, \left[(\bigvee_{i=0}^{n} Q^i(\,\mathbf{f}\,)) \vee Q^{n+1}(\,x\,)\right] = \mu x.\, Q(\,x\,),$$

$$Q(\,\mu x.\, Q^n(\,x\,)\,) = \mu x.\, Q(\,\mathbf{f}\,) \vee Q^n(\,x\,),$$

and

$$\mu x.\, Q^m(\,x\,) \vee Q^n(\,x\,) = \bigvee_{i=0}^{m-1} [\mu x.\, Q^{i\cdot n}(\,\mathbf{f}\,) \vee Q^m(\,x\,)] \;\vee\; \bigvee_{j=0}^{n-1} [\mu y.\, Q^{j\cdot m}(\,\mathbf{f}\,) \vee Q^n(\,y\,)],$$

where the same assumption is required.

The following derived formula is sometimes very useful. It allows us to reduce certain forms of nested $\mu-$recursion into a single recursion.

Proposition 6.2.9

$$\vdash \mu x.\,(\,\mu y.\,P(\,x,y\,)\,) = \mu z.\,P(\,z,z\,).$$

Note for $\mu x.\,(\,\mu y.\,P(\,x,y\,)\,)$ to be well-formed the variable x, y must have the same type.

Proof

$$\begin{aligned} P(\,\mu z.P(\,z,z\,),\ \mu z.\,P(\,z,z\,)\,) &= \mu z.\,P(\,z,z\,), \\ \mu x.\,P(\,x,\ \mu z.P(\,z,z\,)\,) &\leq \mu z.\,P(\,z,z\,), \\ \mu y.\,(\,\mu x.\,P(\,x,y\,)\,) &\leq \mu z.\,P(\,z,z\,). \end{aligned}$$

Also,

$$\mu x.\,(\,\mu y.\,P(\,x,y\,)\,) \leq \mu z.\,P(\,z,z\,).$$

On the other hand,

$$\begin{aligned} \mu t.\,P(\,\mu x.\,[\,\mu y.P(\,x,y\,)\,],\ t\,) &= \mu x.\,(\,\mu y.P(\,x,\,y\,)\,), \\ P\,(\,\mu x.\,[\,\mu y.P(\,x,\,y\,)\,],\ \mu t.\,P(\,\mu x.\,[\,\mu y.P(\,x,\,y\,)\,],\,t\,)\,) &= \mu t.P(\,\mu x.\,[\,\mu y.P(\,x,\,y\,)\,],\ t\,), \\ P(\,\mu x.\,[\,\mu y.\,P(\,x,\,y\,)\,],\ \mu x.\,[\,\mu y.\,P(\,x,\,y\,)\,]\,) &= \mu x.\,[\,\mu y.P(\,x,\,y\,)\,]. \end{aligned}$$

Hence

$$\mu z.\,P(\,z,\,z\,) \leq \mu x.\,[\,\mu y.\,P(\,x,\,y\,)\,].$$

∎

In general we have

$$\mu x_1.\,(\,\mu x_2.\,(\ \cdots\ \mu x_n.\,P(\,x_1,x_2,\cdots,x_n\,)\,)\,) = \mu x.\,P(\,x,\ x,\ \cdots,x\,).$$

As a particular instance of the above formula, we have

$$\mu x.\,\mu y.\,(Q(y) \vee P(x)\,) = \mu z.\,Q(z) \vee P(z).$$

Proposition 6.2.10 For $m > 0$, $n > 0$,

$$\mu x.\,(\,P^m(x) \vee \mu y.\,p \vee P^n(y)\,) = \mu x.\,p \vee P^m(x) \vee P^n(x)$$

provided that P distributes over $\vee$.

Proof It is easy to see that

$$P^m(\,\mu x.\, p \vee P^m(x) \vee P^n(x)\,) \vee \mu y.\, p \vee P^n(y)$$

$$\leq p \vee P^m(\,\mu x.\, p \vee P^m(x) \vee P^n(x)\,) \vee P^n(\,\mu x.\, p \vee P^m(x) \vee P^n(x)\,) \vee \mu y.\, p \vee P^n(y)$$

$$\leq \mu x.\, p \vee P^m(x) \vee P^n(x).$$

By μ−rule, we have

$$\mu x.\,(\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,) \leq \mu x.\, p \vee P^m(x) \vee P^n(x).$$

On the other hand, by Proposition 6.2.7,

$$p \vee P^m(\,\mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,]\,) \vee P^n(\,\mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,]\,)$$

$$\leq p \vee \mu x.\,[\,P^m(x) \vee P^m(\mu y.\, p \vee P^n(y)\,)\,] \;\vee\; \mu x.\,[\,P^m(x) \vee P^n(\mu y.\, p \vee P^n(y)\,)\,]$$

$$\leq p \vee \mu x.\,[\,P^m(x) \vee P^m(\mu y.\, p \vee P^n(y)\,)\,] \;\vee\; \mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,]$$

$$\leq \mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,].$$

The last inequality used the fact that

$$\mu x.\,[\,P^m(x) \vee P^m(\mu y.\, p \vee P^n(y)\,)\,] \leq \mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,],$$

which follows from the μ−rule, noticing that

$$P^m[\mu x.\,(\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,)\,] \;\vee\; P^m(\mu y.\, p \vee P^n(y)\,)$$

$$\leq P^m[\mu x.\,(\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,)\,] \vee \mu y.\, p \vee P^n(y) \;\vee\; P^m(\mu y.\, p \vee P^n(y)\,)$$

$$\leq \mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,] \;\vee\; P^m(\mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,]) \vee \mu y.\, p \vee P^n(y)$$

$$\leq \mu x.\,[\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,].$$

Now applying the μ−rule we get

$$\mu x.\, p \vee P^m(x) \vee P^n(x) \leq \mu x.\,(\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,).$$

∎

From Proposition 6.2.10 it follows that

$$\mu x.\,(\,P^m(x) \vee \mu y.\, p \vee P^n(y)\,) = \mu x.\,(\,P^n(x) \vee \mu y.\, p \vee P^m(y)\,)$$

provided that P distributes over $\vee$, where $m > 0$, $n > 0$.

6.3 A Special Rule

Let us see what we can prove from the μ−calculus of a particular type, the type of natural numbers $\mathbf{N}_\perp$ specified by $rec\, t.\, \mathbf{1}_\perp + t$. As remarked at the beginning, intuitively $\dot{inl}\mathbf{t}_\perp : \mathbf{N}_\perp$ denotes 'zero', $\dot{inr}$ the successor function. By Proposition 6.2.7, we can derive

$$\dot{inr}\ (\mu x.\ [\,\dot{inl}(\mathbf{t}_\perp) \vee \dot{inr}\ (\,\dot{inr}\,(\,x\,)\,)\,]\,) = \mu x.\ [\,\dot{inr}\ (\,\dot{inl}(\mathbf{t}_\perp)\,) \vee \dot{inr}\ (\,\dot{inr}\,(\,x\,)\,)\,],$$

which means if we apply the successor to the whole set of even numbers we get the set of odd numbers, which is expected. As special cases of the propositions given in the previous section we know that many other facts about natural numbers are derivable from the μ−calculus, a few of which are listed as follows:

- $\vdash \mu x.\, \dot{inr}\, x = \mathbf{f}$
- $\vdash \mu x.\ (\dot{inl}(\mathbf{t}_\perp) \vee \dot{inr}^2 x\,) = \dot{inl}(\mathbf{t}_\perp) \vee \mu x.\ \dot{inr}^2(\dot{inl}(\mathbf{t}_\perp) \vee \dot{inr}^2 x\,)$
- $\vdash \mu x.\ (\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2 x\,) = (\,\mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2 x\,)\ \vee\ (\,\mu x.\ \dot{inr}\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2 x\,)$
- $\vdash \mu x.\ (\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2 x\,) = \mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}\, x$

Is it the case that any fact expressable in the μ−calculus are provable, i.e., is the μ−calculus complete? Consider the assertion

$$(\,\mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\, x\,) \wedge (\,\mu x.\ \dot{inr}\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\, x\,).$$

It stands for the intersection of even numbers and odd numbers and hence should give us the empty set, that is, one expects to be able to prove from the μ−calculus that

$$(\,\mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\, x\,) \wedge (\,\mu x.\ \dot{inr}\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\, x\,) = \mathbf{f}.$$

I do not have a proof that one cannot derive the above formula from the μ−calculus with rules given so far. However certainly I see no simple way to derive it and I strongly doubt that it is derivable. The μ−axiom and μ−rule allow us to reason about the least fixed points expressed in the form $\mu x.P(x)$. But least fixed points can be expressed in many other forms and one should not expect that the μ−axiom and μ−rule alone also provide us with power to reason about *equivalences of fixed points*. We introduce the following rule for each type, to deal with the interaction of conjunction and μ construction.

$$(\wedge - \mu) \qquad \frac{p \wedge (\mu x.\, q \vee R(x)) \leq R(p) \qquad R \text{ distributive}}{p \wedge (\mu x.\, q \vee R(x)) \leq \mu x.\, p \wedge (q \vee R(x))}$$

We check the soundness of this rule by mathematical induction, to show that for all n, for all environment ρ,

$$\bigcup_{0 \leq i \leq n} [\![p \wedge R^i(q)]\!]_\rho \subseteq [\![\mu x.\, p \wedge (q \vee R(x))]\!]_\rho$$

provided that

$$\forall \rho.\, [\![p \wedge \mu x.\, q \vee R(x)]\!]_\rho \subseteq [\![R(p)]\!]_\rho.$$

Here, $R^0(x)$ is an abbreviation for x. This proof relies on the soundness of the rest of the $\mu-$calculus. The base case $n = 1$ is trivial. Induction step. First note that, by assumption,

$$\bigcup_{1 \leq i \leq n+1} [\![p \wedge R^i(q)]\!]_\rho \subseteq [\![p \wedge \mu x.\, q \vee R(x)]\!]_\rho \subseteq [\![R(p)]\!]_\rho.$$

Therefore

$$\bigcup_{1 \leq i \leq n+1} [\![p \wedge R^i(q)]\!]_\rho = [\![R(p)]\!]_\rho \cap \bigcup_{1 \leq i \leq n+1} [\![p \wedge R^i(q)]\!]_\rho.$$

Now

$$\begin{aligned}
\cup_{0 \leq i \leq n+1} [\![p \wedge R^i(q)]\!]_\rho &= [\![p \wedge q]\!]_\rho \cup ([\![p]\!]_\rho \cap \cup_{1 \leq i \leq n+1} [\![R^i(q)]\!]_\rho)) \\
&= [\![p \wedge q]\!]_\rho \cup ([\![p]\!]_\rho \cap [\![R(p)]\!]_\rho \cap \cup_{1 \leq i \leq n+1} [\![R^i(q)]\!]_\rho)) \\
&= [\![p \wedge q]\!]_\rho \cup ([\![p]\!]_\rho \cap [\![R(\vee_{0 \leq i \leq n} p \wedge R^i(q))]\!]_\rho) \\
&\subseteq [\![p \wedge q]\!]_\rho \cup ([\![p]\!]_\rho \cap [\![R(\mu x.\, p \wedge (q \vee R(x)))]\!]_\rho) \\
&= [\![\mu x.\, p \wedge (q \vee R(x))]\!]_\rho.
\end{aligned}$$

Note in the second last step we used the induction hypothesis,

$$\bigcup_{0 \leq i \leq n} [\![p \wedge R^i(q)]\!]_\rho \subseteq [\![\mu x.\, p \wedge (q \vee R(x))]\!]_\rho,$$

and the monotonicity, distributivity of R. Thus we have proved

Proposition 6.3.1 The rule $(\wedge - \mu)$ is sound.

The following rule is necessary for us to reason about $\mu-$assertions in sum.

$$(\neg + \mu) \qquad \frac{\mathbf{t} \not\leq P(\mathbf{f}) \qquad \mathbf{t} \not\leq Q(\mathbf{f})}{\dot{inl}\,(\mu x.\, P(x)) \wedge \dot{inr}\,(\mu y.\, Q(y)) \leq \mathbf{f}}$$

Note that to use the proof system for $\not\leq$ in Chapter 5, assertions must be in $\mathcal{A}_\sigma$ with σ the type concerned. Therefore when we write $\mathbf{t} \not\leq P(\mathbf{f})$ and $\mathbf{t} \not\leq Q(\mathbf{f})$, we mean, in particular, that $P(x)$ and $Q(x)$ do not involve μ–assertions themselves, and x is the only possible free variable in P and Q.

Proposition 6.3.2 The rule $(\neg + \mu)$ is sound.

Proof It is enough to show that $[\![\mathbf{t}]\!] \not\subseteq [\![\mu x.\, P(x)]\!]$ and $[\![\mathbf{t}]\!] \not\subseteq [\![\mu x.\, Q(x)]\!]$. We give a proof for the first case; the proof for the second case is similar. By the lemma given in the previous section, there exists p and R, such that

$$\vdash P(x) = p \vee R(x)$$

where p is closed and $R(\mathbf{f}) = \mathbf{f}$, with R distributing over $\vee$. Clearly $[\![\mathbf{t}]\!] \not\subseteq [\![p]\!]$ and $[\![\mathbf{t}]\!] \subseteq [\![R(q)]\!]$ implies $[\![\mathbf{t}]\!] \subseteq [\![q]\!]$. If it were the case that

$$[\![\mathbf{t}]\!] \subseteq [\![\mu x.\, P(x)]\!]$$

then

$$[\![\mathbf{t}]\!] \subseteq [\![\mu x.\, p \vee R(x)]\!],$$

or

$$[\![\mathbf{t}]\!] \subseteq \bigcup_{i\in\omega} [\![R^i(p)]\!].$$

Therefore $[\![\mathbf{t}]\!] \subseteq [\![R^n(p)]\!]$ for some n, which implies $[\![\mathbf{t}]\!] \subseteq [\![p]\!]$, a contradiction. ∎

Now

$$(\mu x.\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x) \wedge (\mu x.\, \dot{inr}\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x) = \mathbf{f}$$

can be derived in the following way. Clearly

$$\dot{inl}\, \mathbf{t}_\perp \wedge \dot{inr}\, \dot{inl}\, \mathbf{t}_\perp = \mathbf{f}.$$

By Proposition 6.2.7,

$$\dot{inr}^2(\mu x.\, \dot{inr}\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x) = \dot{inr}(\mu x.\, \dot{inr}^2\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x).$$

We have $\mathbf{t} \not\leq \mathbf{t}_\perp$ and $\mathbf{t} \not\leq \dot{inr}^2\, \dot{inl}\, \mathbf{t}_\perp$. By $(\neg + \mu)$,

$$\dot{inl}\, \mathbf{t}_\perp \wedge \dot{inr}(\mu x.\, \dot{inr}^2\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x) = \mathbf{f},$$

so

$$\dot{inl}\, \mathbf{t}_\perp \wedge \dot{inr}^2(\mu x.\, \dot{inr}\, \dot{inl}\, \mathbf{t}_\perp \vee \dot{inr}^2\, x) = \mathbf{f}.$$

Similarly,

$$\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \wedge \dot{inr}^2(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) = \mathbf{f}.$$

Therefore

$$\begin{aligned}
&(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) \wedge (\,\mu x.\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,)\\
&= (\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,)\,) \wedge (\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2(\,\mu x.\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,)\,)\\
&= \dot{inr}^2[(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) \wedge (\,\mu x.\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,)]\\
&\leq \dot{inr}^2(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,).
\end{aligned}$$

Now, by taking p to be $\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x$ in $(\wedge - \mu)$, we get

$$\begin{aligned}
&(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) \wedge (\,\mu x.\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,)\\
&\leq \mu y.[\,(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) \wedge (\,\dot{inr}\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,y\,)\,]\\
&\leq \mu y.[\,(\,\mu x.\,\dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2\,x\,) \wedge \dot{inr}^2\,y\,]\\
&\leq \mathbf{f}.
\end{aligned}$$

Proposition 6.3.3 Suppose

$$\vdash p \wedge (\mu x.\,q \vee R(x)) = \mathbf{f},$$

$$\vdash q \wedge (\mu x.\,p \vee R(x)) = \mathbf{f},$$

where R is distributive. Then

$$\vdash (\mu x.\,p \vee R(x)) \wedge (\mu x.\,q \vee R(x)) = \mathbf{f}.$$

Proof We have

$$\begin{aligned}
&(\mu x.\,p \vee R(x)) \wedge (\mu x.\,q \vee R(x))\\
&= (\,p \vee R(\mu x.\,p \vee R(x))\,) \wedge (\,q \vee R(\mu x.\,q \vee R(x))\,)\\
&= R[\,(\mu x.\,p \vee R(x)) \wedge (\mu x.\,q \vee R(x))\,]\\
&\leq R(\mu x.\,p \vee R(x)).
\end{aligned}$$

By $(\wedge - \mu)$,

$$(\mu x.\, p \vee R(x)) \wedge (\mu x.\, q \vee R(x))$$

$$\leq \mu y.\, (\mu x.\, p \vee R(x)) \wedge (\, q \vee R(y)\,)$$

$$\leq \mu y.\, (\mu x.\, p \vee R(x)) \wedge R(y)$$

$$\leq \mu y.\, R(y)$$

$$= \mathbf{f}$$

∎

We remark that the rule $(\wedge - \mu)$ is only used to get Proposition 6.3.3. It is the result of Proposition 6.3.3 that is directly used in proving completeness in the next section. Hence the result of the next section also holds if instead we introduce the result of Proposition 6.3.3 as a new rule.

The following proposition tells us how to reduce some form of conjunction of μ−assertions to a single μ−assertion.

Proposition 6.3.4 Suppose Q is distributive and

$$p \wedge Q^i(\, \mu x.\, p \vee Q^{\mathrm{lcm}(m,n)}(x)\,) = \mathbf{f}$$

for $0 < i < 2 \cdot \mathrm{lcm}(m,n)$ with $m > 0$, $n > 0$, where $\mathrm{lcm}(m,n)$ is the *least common multiple* of m and n. Then

$$(\mu x.\, p \vee Q^m(x)) \wedge (\mu y.\, p \vee Q^n(y)) = \mu z.\, p \vee Q^{\mathrm{lcm}(m,n)}(z).$$

Proof Suppose $m \neq n$. By Proposition 6.2.6, we have

$$\mu z.\, p \vee Q^{\mathrm{lcm}(m,n)}(z) \leq \mu x.\, p \vee Q^m(x),$$

$$\mu z.\, p \vee Q^{\mathrm{lcm}(m,n)}(z) \leq \mu y.\, p \vee Q^n(y).$$

Therefore,

$$\mu z.\, p \vee Q^{\mathrm{lcm}(m,n)}(z) \leq (\mu x.\, p \vee Q^m(x)) \wedge (\mu y.\, p \vee Q^n(y)).$$

On the other hand, write $\mathrm{lcm}(m,n)$ as L and assume $L = sm = tn$, for some $s > 0$, $t > 0$. By Proposition 6.2.6 again,

$$\mu x.\, p \vee Q^m(x) = \mu x.\, \bigvee_{i=0}^{s-1} Q^{i \cdot m}(p) \vee Q^L(x),$$

$$\mu x.\, p \vee Q^n(y) = \mu y.\, \bigvee_{j=0}^{t-1} Q^{j\cdot n}(p) \vee Q^L(y).$$

Hence

$$[\,\mu x.\, p \vee Q^m(x)\,] \wedge [\,\mu y.\, p \vee Q^n(y)\,] = \; [\,\textstyle\bigvee_{i=0}^{s-1}(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,)\,] \wedge$$

$$[\,\textstyle\bigvee_{j=0}^{t-1}(\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,)\,].$$

It is enough to show that, for $0 < i < s$, $0 < j < t$,

$$\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right] \wedge \left[\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,\right] = \mathbf{f}.$$

In fact,

$$\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right] \wedge \left[\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,\right]$$

$$= \left[\, Q^{i\cdot m}(p) \vee Q^L(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,)\,\right] \wedge \left[\, Q^{j\cdot n}(p) \vee Q^L(\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,)\,\right]$$

$$= Q^L\left[\,(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,) \wedge (\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,)\,\right]$$

$$\vee \left[\, Q^{i\cdot m}(p) \wedge Q^L(\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,)\,\right] \vee \left[\, Q^{j\cdot n}(p) \wedge Q^L(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,)\,\right].$$

However

$$Q^{i\cdot m}(p) \wedge Q^L(\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,) = \mathbf{f},$$

$$Q^{j\cdot n}(p) \wedge Q^L(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,) = \mathbf{f},$$

by assumption. Hence

$$\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right] \wedge \left[\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,\right]$$

$$= Q^L(\,\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right] \wedge \left[\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,\right]\,)$$

$$\leq Q^L(\,\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right]\,)$$

Applying $(\wedge - \mu)$ we get

$$\left[\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,\right] \wedge \left[\,\mu y.\, Q^{j\cdot n}(p) \vee Q^L(y)\,\right]$$

$$\leq \mu y.\, \left[\,(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,) \wedge (\,Q^{j\cdot n}(p) \vee Q^L(y)\,)\,\right]$$

$$\leq \mu y.\, \left[\,(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,) \wedge Q^L(y)\,\right]$$

$$\leq \mathbf{f}.$$

Note that

$$(\,\mu x.\, Q^{i\cdot m}(p) \vee Q^L(x)\,) \wedge Q^{j\cdot n}(p) = \mathbf{f}$$

since

$$\mu x.\ Q^{i\cdot m}(p) \vee Q^L(x) = Q^{i\cdot m}(p) \vee Q^L(\ \mu x.\ Q^{i\cdot m}(p) \vee Q^L(x)\ ),$$

$$Q^{i\cdot m}(p) \wedge Q^{j\cdot n}(p) = \mathbf{f},$$

and

$$Q^{j\cdot n}(p) \wedge Q^L(\ \mu x.\ Q^{i\cdot m}(p) \vee Q^L(x)\ ) = \mathbf{f}.$$

∎

As a special case of Proposition 6.3.4, we have

$$(\mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^2 x) \wedge (\mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee \dot{inr}^3 x\ ) = \mu x.\ \dot{inl}\,\mathbf{t}_\perp \vee inr^6 x.$$

6.4 Soundness, Completeness, and Expressiveness

In this section we give soundness, completeness, and expressiveness results for the μ–calculus. The proof of completeness for type $\mathbf{N}_\perp$ is given in detail, via a normal form theorem. The expressiveness of assertions of $\mathbf{N}_\perp$ is then immediate. It is conjectured that in general the completeness of the μ–calculus can be shown through completeness of assertions in normal form, following the same pattern as the one given for $\mathbf{N}_\perp$.

Since this section is concerned specially with the μ–calculus for the integers $\mathbf{N}_\perp$, we shall abbreviate $\dot{inl}\ \mathbf{t}_\perp$ as 0 and $\dot{inr}$ as s, the successor function.

We have presented all the axioms and rules for the μ–calculus. Now when we write $\vdash \varphi \leq \psi$ we mean the formula $\varphi \leq \psi$ is provable from those axioms and rules. The μ–calculus is sound: the soundness of the μ–axiom and the μ–rule follows from Proposition 6.1.2, and the rest of the axioms and rules have already been shown to preserve validity of formulae. Therefore, we have

Theorem 6.4.1 Suppose φ, ψ are assertions of type σ. If $\vdash \varphi \leq \psi$, then for any environment ρ, $[\![\varphi]\!]\rho \subseteq [\![\psi]\!]\rho$.

To give a proof for the completeness of the μ–calculus of $\mathbf{N}_\perp$, we first show that essentially each assertion of $\mathbf{N}_\perp$ is provable to be equivalent to a finite disjunction of μ–assertions of the form

$$\mu x.\ s^m\, 0 \vee s^n(x).$$

We put two syntactic restrictions on the μ−assertions: i. When a μ−assertion $\mu x.\, P(x)$ is formed, x is the only possible free variable in P; ii. When a conjunction $\varphi_0 \wedge \varphi_1$ is formed, φ_0 and φ_1 are required to be closed. These restrictions make the task of proving completeness feasible but it does not affect the expressive power of μ−assertions.

Definition 6.4.1 A closed assertion φ of $\mathbf{N}_\perp$ is called a normal form if

$$\varphi \equiv \bigvee_{0<k<n} \mu x.\, p_k \vee s^{j_k}(x)$$

where p_k's are either of the form $s^{i_k}\, 0$ or $\mathbf{t}$, or $\mathbf{f}$.

When an assertion is provably equivalent to a normal form we say that this assertion *has a normal form.* The following is a key theorem for completeness.

Theorem 6.4.2 (The first normal form theorem) Each closed assertion of $\mathbf{N}_\perp$ has a normal form.

Proof By structured induction. We show that the constructors $\wedge$, $\vee$, s, and μ preserve the property of having a normal form. For all the cases we assume that in the normal form no assertion is equivalent to either $\mathbf{t}$ or $\mathbf{f}$.

It is obvious that if φ_0 and φ_1 are in normal forms the so is $\varphi_0 \vee \varphi_1$.

If φ has a normal form then so does $s\varphi$, by Proposition 6.2.7.

Suppose φ_0 and φ_1 are in normal forms. To show $\varphi_0 \wedge \varphi_1$ has a normal form, it is enough to show that

$$(\ \mu x.\, s^i\, 0 \vee s^j(x)\ ) \wedge (\ \mu x.\, s^m\, 0 \vee s^n(x)\ )$$

has a normal form in general. By using Proposition 6.2.6 first and then Proposition 6.2.5 we can raise the power of the recursive unit so that both μ−assertions have the same power for the recursive units. Therefore we can assume $j = n$.

Suppose $i \neq m$, and, without loss of generality, $i < m$. We have

$$\vdash (\ \mu x.\, s^i\, 0 \vee s^n(x)\ ) \wedge (\ \mu x.\, s^m\, 0 \vee s^n(x)\ )$$

$$= s^i\, [\ (\ \mu x.\, 0 \vee s^n(x)\ ) \wedge (\ \mu x.\, s^{m-i}\, 0 \vee s^n(x)\ )\ ].$$

Thus it is sufficient to show that

$$(\ \mu x.\, 0 \vee s^n(x)\ ) \wedge (\ \mu x.\, s^k\, 0 \vee s^n(x)\ )$$

has a normal form. If

$$\vdash (\ \mu x.\ 0 \vee s^n(x)\ ) \wedge\ s^k\, 0 = \mathbf{f}$$

then by Proposition 6.3.3

$$\vdash (\ \mu x.\ 0 \vee s^n(x)\ ) \wedge (\ \mu x.\ s^k\, 0 \vee s^n(x)\ ) = \mathbf{f}$$

since it follows from $(\neg + \mu)$ that, when $k > 0$,

$$\vdash 0 \wedge (\ \mu x.\ s^k\, 0 \vee s^n(x)\ ) = \mathbf{f}.$$

Otherwise $k \geq n$, and we can easily show that

$$\vdash (\ \mu x.\ 0 \vee s^n(x)\ ) \wedge\ s^k\, 0 = s^k\, 0$$

by unwinding $\mu x.\ 0 \vee s^n(x)$ a number of times. Let t be the least number such that $t \cdot n = k$. Using Proposition 6.2.6 and Proposition 6.2.5 again,

$$\vdash \mu x.\ 0 \vee s^n(x) = \bigvee_{0 \leq i \leq t} \mu x.\ s^{ni} 0 \vee s^{(t+1)\cdot n}(x).$$

By Proposition 6.3.3,

$$\vdash (\ \mu x.\ s^{ni} 0 \vee s^{(t+1)\cdot n}(x)\ ) \wedge (\ \mu x.\ s^k\, 0 \vee s^n(x)\ ) = \mathbf{f}$$

when $i < t$. Therefore

$$\vdash (\ \mu x.\ 0 \vee s^n(x)\ ) \wedge (\ \mu x.\ s^k\, 0 \vee s^n(x)\ )$$

$$= (\ \mu x.\ s^k 0 \vee s^{(t+1)\cdot n}(x)\ ) \wedge (\ \mu x.\ s^k\, 0 \vee s^n(x)\ ).$$

We can then use Proposition 6.3.4 to get a normal form.

To show $\mu x.\ P(x) : \mathbf{N}_\perp$ has a normal form we first transform $P(x)$ into a disjunctive normal form

$$\vdash P(x) = p \vee \bigvee_{0 < i < n} s^{k_i}(x),$$

where p is closed. This is possible because of the two syntactic restrictions we imposed. Applying the generalised version of Proposition 6.2.8 and Proposition 6.2.5 we get

$$\vdash \mu x.\ P(x) = \bigvee_{0 < i < n} \mu x.\ p_i \vee s^{k_i}(x),$$

where p_i's can be assumed to be already of the form

$$\mu x.\ s^k 0 \vee s^m(x).$$

It remains to check that assertions of the form

$$\mu x.\ (\mu y.\ s^k 0 \vee s^j(y)\ ) \vee s^m(x)$$

have normal forms. But by Proposition 6.2.10,

$$\vdash \mu x.\ (\mu y.\ s^k 0 \vee s^j(y)\ ) \vee s^m(x)$$

$$= \mu x.\ s^k 0 \vee s^j(x) \vee s^m(x).$$

It follows from Proposition 6.2.8 that

$$\mu x.\ s^k 0 \vee s^j(x) \vee s^m(x)$$

has a normal form. Hence any such $\mu x.\ P(x)$ has a normal form. ∎

Note that it is easy, by Proposition 6.2.6, to transform any normal form into an assertion

$$p \vee (\mu x.\ p_0 \vee s^n(x)),$$

where p and p_0 are assertions of $\mathcal{A}_{\mathbf{N}_\perp}$ in their prime normal form (which does not involve sub-μ−assertions, neither free variables). We can further assume that for each disjuncts $s^k 0$ of p_0, $k < n$. This is indeed another normal form.

Theorem 6.4.3 (The second normal form theorem) Let $\varphi : \mathbf{N}_\perp$. Then

$$\vdash \varphi = p \vee (\mu x.\ p_0 \vee s^n(x)),$$

where p and p_0 are prime normal forms of $\mathcal{A}_{\mathbf{N}_\perp}$ and for each disjuncts $s^k 0$ of p_0, $k < n$.

We now prove that for assertions in the second normal forms our proof system is complete. Following the notation introduced before, we write $\models \varphi \leq \psi$ when

$$\forall \rho.\ [\![\varphi]\!]_\rho \subseteq [\![\varphi]\!]_\rho.$$

Theorem 6.4.4 Suppose φ and ψ are assertions of $\mathbf{N}_\perp$, in the second normal form. Then $\models \varphi \leq \psi$ implies $\vdash \varphi \leq \psi$.

Proof Suppose

$$\varphi \equiv p \vee (\mu x.\ p_0 \vee s^n(x))$$

and

$$\psi \equiv q \vee (\mu x.\ q_0 \vee s^m(x))$$

and suppose $\models \varphi \leq \psi$. Since $\models \varphi \leq \psi$, we can unwind ψ a number of times so that $\models p \leq q$. Hence $\vdash p \leq \psi$. Now by virtue of Proposition 6.2.6 we may assume, without loss of generality, $n = m$. We have

$$\models \mu x.\ p_0 \vee s^n(x) \leq q \vee (\mu x.\ q_0 \vee s^n(x)).$$

It is sufficient to prove that $\models p_0 \leq q_0$. Suppose $\not\models p_0 \leq q_0$. Then for some disjuncts $s^k 0$ of p_0, $[\![s^k 0]\!] \neq [\![s^i 0]\!]$ for any disjuncts $s^i 0$ of q_0. This implies

$$\not\models \mu x.\ p_0 \vee s^n(x) \leq q \vee (\mu x.\ q_0 \vee s^n(x)),$$

a contradiction. ∎

Immediately there is the

Corollary (Completeness of μ−calculus for integers) The μ−calculus for $\mathbf{N}_\perp$, consisting of $\mu-$ axiom, $\mu-$ rule, $(\wedge - \mu)$, $(\neg + \mu)$, and those rules for $\mathcal{A}_{\mathbf{N}_\perp}$ is complete provided we impose the two syntactic restrictions on the assertions.

Expressiveness is then clear: the open set expressible by the restricted form of μ−assertions of $\mathbf{N}_\perp$ are $\emptyset$, $\mathbf{N}_\perp$ and 'natural numbers' of the form

$$N_0 \cup \bigcup_{0<i<n} \{\, m_i + k \cdot n_i \mid k \in \omega \,\}$$

where m_i, n_i are natural numbers and N_0 is a finite set of natural numbers. This result agrees with the one mentioned in [Sst87] where the semantics of a μ−assertion language is investigated but no proof systems proposed. Our second normal form theorem tells us, however, a rather interesting result that open sets of the above class can also be represented in the form

$$N_0 \cup \bigcup_{k \in \omega} (k \cdot n_0) + N_1,$$

where n_0 is a natural number, N_0 and N_1 are finite sets of natural numbers, and

$$k + N =^{\text{def}} \{\, k + n \mid n \in N \,\}$$

for a natural number k and a set N.

From the expressiveness result we know that the restriction on the syntax does not affect the expressive power. This fact is expected to be true for other types, too. But we have to explain why there should be such a restriction. The first reason for the restriction

is that we have not yet found any way to bring the $\mu y.$ construction outside assertions of the form

$$\mu x.\ P(\ x,\ \mu y.\ Q(x,y))$$

to get a reasonable simple normal form. The second reason for the syntactic restriction comes from the fact that we have no obvious way to 'simplify' an assertion like

$$\mu x.\ p_0 \vee (s^m(x) \wedge p_1) \vee (s^n(x) \wedge p_2)$$

of $\mathbf{N}_\perp$ in general.

Chapter 7

Categories of DI-Domains and Event Structures

The remaining part of the thesis is devoted to developing logics of dI-domains. To be more precise, we aim to build a logic analogous to that given in Chapter 5 on each category of domains (in a broader sense). There are many different categories of domains with dI-domains as objects and certain kind of stable functions as morphisms. However, they are scattered in the literature. To provide a background knowledge for the work in later chapters, this chapter gives a survey of such categories (some of them are new) and indicates relationships among them (if possible). We actually present more categories than that are used later, so as to make this chapter a useful reference for different categories of dI-domains. These categories have been discovered with different computational motivations. Due to the lack of space we cannot give a full explanation of the motivations here. But this chapter is hoped to contain enough information on the mathematics side of the categories.

The following is a list of the categories we are going to discuss:

DI	—	dI-domains with stable functions
DL	—	dI-domains with linear functions
$\mathbf{SEV}_s$	—	stable event structures with stable functions
$\mathbf{SEV}_l$	—	stable event structures with linear functions
$\mathbf{SEV}^*_{syn}$	—	stable event structures with partially synchronous morphisms
$\mathbf{SEV}_{syn}$	—	stable event structures with synchronous morphisms
$\mathbf{SF}_s$	—	stable families with stable functions
$\mathbf{SF}_l$	—	stable families with linear functions

and

$\mathbf{COH}_s$ — coherent families with stable functions

$\mathbf{COH}_l$ — coherent families with linear functions

$\mathbf{FF}$ — finitary families with linear maps

where $\mathbf{DI}$ is due to Berry [Be78]; $\mathbf{SEV}_s$, $\mathbf{SEV}^*_{syn}$, $\mathbf{SEV}_{syn}$ and $\mathbf{SF}_s$ are due to Winskel [Wi82], [Wi86], [Wi88]; $\mathbf{COH}_s$ and $\mathbf{COH}_l$ are due to Girard [Gi87a], [Gi87b], though presented in a slightly different form here; and $\mathbf{SEV}_l$, $\mathbf{FF}$ are new. The remaining $\mathbf{DL}$ and $\mathbf{SF}_l$ are known to exist, but people have not really used them yet. There is some recent work on representing $\mathbf{DI}$ as *stable information systems* [Zh89]. The category $\mathbf{SIS}$ of stable information systems is equivalent to $\mathbf{DI}$. But it is not included in this chapter due to lack of space.

This chapter is organised as follows. Section 1 presents cartesian closed categories from the above list. Section 2 gives an introduction to $\mathbf{SEV}_{syn}$ and $\mathbf{SEV}^*_{syn}$, categories for concurrency. Section 3 introduces monoidal closed categories. In the last section relationships among the categories are discussed.

7.1 Cartesian Closed Categories

This section presents cartesian closed categories

$$\mathbf{DI},\ \mathbf{SEV}_s,\ \mathbf{SF}_s,\ \mathbf{COH}_s$$

among the list given at the begining.

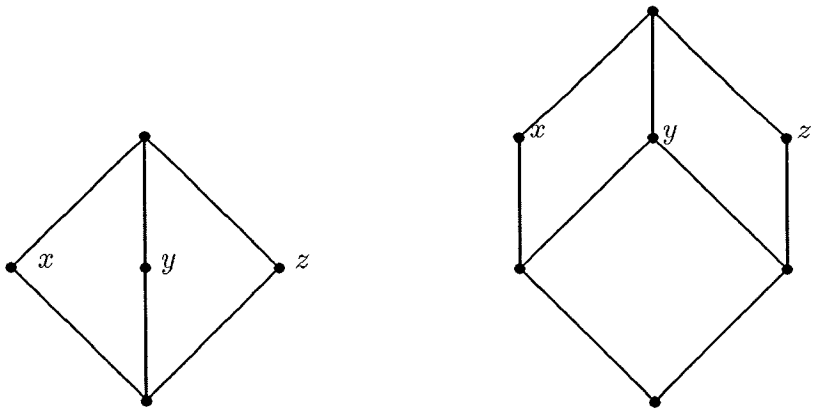
The Category DI

DI-Domains were discovered by Berry [Be78] from the study of the full-abstraction problem for typed λ-calculi. They are special kinds of Scott domains which have a more operational interpretation. The functions between dI-domains are stable functions under an order which takes into account the manner in which they compute. DI-domains with stable functions form a cartesian closed category, making it an alternative framework in which to do denotational semantics.

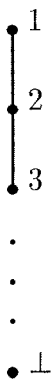
Definition 7.1.1 A dI-domain is a consistently complete domain D which satisfies

- axiom d: $\forall x, y, z \in D.\ y \uparrow z \Longrightarrow x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$
- axiom I: $\forall d \in D^0.\ |\{ x \mid x \sqsubseteq d \}| < \infty$

Example 7.1.1 Two Scott domains lacking distributivity(axiom d):



Example 7.1.2 A Scott domain which violates axiom I:



Proposition 7.1.1 A domain D satisfies axiom d iff for all $x, y, z \in D$

$$\{ x, y, z \} \uparrow \Longrightarrow x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z).$$

Proof

($\Rightarrow$): Trivial.

($\Leftarrow$): For any compatible pair y, z in D, $\{x \sqcap (y \sqcup z),\ y,\ z\}$ is a compatible set.

Therefore,

$$[x \sqcap (y \sqcup z)] \sqcap (y \sqcup z) = \Big[[x \sqcap (y \sqcup z)] \sqcap y\Big] \sqcup \Big[[x \sqcap (y \sqcup z)] \sqcap z\Big],$$

or

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z).$$

∎

This proposition means we can replace axiom d by a seemingly weaker one, requiring distributivity to hold only for compatible triples.

Definition 7.1.2 A function $f : D \to E$ between two dI-domains D and E is stable if it is Scott-continuous and preserves meets of pairs of compatible elements, i.e.,

$$\forall x,\ y \in D.\ x \uparrow y \Longrightarrow f(x \sqcap y) = f(x) \sqcap f(y).$$

It is *linear* if it is stable and, further,

$$\forall X \subseteq D.\ X \uparrow \Longrightarrow f(\bigsqcup X) = \bigsqcup \{\, f(x) \mid x \in X \,\}.$$

Definition 7.1.3 Let f, g be two stable functions from D to E. f stably less than g, written $f \sqsubseteq_s g$, if $\forall x,\ y \in D.\ x \sqsubseteq y \Longrightarrow f(x) = f(y) \sqcap g(x)$. Write $[D \to_s E]$ for the set of stable functions from D to E.

DI-domains with stable functions form a category **DI**.

Theorem 7.1.1 (Berry) **DI** is a cartesian closed category. The products are the Cartesian product ordered coordinatewise and the function space of dI-domains D and E consists of the stable functions between them and ordered under the stable order.

The following lemmas will be frequently used.

Lemma 7.1.1 Let D, E be dI-domains and $f, g : D \to E$ stable functions. $f \uparrow g$ iff

$$\forall x,\ y \in D.\ x \uparrow y \Longrightarrow \begin{cases} f(x) \sqcap g(y) = f(x \sqcap y) \sqcap g(x \sqcap y) \\ f(x) \uparrow g(y) \end{cases}$$

Proof

$(\Rightarrow)$: There exists $h \in D \to_s E$ such that $f \sqsubseteq_s h$ and $g \sqsubseteq_s h$ as $f \uparrow g$. Hence

$$f(x \sqcap y) = f(x) \sqcap h(x \sqcap y),$$

$$g(x \sqcap y) = g(y) \sqcap h(x \sqcap y).$$

Therefore

$$f(x) \sqcap g(y) \sqcap h(x \sqcap y) = f(x \sqcap y) \sqcap g(x \sqcap y),$$

or

$$f(x) \sqcap g(y) = f(x \sqcap y) \sqcap g(x \sqcap y).$$

($\Leftarrow$) : Define $H = \lambda x.f(x) \sqcup g(x)$. We show that $f \sqsubseteq_s H$ and $g \sqsubseteq_s H$. $f \sqsubseteq_s H$ because for y, $z \in D$ and $y \sqsubseteq z$,

$$\begin{aligned}
& f(z) \sqcap g(y) = f(y) \sqcap g(y) \\
& \quad \Longrightarrow f(y) = f(y) \sqcup (f(z) \sqcap g(y)) \\
& \quad \Longrightarrow f(y) = (f(z) \sqcap f(y)) \sqcup (f(z) \sqcap g(y)) \qquad (\ f(y) \sqsubseteq f(z)\ ) \\
& \quad \Longrightarrow f(y) = f(z) \sqcap \big[f(y) \sqcup g(y)\big] \qquad (\ E \text{ is distributive}\ ) \\
& \quad \Longrightarrow f(y) = f(z) \sqcap H(y).
\end{aligned}$$

Similarly $g \sqsubseteq_s H$ holds. H is clearly stable. ∎

The following two corollaries are obvious.

Corollary 7.1.1 If f and g are compatible and f is pointwisely less than g then it is stablely less than g.

Corollary 7.1.2 $f \uparrow g$ implies $\forall x, y\,.\, x \uparrow y \Longrightarrow f(x) \sqcap g(y) = f(y) \sqcap g(x)$.

Lemma 7.1.2 If $F \subseteq [D \to_s E]$ is compatible then

$$\bigsqcup F = \lambda x. \bigsqcup_{f \in F} f(x).$$

Proof It is enough to show that $\lambda x. \bigsqcup_{f \in F} f(x)$ is continuous, stable and greater than any function in F under the stable order.

Continuity: Let

$$x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$$

be a chain in D. Obviously

$$\begin{aligned}
\bigsqcup_{f \in F} f\Big(\bigsqcup_{i \in \omega} x_i\Big) &= \bigsqcup_{f \in F} \Big(\bigsqcup_{i \in \omega} f(x_i)\Big) \\
&= \bigsqcup_{i \in \omega} \Big(\bigsqcup_{f \in F} f(x_i)\Big).
\end{aligned}$$

Hence it is continuous.

Stability: For any compatible pair x, y in D, we have

$$\begin{aligned}
\bigsqcup_{f\in F} f(x\sqcap y) &\sqsubseteq \big(\bigsqcup_{f\in F} f(x)\big)\sqcap\big(\bigsqcup_{g\in F} g(y)\big)\\
&= \bigsqcup_{f\in F}\bigsqcup_{g\in F}\big(f(x)\sqcap g(y)\big)\\
&= \bigsqcup_{f\in F}\bigsqcup_{g\in F}\big(f(x\sqcap y)\sqcap g(x\sqcap y)\big) \qquad \text{(by Lemma 7.1.1)}\\
&\sqsubseteq \bigsqcup_{f\in F} f(x\sqcap y).
\end{aligned}$$

Therefore

$$\bigsqcup_{f\in F} f(x\sqcap y) = \big(\bigsqcup_{f\in F} f(x)\big)\sqcap\big(\bigsqcup_{g\in F} g(y)\big).$$

Let $y\sqsubseteq z$ and $g\in F$. We have

$$\begin{aligned}
g(y) &\sqsubseteq g(z)\sqcap\bigsqcup_{f\in F} f(y)\\
&= \bigsqcup_{f\in F}\big(g(z)\sqcap f(y)\big)\\
&= \bigsqcup_{f\in F}\big(g(y\sqcap z)\sqcap f(y\sqcap z)\big) \qquad \text{(by Lemma 7.1.1)}\\
&\sqsubseteq g(y),
\end{aligned}$$

which implies

$$g(y) = g(z)\sqcap\bigsqcup_{f\in F} f(y).$$

In other words,

$$g\sqsubseteq_s \lambda x.\bigsqcup_{f\in F} f(x).$$

∎

Now we are in a position to explain why axiom I is necessary. It has to do with the fact that stable functions between $\omega-$algebraic, distributive Scott domains under the stable order need not necessary form an $\omega-$algebraic domain: There can be too many finite elements in the function space. Consider the stable function space from the domain given in Example 7.1.2 to itself, where the order of elements in the domain is the reverse of the order for natural numbers. Each stable function f corresponds to an infinite sequence

$$(f(1),\ f(2),\ f(3),\cdots,\ f(n),\cdots)$$

where $i \sqsubseteq j \Longrightarrow f(i) \sqsubseteq f(j)$. We claim that all the stable functions are finite elements of the stable function space.

Let $\{\phi_i \mid i \in I\}$ be a directed set of stable functions such that $\bigsqcup_{i\in I} \phi_i \sqsupseteq_s f$. Especially, $\bigsqcup_{i\in I} \phi_i(1) \sqsupseteq f(1)$, by Lemma 7.1.2. Therefore, there exists $i_0 \in I$ such that $\phi_{i_0}(1) \sqsupseteq f(1)$, as $f(1)$ is a finite element. Since ϕ_{i_0} and f are compatible, for all k

$$f(k) \sqcap \phi_{i_0}(1) = \phi_{i_0}(k) \sqcap f(1).$$

Hence for all k

$$\begin{aligned} f(k) &= f(k) \sqcap f(1) \\ &= f(k) \sqcap \phi_{i_0}(1) \\ &= \phi_{i_0}(k) \sqcap f(1) \\ &\sqsubseteq \phi_{i_0}(k). \end{aligned}$$

This actually implies $f \sqsubseteq_s \phi_{i_0}$ by virtue of Corollary 7.1.1, which shows that f is a finite element in the function space. By the diagonal argument we deduce that there are uncountably many such functions. Thus the function space is not $\omega-$algebraic.

Lemma 7.1.3 If $F \subseteq [D \to_s E]$ is compatible then

$$\bigsqcap F = \lambda x. \bigsqcap_{f\in F} f(x).$$

Proof First we verify the non-trivial fact that $\lambda x. \bigsqcap_{f\in F} f(x)$ is continuous. Let

$$x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$$

be a chain in D. What is needed is the equation

$$\bigsqcap_{f\in F} \Big[\bigsqcup_{i\in\omega} f(x_i)\Big] = \bigsqcup_{i\in\omega} \Big[\bigsqcap_{f\in F} f(x_i)\Big].$$

Clearly

$$\bigsqcap_{f\in F} \Big[\bigsqcup_{i\in\omega} f(x_i)\Big] \sqsupseteq \bigsqcup_{i\in\omega} \Big[\bigsqcap_{f\in F} f(x_i)\Big].$$

The argument for the inequality in the other direction runs as follows:

$$\begin{aligned} d \in D^0 \;\&\; \bigsqcap_{f\in F} \Big[\bigsqcup_{i\in\omega} f(x_i)\Big] &\sqsupseteq d \\ &\Longrightarrow \forall f \in F.\ \bigsqcup_{i\in\omega} f(x_i) \sqsupseteq d \\ &\Longrightarrow \forall f \in F\ \exists i_f.\ f(x_{i_f}) \sqsupseteq d \\ &\Longrightarrow \bigsqcap_{f\in F} f(x_{i_f}) \sqsupseteq d. \end{aligned}$$

Let $i_h = \min\{ i_f \mid f \in F \}$ where $h \in F$. By Lemma 7.1.1 we have

$$\forall f \in F.\ h(x_{i_h}) \sqcap f(x_{i_f}) = h(x_{i_h}) \sqcap f(x_{i_h}).$$

Therefore,

$$\prod_{f \in F} f(x_{i_f}) = \prod_{f \in F} f(x_{i_h}),$$

which implies

$$\bigsqcup_{i \in \omega} \Big[\prod_{f \in F} f(x_i) \Big] \sqsupseteq \prod_{f \in F} f(x_{i_h}) \sqsupseteq d.$$

That $\lambda x. \prod_{f \in F} f(x)$ is stable and $\lambda x. \prod_{f \in F} f(x) \sqsubseteq_s g$ for all $g \in F$ are routine. ∎

Notice that we require F to be compatible. Meet (greatest lower bound) always exists but may not equal to the function specified by pointwise meets.

It is useful to know the connection between distributivity and prime algebraicness.

Definition 7.1.4 Let D be a consistently complete partial order. A *complete prime* of D is an element $p \in D$ such that

$$p \sqsubseteq \bigsqcup X \Longrightarrow \exists x \in X.\, p \sqsubseteq x$$

for all compatible set X. D is *prime algebraic* if

$$x = \bigsqcup \{ p \mid p \sqsubseteq x \ \&\ p \text{ is a complete prime} \}$$

for all $x \in D$.

Lemma 7.1.4 (Winskel) Suppose D is a consistently complete partial order which satisfies axiom I. Then D is a prime algebraic domain iff it is a dI-domain.

Proof See [Wi88]. ∎

The Category $\mathbf{SEV}_s$

DI-domains can be represented as stable event structures which are models for processes of concurrent computation. An event structure is a description of a set of events in terms of *consistency* and *enabling* relations. The consistency relation indicates whether some

events can occur together or not, and the enabling relation specifies the condition when a particular event may occur with regards to the occurrence of other events. A *configuration* of an event structure is a set of events which is consistent and each of its event is enabled by a set of events of the configuration occurred previously. Therefore, a configuration is a set of events which have occurred by certain stage in a process. More formally,

Definition 7.1.5 An *event structure* is a triple

$$\underline{E} = (E,\ Con,\ \vdash)$$

where

- E is a countable set of events,
- Con is a non-empty subset of $\mathrm{Fin}(E)$, the finite subsets of E

  called the consistency predicate which satisfies

  $$X \subseteq Y \ \&\ Y \in Con \Longrightarrow X \in Con,$$
- $\vdash \subseteq Con \times E$ is the enabling relation which satisfies

  $$(X \vdash e \ \&\ X \subseteq Y \ \&\ Y \in Con) \Longrightarrow Y \vdash e.$$

It is not essential to require E to be countable. When $X \vdash e$, we say e is enabled by X. Although an event structure looks similar to an information system, it is based on a different intuition and they are totally different structures. Typically, for an information system if $X \vdash a$ and $X \vdash a'$, a and a' must be consistent while for an event structure, we cannot say anything about the consistency of two events e, e' enabled by the same set of events.

Definition 7.1.6 Let $\underline{E} = (E,\ Con,\ \vdash)$ be an event structure. A configuration of $\underline{E}$ is a subset $x \subseteq E$ which is

- consistent: $\forall X \subseteq^{fin} x.\ X \in Con$,
- secured: $\forall e \in x\ \exists e_0, e_1, \cdots, e_n \in x.\ e_n = e\ \&$

  $$\forall i \leq n.\ \{ e_k \mid 0 \leq k \leq i-1 \} \vdash e_i.$$

Write the set of configurations of an event structure $\underline{E}$ as $\mathcal{F}(\underline{E})$.

There is a special class of event structures for which each configuration determines a partial order of causal dependency on the events. Intuitively, an event e_1 causally depends on an event e_0 if the occurrence of the event e_0 is necessary in order for e_1 to occur. Event structures of this kind are called *stable*.

Definition 7.1.7 An event structure $\underline{E}$ is stable if it satisfies the following axiom

$$(X \vdash e \;\&\; Y \vdash e \;\&\; X \cup Y \cup \{e\} \in Con) \Longrightarrow X \cap Y \vdash e.$$

Theorem 7.1.2 (Winskel) Let $\underline{E}$ be a stable event structure. Then $(\mathcal{F}(\underline{E}), \subseteq)$ is a dI-domain.

Stable event structures with stable function on the set of configurations form a category $\mathbf{SEV}_s$ which is equivalent to the category of dI-domains. The function space of stable event structures are defined as follows.

Definition 7.1.8 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. Their function space, $[\underline{E_0} \to \underline{E_1}]$, is defined to be the event structure $(E, Con, \vdash)$ with events E consisting of pairs (x, e) where x is a finite configuration in $\mathcal{F}(\underline{E_0})$ and $e \in E_1$, a consistency predicate Con given by

$\{(x_i, e_i) \mid 0 \leq i \leq n-1\} \in Con$ iff

- $\forall I \subseteq \{0, 1, \cdots, n-1\}.\ \bigcup_{i \in I} x_i \in Con_0 \Longrightarrow \{e_i \mid i \in I\} \in Con_1$
- $\forall i, j \leq n.\ (x_i \uparrow x_j \;\&\; e_i = e_j) \Longrightarrow x_i = x_j,$

and an enabling relation given by

$$\{(x_i, e_i) \mid 0 \leq i \leq n-1\} \vdash (x, e) \text{ iff } \{e_i \mid x_i \subseteq x\} \vdash_1 e.$$

Theorem 7.1.3 (Winskel) The stable function space of two stable event structures is a stable event structure. There is a $1-1$ order preserving correspondence between configurations $\mathcal{F}[\underline{E_0} \to \underline{E_1}]$ with set inclusion as its order and stable, continuous functions $\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})$ with the stable order. The associated maps are

$$\phi :\ \mathcal{F}[\underline{E_0} \to \underline{E_1}] \to [\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})]$$

by letting

$$F \longmapsto \lambda x \in \mathcal{F}(\underline{E_0}).\ \{e \in E_1 \mid \exists x' \subseteq x.\ (x', e) \in F\},$$

and

$$\mu :\ [\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})] \to \mathcal{F}[\underline{E_0} \to \underline{E_1}]$$

by letting

$$f \longmapsto \{(x, e) \mid e \in f(x) \;\&\; (\forall x' \subseteq x.\ e \in f(x') \Longrightarrow x' = x)\}.$$

We mentioned earlier that a stable event structure is an event structure each configuration of which determines a partial order of causal dependency on the events of the configuration. This partial order is specified as follows.

Definition 7.1.9 Let $\mathcal{F}(\underline{E})$ be the family of configurations of a stable event structure $\underline{E}$. Let $x \in \mathcal{F}(\underline{E})$ and $e, e' \in x$. Define

$$e' \leq_x e \text{ if } \forall y \in \mathcal{F}(\underline{E}).\ (e \in y \ \& \ y \subseteq x \Longrightarrow e' \in y)$$

$$\lceil e \rceil_x = \bigcap \{ y \in \mathcal{F}(\underline{E}) \mid e \in y \ \& \ y \subseteq x \}.$$

$\leq_x$ is a partial order and $\lceil e \rceil_x = \{ e' \in x \mid e' \leq_x e \}$. Some times $\lceil e \rceil_x$ is very useful to work with.

The Category $\mathbf{SF}_s$

Forgetting about the enabling and consistency relation but keeping the configurations of a stable event structure, one gets a *stable family*. *Stable families* are an axiomatisation of the configurations determined by stable event structures.

Recall some notations first. A subset X of a partial order $(P, \sqsubseteq)$ is *compatible* , written $X\uparrow$, if $\exists p \in P \forall x \in X.\, x \sqsubseteq p$. It is said to be *finitely compatible*, written $X\uparrow^{fin}$ if $Y\uparrow$ for any finite subset Y of X.

Definition 7.1.12 A *stable family* is a set of subsets $\mathcal{F}$ of a countable set E which satisfy

- finite completeness:
  $X \subseteq \mathcal{F} \ \& \ X\uparrow^{fin} \Longrightarrow \bigcup X \in \mathcal{F}$
- finiteness:
  $\forall x \in \mathcal{F} \forall e \in x \exists z \in \mathcal{F}.\ (\mid z \mid < \infty \ \& \ e \in z \ \& z \subseteq x)$
- stability:
  $\forall X \subseteq \mathcal{F}.\ X \neq \emptyset \ \& \ X\uparrow \Longrightarrow \bigcap X \in \mathcal{F}$
- coincidence freeness:
  $\forall x \in \mathcal{F} \forall e, e' \in x.\ e \neq e' \Longrightarrow (\exists y \in \mathcal{F}.\ y \subseteq x \ \& \ [e \in y \Leftrightarrow e' \notin y])$

Let $\mathcal{F}$ be a finitary family over E. The elements of $\mathcal{F}$ are called configurations and the elements of E called events. $\mathcal{F}$ is said to be *full* if $\bigcup \mathcal{F} = E$. If $\underline{E}$ is a stable event structure then $\mathcal{F}(\underline{E})$ is a stable family.

A stable family with set-inclusion gives a dI-domain. Stable families with stable function as morphisms form a cartesian closed category $\mathbf{SF}_s$. $\mathbf{SF}_s$ is equivalent to $\mathbf{SEV}_s$. Similar to $\mathbf{SEV}_s$, one can define constructions like sum, product, and stable function space on $\mathbf{SF}_s$.

The Category $\mathbf{COH}_s$

Coherent spaces are special kinds of dI-domains, or event structures. They were introduced first as a model for the system F [Gi87a], and later as a semantics for linear logic[Gi87b].

As a dI-domain, a coherent space is a domain D which is *coherent*, or *pairwise complete* in the sense that it has least upper bounds for pairwise compatible sets, and d is a complete prime iff

$$d \neq \bot_D \ \& \ (x \sqsubset d \Longrightarrow x = \bot_D).$$

As a stable event structure, a coherent space is a structure of the form

$$(\, E,\, Con,\, \{\, \emptyset \vdash e \mid e \in E \,\}\,),$$

where Con is determined by a conflict relation.

However for later use we present coherent spaces as special kinds of stable families called *coherent families*. It is assumed that coherent families are full as stable families. Thus it is not necessary to specify the event set.

Definition 7.1.13 Let $\mathcal{F}$ be a stable family. It is called a *coherent family* if

- $\forall x \in \mathcal{F}.\, y \subseteq x \Longrightarrow y \in \mathcal{F}$
- $(\, X \subseteq \mathcal{F} \ \& \ \forall x, y \in X.\, x \uparrow y \,) \Longrightarrow \bigcup X \in \mathcal{F}$

In fact the above conditions implies the axiom of finiteness, coincidence-freeness, and stability for stable families. So the above two axioms are enough to determine a coherent family.

Proposition 7.1.2 Let $\mathcal{F}$ be a coherent family. Then $(\, \mathcal{F},\, \subseteq\,)$ is a dI-domain with $\{\, \{\, a \,\} \mid \{\, a \,\} \in \mathcal{F} \,\}$ the set of its complete primes.

Proof Straightforward. ∎

Coherent families with stable functions form a cartesian closed category. The product, co-product (sum), and function space construction are defined as follows.

Definition 7.1.14 (Sum) Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families. Their *sum*, $\mathcal{F}_0 + \mathcal{F}_1$, is a family of subsets which satisfies

$$x \in \mathcal{F}_0 + \mathcal{F}_1 \iff \bullet\ x \subseteq (\{0\} \times \bigcup \mathcal{F}_0) \cup (\{1\} \times \bigcup \mathcal{F}_1)\ \&$$
$$\bullet\ \exists x_0 \in \mathcal{F}_0.\, x = \{0\} \times x_0 \text{ or } \exists x_1 \in \mathcal{F}_1.\, x = \{1\} \times x_1$$

Definition 7.1.15 (Product) Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families. Their *product*, $\mathcal{F}_0 \times \mathcal{F}_1$, is a family of subsets which satisfies

$$x \in \mathcal{F}_0 \times \mathcal{F}_1 \iff \bullet\ x \subseteq (\{0\} \times \bigcup \mathcal{F}_0) \cup (\{1\} \times \bigcup \mathcal{F}_1)\ \&$$
$$\bullet\ \exists x_0 \in \mathcal{F}_0,\ x_1 \in \mathcal{F}_1.\, x = \{0\} \times x_0 \cup \{1\} \times x_1$$

It is easy to see that the sum and the product of coherent families are still coherent families.

Definition 7.1.16 (Stable Function Space) Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families. Their *stable function space*, $\mathcal{F}_0 \to \mathcal{F}_1$, is a family of subsets which satisfies

$$x \in \mathcal{F}_0 \to \mathcal{F}_1 \iff \bullet\ x \subseteq \{y \in \mathcal{F}_0 \mid |\, y \,| < \infty\} \times \bigcup \mathcal{F}_1\ \&$$
$$\bullet\ \forall x_0 \in \mathcal{F}_0.\, \{\pi_1 s \mid \pi_0 s \subseteq x_0\ \&\ s \in x\} \in \mathcal{F}_1\ \&$$
$$\bullet\ [y_0 \cup y_0' \in \mathcal{F}_0\ \&\ (y_0, e_1),\ (y_0', e_1) \in x] \Longrightarrow e_0 = e_0'$$

By the coherence of $\mathcal{F}_0$ and $\mathcal{F}_1$, $\forall x_0 \in \mathcal{F}_0.\, \{\pi_1 s \mid \pi_0 s \subseteq x_0\ \&\ s \in x\} \in \mathcal{F}_1$ iff

$$\forall (y_0, e_1),\ (y_0', e_1') \in x.\, y_0 \cup y_0' \in \mathcal{F}_0 \Longrightarrow \{e_1, e_1'\} \in \mathcal{F}_1.$$

Proposition 7.1.3 The family $\mathcal{F}_0 \to \mathcal{F}_1$ is a coherent family which is isomorphic to $[\mathcal{F}_0 \to_s \mathcal{F}_1]$, the dI-domain of stable functions from $\mathcal{F}_0$ to $\mathcal{F}_1$.

Proof Suppose $y \subseteq x \in \mathcal{F}_0 \to \mathcal{F}_1$. Then for any $x_0 \in \mathcal{F}_0$, $\{\pi_1 s \mid \pi_0 s \subseteq x_0\ \&\ s \in x\} \in \mathcal{F}_1$. But

$$\{\pi_1 s \mid \pi_0 s \subseteq x_0\ \&\ s \in y\} \subseteq \{\pi_1 s \mid \pi_0 s \subseteq x_0\ \&\ s \in x\}.$$

Thus $\{ \pi_1 s \mid \pi_0 s \subseteq x_0 \ \& \ s \in y \} \in \mathcal{F}_1$. It is easy to see, therefore, that $\mathcal{F}_0 \to \mathcal{F}_1$ is a coherent familie.

Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families and $x \in \mathcal{F}_0 \to \mathcal{F}_1$. Define $Pt(x)$ to be a function from $\mathcal{F}_0$ to $\mathcal{F}_1$ such that for $x_0 \in \mathcal{F}_0$,

$$Pt\, x\, (x_0) = \{ e \in \bigcup \mathcal{F}_1 \mid \exists y_0 \subseteq x_0 .\ (y_0, e) \in x \}.$$

$Pt\, x$ is continuous because y is finite for any $(y, e) \in x$. If $x_0 \uparrow x_0'$ then $\{ e \in \bigcup \mathcal{F}_1 \mid \exists y_0 \subseteq x_0 .\ (y_0, e) \in x \}$ and $\{ e \in \bigcup \mathcal{F}_1 \mid \exists y_0 \subseteq x_0 .\ (y_0, e) \in x \}$ are compatible. From which it follows that $Pt\, x\, (x_0 \cap x_0') = Pt\, x\, (x_0) \cap Pt\, x\, (x_0')$, *i.e.* $Pt\, x$ is stable. Suppose $x, y \in \mathcal{F}_0 \to \mathcal{F}_1$, and $Pt\, x(x_0) = Pt\, y(x_0)$ for any $x_0 \in \mathcal{F}_0$. Let $(y_0, e_1) \in x$. Then $e_1 \in Pt\, x(y_0)$, and so $e_1 \in Pt\, y(y_0)$. This implies that, for some $y_0' \subseteq y_0$, $(y_0', e_1) \in y$. But now $e_1 \in Pt\, y(y_0')$; We must have $e_1 \in Pt\, x(y_0')$, too. Hence $y_0 = y_0'$. We have proved that $x \subseteq y$. By symmetry $y \subseteq x$, thus $x = y$.

Let $g \in [\mathcal{F}_0 \to_s \mathcal{F}_1]$. We show that there is a configuration $Cof\, g \in \mathcal{F}_0 \to \mathcal{F}_1$ such that $g = Pt(\, Cof\, g\,)$. $Cof\, g$ is defined as follows:

$$(y_0, e_1) \in Cof\, g \iff e_1 \in g(y_0) \ \& \ [y_0' \subseteq y_0 \ \& \ e_1 \in g(y_0') \Longrightarrow y_0 = y_0'].$$

It is easy to check that $Cof g$ is a member of $\mathcal{F}_0 \to \mathcal{F}_1$, and $g = Pt(\, Cof\, g\,)$. One can show further that, for $x, y \in \mathcal{F}_0 \to \mathcal{F}_1$, $x \subseteq y$ iff $Pt\, x \sqsubseteq_s Pt\, y$. ∎

7.2 Categories for Concurrency

Now we introduce the categories $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$ of Winskel. $\mathbf{SEV}^*_{syn}$ is a category with stable event structures as its objects and the *partially synchronous morphisms* as morphisms. $\mathbf{SEV}_{syn}$ is a category with stable event structures as its objects and the *synchronous morphisms* as morphisms.

Definition 7.2.1 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. A *partially synchronous morphism* from $\underline{E_0}$ to $\underline{E_1}$ is a partial function $\theta : E_0 \to E_1$ on on events which satisfies

- $X \in Con_0 \Longrightarrow \theta X \in Con_1$
- $\{ e,\ e' \} \in Con_0 \ \& \ \theta(e) = \theta(e') \Longrightarrow e = e'$
- $X \vdash_0 e \ \& \ \theta(e)$ is defined $\Longrightarrow \theta X \vdash_1 \theta(e)$

A partially synchronous morphism θ is *synchronous* if it is a total function.

Note the truth of $\theta(e) = \theta(e')$ asserts also that $\theta(e)$ and $\theta(e')$ are defined. $\mathbf{SEV}^*_{syn}$ is the category with stable event structures as objects and partially synchronous morphisms as morphisms. $\mathbf{SEV}_{syn}$ is the category with stable event structures as objects and partially synchronous morphisms as morphisms.

Definition 7.2.2 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. Their *partially synchronous product*, $\underline{E_0} \times \underline{E_1}$, is a structure $(E, Con, \vdash)$ where

- $E = \{(e, *) \mid e \in E_0\} \cup \{(*, e') \mid e' \in E_1\} \cup \{(e, e') \mid e \in E_0 \ \& \ e' \in E_1\}$
- $X \in Con \iff \pi_0 X \in Con_0 \ \& \ \pi_1 X \in Con_1 \ \&$

$$\forall e,\ e' \in X.\, [\pi_0(e) = \pi_0(e') \text{ or } \pi_1(e) = \pi_1(e')] \Rightarrow e = e'$$

- $X \vdash e \iff [\pi_0(e) \text{ is defined } \Rightarrow \pi_0 X \vdash_0 \pi_0(e)] \ \&$

$$[\pi_1(e) \text{ is defined } \Rightarrow \pi_1 X \vdash_1 \pi_1(e)].$$

An event $(e_0, *)$ is one which can occur independently of the events of $\underline{E_1}$. A pair of events (e_0, e_1) is understood as the synchronisation of the event e_0 from $\underline{E_0}$ and e_1 from $\underline{E_1}$. The consistent predicate for the partially synchronous product indicates that in any computation, an event cannot synchronise with two distinct events.

Theorem 7.2.1 The partially synchronous product with projections π_0 and π_1 is a product in the category $\mathbf{SEV}^*_{syn}$.

Definition 7.2.3 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. Their *synchronous product*, $\underline{E_0} \oplus \underline{E_1}$, is a structure $(E, Con, \vdash)$ where

- $E = \{(e, e') \mid e \in E_0 \ \& \ e' \in E_1\}$
- $X \in Con \iff \pi_0 X \in Con_0 \ \& \ \pi_1 X \in Con_1 \ \&$

$$\forall e,\ e' \in X.\, [\pi_0(e) = \pi_0(e') \text{ or } \pi_1(e) = \pi_1(e')] \Rightarrow e = e'$$

- $X \vdash e \iff (\pi_0 X \vdash_0 \pi_0(e) \ \& \ \pi_1 X \vdash_1 \pi_1(e))$.

Theorem 7.2.2 The synchronous product with projections π_0 and π_1 is a product in the category $\mathbf{SEV}_{syn}$.

Stable event structures can be used to give semantics to languages like CCS and

CSP, with parallel compositions are modeled by the partially synchronous product. The nondeterministic choice operator is modeled by the sum construction.

Definition 7.2.4 Let $\underline{E_0} = (E_0,\ Con_0,\ \vdash_0)$ and $\underline{E_1} = (E_1,\ Con_1,\ \vdash_1)$ be stable event structures. Their *sum*, $\underline{E_0} + \underline{E_1}$, is a structure $(E,\ Con,\ \vdash)$ where

- $E = \{(0, e) \mid e \in E_0\} \cup \{(1, e) \mid e \in E_1\}$
- $X \in Con \iff \exists X_0 \in Con_0.\, X = \{(0, e) \mid e \in X_0\}$ or

  $$\exists X_1 \in Con_1.\, X = \{(1, e) \mid e \in X_1\}$$
- $X \vdash e \iff [\exists X_0 \in Con_0,\ e_0 \in E_0.\ X_0 \vdash_0 e_0\ \&\ X = \{(0, e') \mid e' \in X_0\}\ \&\ e = (0, e_0)]$

  or

  $$[\exists X_1 \in Con_1,\ e_1 \in E_1.\ X_1 \vdash_1 e_1\ \&\ X = \{(1, e') \mid e' \in X_1\}\ \&\ e = (1, e_1)].$$

Theorem 7.2.3 The sum is a coproduct in both the categories $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$.

7.3 Monoidal Closed Categories

This section introduces monoidal closed categories $\mathbf{SEV}_l$, $\mathbf{COH}_l$, and $\mathbf{FF}$.

The Category $\mathbf{SEV}_l$

Tensor product and linear function space constructions are introduced on stable event structures. These constructions determine a monoidal closed category of stable event structures.

Definition 7.3.1 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. Their tensor product, $\underline{E_0} \otimes \underline{E_1}$, is defined as the structure $(E,\ Con,\ \vdash,\ )$ where

$$E = E_0 \times E_1$$

$$X \in Con \iff \pi_0 X \in Con_0\ \&\ \pi_1 X \in Con_1$$

$$X \vdash e \iff \exists X_0, X_1.\ X_0 \vdash_0 \pi_0 e\ \&\ X_1 \vdash_1 \pi_1 e\ \&\ X_0 \times X_1 \subseteq X.$$

Proposition 7.3.1 If $\underline{E_0}$ and $\underline{E_1}$ are stable then so is $\underline{E_0} \otimes \underline{E_1}$.

Proof We check the stability axiom. Assume $X \vdash e$, $Y \vdash e$, and $X \cup Y \cup \{e\} \in Con$. By definition, there exist X_0, X_1, Y_0, and Y_1, such that

$$\begin{array}{ll} X_0 \vdash_0 \pi_0 e, & X_1 \vdash_1 \pi_1 e, \\ Y_0 \vdash_0 \pi_0 e, & Y_1 \vdash_1 \pi_1 e, \\ X \supseteq X_0 \times X_1, & Y \supseteq Y_0 \times Y_1. \end{array}$$

Also we have $X_0 \cup Y_0 \cup \{\pi_0 e\} \in Con_0$ and $X_1 \cup Y_1 \cup \{\pi_1 e\} \in Con_1$. Therefore $X_0 \cap Y_0 \vdash_0 \pi_0 e$ and $X_1 \cap Y_1 \vdash_1 \pi_1 e$, by the stability of $\underline{E_0}$ and $\underline{E_1}$. However, $X \cap Y \supseteq (X_0 \cap Y_0) \times (X_1 \cap Y_1)$. Hence $X \cup Y \vdash e$. ∎

Let $x \in \mathcal{F}(\underline{E_0} \otimes \underline{E_1})$. We have $\pi_0 x \in \mathcal{F}(\underline{E_0})$. In fact, $\pi_0 x$ is clearly consistent and, if $e_0 \in \pi_0 x$, then there is some e_1 such that $(e_0, e_1) \in x$. The way enabling is defined for tensor product ensures that when (e_0, e_1) is secured in x, e_0 is secured in $\pi_0 x$. Hence $\pi_0 x$ is a configuration of $\underline{E_0}$. Similarly $\pi_1 x \in \mathcal{F}(\underline{E_1})$. From this we know that for any (e_0, e_1) in x, $\lceil e_0 \rceil_{\pi_0 x} \times \lceil e_1 \rceil_{\pi_1 x} \subseteq x$.

Suppose, on the other hand, that x is a subset of $E_0 \times E_1$ with the property that $\pi_0 x \in \mathcal{F}(\underline{E_0})$, $\pi_1 x \in \mathcal{F}(\underline{E_1})$, and

$$\forall (e_0, e_1) \in x \exists x_0 \in \mathcal{F}(\underline{E_0}),\ x_1 \in \mathcal{F}(\underline{E_1}).\ (e_0, e_1) \in x_0 \times x_1 \subseteq x.$$

x is obviously a consistent set. Also, since $e_0 \in x_0$ and $e_1 \in x_1$, $\lceil e_0 \rceil_{\pi_0 x} \times \lceil e_1 \rceil_{\pi_1 x} \subseteq x_0 \times x_1 \subseteq x$. Hence (e_0, e_1) is secured in x. In summary, we have proved that

Proposition 7.3.2 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. We have $x \in \mathcal{F}(\underline{E_0} \otimes \underline{E_1})$ iff

- $\pi_0 x \in \mathcal{F}(\underline{E_0})$ & $\pi_1 x \in \mathcal{F}(\underline{E_1})$
- $\forall (e_0, e_1) \in x \exists x_0 \in \mathcal{F}(\underline{E_0}),\ x_1 \in \mathcal{F}(\underline{E_1}).\ (e_0, e_1) \in x_0 \times x_1 \subseteq x$

Definition 7.3.2 Let $\underline{E_0} = (E_0, Con_0, \vdash_0)$ and $\underline{E_1} = (E_1, Con_1, \vdash_1)$ be stable event structures. Their *linear function space*, $\underline{E_0} \multimap \underline{E_1}$, is defined as the structure $(E, Con, \vdash,)$ where

$$E = \{(\lceil e_0 \rceil_x, e_1) \mid e_0 \in x \in \mathcal{F}(\underline{E_0}), e_1 \in E_1\}$$

$$X \in Con \text{ iff } \forall Y \subseteq X.\ \pi_0 Y \in Con_0 \Longrightarrow \pi_1 Y \in Con_1 \ \&$$

$$\forall a, b \in X.(\pi_0 a \uparrow \pi_0 b \ \&\ \pi_1 a = \pi_1 b) \Longrightarrow a = b$$

$$X \vdash (x, e_1) \text{ iff } \{e \mid (y, e) \in X \ \&\ y \subseteq x\} \vdash_1 e_1.$$

Note that $\lceil e \rceil_x$ is always finite.

Proposition 7.3.3 The linear function space of two stable event structures is a stable event structure. There is a $1-1$ order preserving correspondence between configurations $\mathcal{F}[\underline{E_0} \multimap \underline{E_1}]$ with set inclusion as its order and linear, stable functions $\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})$ with the stable order. The associated maps are

$$\phi : \mathcal{F}[\underline{E_0} \to \underline{E_1}] \to [\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})]$$

by letting

$$F \longmapsto \lambda x \in \mathcal{F}(\underline{E_0}).\ \{ e \in E_1 \mid \exists x' \subseteq x.\ (x', e) \in F \},$$

and

$$\mu : [\mathcal{F}(\underline{E_0}) \to \mathcal{F}(\underline{E_1})] \to \mathcal{F}[\underline{E_0} \to \underline{E_1}]$$

by letting

$$f \longmapsto \{ (\lceil e \rceil_x, e') \mid e' \in f(\lceil e \rceil_x) \ \&\ (\forall x' \subseteq \lceil e \rceil_x.\ e \in f(x') \Longrightarrow x' = \lceil e \rceil_x) \}.$$

Proof Similar to the proof of Theorem 7.1.3. ∎

Proposition 7.3.4 (Monoidal Closedness) Let $\underline{E_0}$, $\underline{E_1}$, and $\underline{E_2}$ be stable event structures. Then

$$(\underline{E_0} \otimes \underline{E_1}) \multimap \underline{E_2} \cong \underline{E_0} \multimap [\underline{E_1} \multimap \underline{E_2}].$$

The isomorphism $\underline{E} \cong \underline{E'}$ amounts to saying that there is a $1-1$ correspondence between E and E', which preserves the consistency and the entailment relations. Monoidal closedness suggests, usually, an underlying monoidal category. One can make this more precise by showing that the tensor product really induces a *monoidal* category which is *closed* [Mc71]. But doing so here would lead us too far astray.

Proof We show that the isomorphism is given by

$$\theta : (\lceil (e_0, e_1) \rceil_x,\ e_2) \longmapsto (\lceil e_0 \rceil_{\pi_0 x},\ (\lceil e_1 \rceil_{\pi_1 x},\ e_2)).$$

Clearly θ is onto as $x_0 \times x_1$ is a configuration in the tensor product if x_0 and x_1 are. It is then clear that θ is a bijection. Write

$$(\underline{E_0} \otimes \underline{E_1}) \multimap \underline{E_2} = (P, Con_P, \vdash_P),$$

$$\underline{E_0} \multimap [\underline{E_1} \multimap \underline{E_2}] = (F, Con_F, \vdash_F),$$

$$\underline{E_1} \multimap \underline{E_2} = (E, Con_E, \vdash_E).$$

We have, for a finite set I,

$\{ (\lceil (e_i, f_i) \rceil_{x_i}, g_i) \mid i \in I \} \in Con_P$

$\iff \forall J \subseteq I. \bigcup_{j \in J} \lceil (e_j, f_j) \rceil_{x_j} \in Con_{\underline{E_0} \otimes \underline{E_1}} \implies \{ g_j \mid j \in J \} \in Con_{\underline{E_2}}$ &

$\forall i, j \in I. (\lceil (e_i, f_i) \rceil_{x_i} \uparrow \lceil (e_j, f_j) \rceil_{x_j} \;\&\; g_i = g_j) \implies \lceil (e_i, f_i) \rceil_{x_i} = \lceil (e_j, f_j) \rceil_{x_j}$

$\iff \forall J \subseteq I. \bigcup_{j \in J} \lceil e_j \rceil_{\pi_0 x_j} \in Con_{\underline{E_0}} \bigcup_{j \in J} \lceil f_j \rceil_{\pi_1 x_j} \in Con_{\underline{E_1}} \implies \{ g_j \mid j \in J \} \in Con_{\underline{E_2}}$ &

$\forall i, j \in I. (\lceil e_i \rceil_{\pi_0 x_i} \uparrow \lceil e_j \rceil_{\pi_0 x_j} \;\&\; \lceil f_i \rceil_{\pi_1 x_i} \uparrow \lceil f_j \rceil_{\pi_1 x_j} \;\&\; g_i = g_j) \implies \lceil (e_i, f_i) \rceil_{x_i} = \lceil (e_j, f_j) \rceil_{x_j}$

$\iff \forall J \subseteq I. \bigcup_{j \in J} \lceil e_j \rceil_{\pi_0 x_j} \in Con_{\underline{E_0}} \implies \{ (\lceil f_j \rceil_{\pi_1 x_j}, g_j) \mid j \in J \} \in Con_{\underline{E_1} \multimap \underline{E_2}}$ &

$\forall i, j \in I. (\lceil e_i \rceil_{\pi_0 x_i} \uparrow \lceil e_j \rceil_{\pi_0 x_j} \;\&\; (\lceil f_i \rceil_{\pi_1 x_i}, g_i) = (\lceil f_j \rceil_{\pi_1 x_j}, g_j)) \implies \lceil e_i \rceil_{\pi_0 x_i} = \lceil e_j \rceil_{\pi_0 x_j}$

Thus $X \in Con_P$ iff $\theta X \in Con_F$.

Also

$$\{ (\lceil (e_i, f_i) \rceil_{x_i}, g_i) \mid i \in I \} \vdash_P (\lceil (e, f) \rceil_x, g)$$

$$\iff \{ g_i \mid i \in I \;\&\; \lceil (e_i, f_i) \rceil_{x_i} \subseteq \lceil (e, f) \rceil_x \} \vdash_E g$$

$$\iff \{ g_i \mid i \in I \;\&\; \lceil e_i \rceil_{\pi_0 x_i} \subseteq \lceil e \rceil_{\pi_0 x} \;\&\; \lceil f_i \rceil_{\pi_1 x_i} \subseteq \lceil f \rceil_{\pi_1 x} \} \vdash_E g$$

$$\iff \{ (\lceil f_i \rceil_{\pi_1 x_i}, g_i) \mid i \in I \;\&\; \lceil e_i \rceil_{\pi_0 x_i} \subseteq \lceil e \rceil_{\pi_0 x} \} \vdash_E (\lceil f \rceil_{\pi_1 x}, g)$$

Hence $X \vdash_P e$ iff $\theta X \vdash_F \theta e$. ∎

There is a corresponding monoidal closed category **DL** of dI-domains with linear, stable functions. It is a direct translation of the situation of the monoidal closed category $\mathbf{SEV}_l$ of stable event structures.

The Category $\mathbf{COH}_l$

Coherent families with linear, stable functions form a monoidal closed category $\mathbf{COH}_l$.

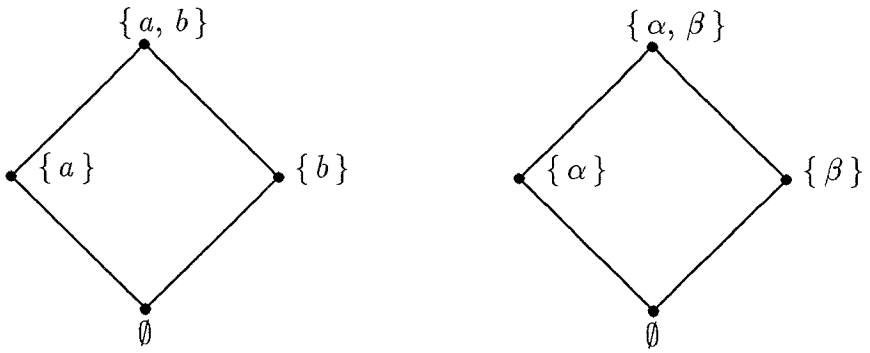
Definition 7.3.3 (Tensor Product) Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families. Their *tensor product*, $\mathcal{F}_0 \otimes \mathcal{F}_1$, is a family of subsets which satisfies

$$x \in \mathcal{F}_0 \otimes \mathcal{F}_1 \iff \bullet\ x \subseteq \bigcup \mathcal{F}_0 \times \bigcup \mathcal{F}_1 \;\&$$

$$\bullet\ \pi_0 x \in \mathcal{F}_0 \;\&\; \pi_1 x \in \mathcal{F}_1$$

Suppose $\mathcal{F}_0$ and $\mathcal{F}_1$ are coherent families and $y \subseteq x \in \mathcal{F}_0 \otimes \mathcal{F}_1$. Then clearly $y \subseteq \bigcup\mathcal{F}_0 \times \bigcup\mathcal{F}_1$ and $\pi_0 y \subseteq \pi_0 x$, $\pi_1 y \subseteq \pi_1 x$. Therefore $\pi_0 y \in \mathcal{F}_0$, $\pi_1 y \in \mathcal{F}_1$, and hence $y \in \mathcal{F}_0 \otimes \mathcal{F}_1$. This means $\mathcal{F}_0 \otimes \mathcal{F}_1$ is again a coherent family (the pairwise completeness is trivial).

Example 7.3.1 If not very careful one might think that product and tensor product produce isomorphic coherent families. Consider the product and tensor product of the following two coherent spaces.



$\{(a, \alpha),\ (a, \beta),\ (b, \alpha)\}$ and $\{(a, \alpha),\ (a, \beta),\ (b, \beta)\}$ are different elements in the tensor product; But they have the same projections on the first component and the second component. Thus tensor product has far more elements in it than product does.

There is the more simple construction of linear function space.

Definition 7.3.4 (Linear Function Space) Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent families. Their *linear function space*, $\mathcal{F}_0 \multimap \mathcal{F}_1$, is a family of subsets which satisfies

$$x \in \mathcal{F}_0 \multimap \mathcal{F}_1 \iff \bullet\ x \subseteq \bigcup\mathcal{F}_0 \times \bigcup\mathcal{F}_1\ \&$$

$$\bullet\ \forall y \subseteq x.\ \pi_0 y \in \mathcal{F}_0 \Longrightarrow \pi_1 y \in \mathcal{F}_1\ \&$$

$$\bullet\ [\{e_0, e_0'\} \in \mathcal{F}_0\ \&\ (e_0, e_1),\ (e_0', e_1) \in x] \Longrightarrow e_0 = e_0'$$

Any element x in $\mathcal{F}_0 \multimap \mathcal{F}_1$ determines a linear stable function $Pt\,x$, linear in the sense that

$$X \subseteq \mathcal{F}_0\ \&\ X\uparrow \Longrightarrow Pt\,x(\bigcup X) = \bigcup\{Pt\,x(x_0) \mid x_0 \in X\},$$

where

$$Pt\,x\,(x_0) = \{e_1 \in \bigcup\mathcal{F}_1 \mid \exists e_0 \in x_0.\ (e_0, e_1) \in x\}.$$

Linear function space is coherent, and is isomorphic to the linear functions from $\mathcal{F}_0$ to $\mathcal{F}_1$. The proof is very similar to the case of stable function space. Notice that for any

$x \in \mathcal{F}_0 \multimap \mathcal{F}_1$, $i(x) \in \mathcal{F}_0 \to \mathcal{F}_1$, where

$$i(x) = \{ (\{ e_0 \}, e_1) \mid (e_0, e_1) \in x \}.$$

$\mathbf{COH}_l$ is a monoidal closed category, with the tensor product and linear function space introduced above. In particular, we have the monoidal closedness, which says that for any $\mathcal{F}_0$ $\mathcal{F}_1$, and $\mathcal{F}_2$,

$$[\mathcal{F}_0 \otimes \mathcal{F}_1 \multimap \mathcal{F}_2] \cong [\mathcal{F}_0 \multimap [\mathcal{F}_1 \multimap \mathcal{F}_2]].$$

This is just a special case of Proposition 7.3.11 later.

The Category FF

The events of linear function space of coherent families consist of pairs of events (e_0, e_1) while the events of linear function space of stable event structures consist of pairs of the form $(\lceil e_0 \rceil_x, e_1)$. Constructions on coherent families are much simpler in general, but the expressive power of the coherent families is limited. For example, they cannot express the causality of events. Is there any category closely related to $\mathbf{COH}_l$ and $\mathbf{SEV}_l$, which is more expressive and yet simple, in particular, whose linear function space is built up of pairs of events (e_0, e_1)? This section answers the question positively. We hoped that such a category would induce a smipler logic, but it turned out to be not the case.

Definition 7.3.5 A *finitary family* is a set of subsets $\mathcal{F}$ of E which satisfies

- finite completeness:
  $X \subseteq \mathcal{F} \;\&\; X \uparrow^{fin} \Longrightarrow \bigcup X \in \mathcal{F}$
- finiteness:
  $\forall x \in \mathcal{F} \forall e \in x \exists z \in \mathcal{F}. (\mid z \mid < \infty \;\&\; e \in z \;\&\; z \subseteq x)$
- stability:
  $\forall X \subseteq \mathcal{F}. X \neq \emptyset \;\&\; X \uparrow \Longrightarrow \bigcap X \in \mathcal{F}$

$\emptyset$ is always in $\mathcal{F}$ because of finite completeness. The following proposition is obvious.

Proposition 7.3.5 Let $\mathcal{F}$ be a finitary family. $(\mathcal{F}, \subseteq)$ is a dI-domain.

Note that coincidence freeness is not necessary for $\mathcal{F}$ to be a domain under set inclusion.

Definition 7.3.6 Let $\mathcal{F}_0$, $\mathcal{F}_1$ be finitary families. $R : \mathcal{F}_0 \to \mathcal{F}_1$ is a *linear map* if $R \subseteq E_0 \times E_1$ is such that

- $x_0 \in \mathcal{F}_0 \Longrightarrow (\Delta R)x_0 \in \mathcal{F}_1$
- $(e_0,\ e_1), (e_0',\ e_1) \in R\ \&\ e_0,\ e_0' \in x_0 \Longrightarrow e_0 = e_0'$
- $\forall (e_0,\ e_1) \in R \exists R' \subseteq R.\ (e_0,\ e_1) \in R'\ \&\ \mid R' \mid < \infty\ \&\ (x_0 \in \mathcal{F}_0 \Longrightarrow (\Delta R')x_0 \in \mathcal{F}_1\ )$

where $(\Delta R)(x_0) = \{\ e_1 \mid \exists e_0 \in x_0.\ (e_0,\ e_1) \in R\ \}$.

The reader might wonder why we didn't choose a simpler definition of linear map as relation $R : \mathcal{F}_0 \to \mathcal{F}_1$ which satisfies

- $x_0 \in \mathcal{F}_0 \Longrightarrow (\Delta R)\ x_0 \in \mathcal{F}_1$
- $e_0,\ e_0' \in x_0\ \&\ (\ e_0,\ e_1\ ),\ (e_0',\ e_1\ ) \in R \Longrightarrow e_0 = e_0'$.

The reason is that finiteness might no longer hold if we use such a definition, as shown by the following example.

Example 7.3.2 Let

$$\mathcal{F}_0 = \{\ \{\ e,\ e_i\ \}\ \mid\ i \in \omega\ \} \cup \{\ \{\ e_i\ \}\ \mid\ i \in \omega\ \} \cup \{\ \emptyset\ \},$$

$$\mathcal{F}_1 = \{\ \emptyset,\ \{\ 2\ \},\ \{\ 1,\ 2\ \}\ \}.$$

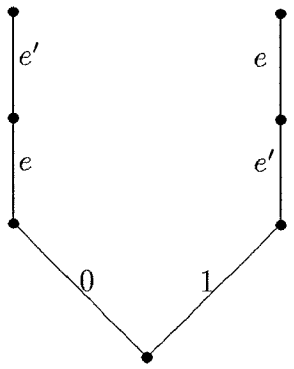
Obviously $\{\ (\ e,\ 1\ )\ \} \cup \{\ (\ e_i,\ 2\ )\ \mid\ i \in \omega\}$ is a well defined 'linear map' in this sense, but there does not exist a finite such linear map $R' \subseteq^{fin} R$ for which $(\ e,\ 1\ ) \in R'$.

A stable family has the property of coincidence-freeness. The other question the reader might ask is why we do not require it for finitary families. The reason is that it is not preserved by the construction of linear maps. Consider the following example:

Example 7.3.3 (Winskel) Let

$$\mathcal{F} = \{\ \{\ 0\ \},\ \{\ 1\ \},\ \{\ 0,\ e\ \},\ \{\ 1,\ e'\ \},\ \{\ 0,\ e,\ e'\ \},\ \{\ 1,\ e',\ e\ \}\ \}.$$

We can draw a picture of this family, using its events, as



This is a finitary family which is coincidence-free. However, the linear maps, $[\mathcal{F} \multimap \mathcal{F}]$, is *not*. Consider the identity linear map

$$Id_{\mathcal{F}} = \{ (0, 0), (1, 1), (e, e), (e', e') \}.$$

(e, e) and (e', e') are not separatable with respect to $Id_{\mathcal{F}}$ because for any $R \subseteq Id_{\mathcal{F}}$, if $(e, e) \in R$ but not $(e', e') \in R$ then R cannot be a linear map and similarly so when we switching the positions of (e, e) and (e', e'). In particular,

$$R = \{ (0, 0), (1, 1), (e, e) \}$$

is not a linear map since $\Delta R \{ 1, e, e' \} = \{ 1, e \} \notin \mathcal{F}$.

Proposition 7.3.6 If $R : \mathcal{F}_0 \to \mathcal{F}_1$ is a linear map, then $\Delta R : \mathcal{F}_0 \to \mathcal{F}_1$ is a linear, stable function.

Proof Easy.

It is also easy to show that $\overline{R} \sqsubseteq_s \overline{R'}$ iff $R \subseteq R'$. Note that the linear maps from $\mathcal{F}_0$ to $\mathcal{F}_1$ does not necessary represent all linear stable functions.

Proposition 7.3.7 Finitary families with linear maps form a category.

Proof The identity linear map is $\{ (e, e) \mid e \in \bigcup \mathcal{F} \}$ for any finitary family $\mathcal{F}$.

Let $R : \mathcal{F}_0 \to \mathcal{F}_1$ and $S : \mathcal{F}_1 \to \mathcal{F}_2$ be linear maps. Define $S \circ R : \mathcal{F}_0 \to \mathcal{F}_2$ to be the relational composition: $S \circ R = \{ (e_0, e_2) \mid \exists e. (e_0, e) \in R \ \& \ (e, e_2) \in S \}$. Suppose

$x_0 \in \mathcal{F}_0$.

$$\begin{aligned}
\Delta(S \circ R)\, x_0 &= \{\, e_2' \in \bigcup \mathcal{F}_2 \mid \exists e_0' \in x_0.\ (e_0',\ e_2') \in S \circ R \,\} \\
&= \{\, e_2' \in \bigcup \mathcal{F}_2 \mid \exists e_0' \in x_0 \exists e_1'.\ (e_0',\ e_1') \in R \ \&\ (e_1',\ e_2') \in S \,\} \\
&= \{\, e_2' \in \bigcup \mathcal{F}_2 \mid \exists e_1' \in (\Delta R)\, x_0.\ (e_1',\ e_2') \in S \,\} \\
&= (\Delta S)\,((\Delta R)\, x_0) \\
&\in \mathcal{F}_2.
\end{aligned}$$

Let $(e_0,\ e_2) \in S \circ R$. Then for some $e_1 \in \bigcup \mathcal{F}_1$, $(e_0,\ e_1) \in R$ and $(e_1,\ e_2) \in S$. Therefore, there exist $R' \subseteq R$ and $S' \subseteq S$, with R', S' finite, such that $x_0 \in \mathcal{F}_0 \Longrightarrow (\Delta R')\, x_0 \in \mathcal{F}_1$ and $x_1 \in \mathcal{F}_1 \Longrightarrow (\Delta S')\, x_1 \in \mathcal{F}_2$. It is then clear that $(e_0,\ e_2) \in S' \circ R' \subseteq S \circ R$, $S' \circ R'$ is finite, and $x_0 \in \mathcal{F}_0 \Longrightarrow \Delta(S' \circ R')\, x_0 \in \mathcal{F}_2$. Thus $S \circ R$ is a linear map. The associativity is easy. ∎

Definition 7.3.7 (The tensor product) Let $\mathcal{F}_0$ and $\mathcal{F}_1$ be finitary families. $\mathcal{F}_0 \otimes \mathcal{F}_1$, the tensor product of $\mathcal{F}_0$ and $\mathcal{F}_1$, is a family such that $x \in \mathcal{F}_0 \otimes \mathcal{F}_1$ iff

- $\pi_0 x \in \mathcal{F}_0$ & $\pi_1 x \in \mathcal{F}_1$
- $\forall (\, e_0,\ e_1\,) \in x \exists x_0 \in \mathcal{F}_0 \exists x_1 \in \mathcal{F}_1.\ (\, e_0 \in x_0$ & $e_1 \in x_1$ & $x_0 \times x_1 \subseteq x\,)$.

Proposition 7.3.8 If $\mathcal{F}_0$ and $\mathcal{F}_1$ are finitary families then so is $\mathcal{F}_0 \otimes \mathcal{F}_1$.

Proof Suppose $\mathcal{F}_0$ and $\mathcal{F}_1$ are finitary families. We show that $\mathcal{F}_0 \otimes \mathcal{F}_1$ is finitely complete, finitary, and stable.

Finitely completeness. Suppose $X \subseteq \mathcal{F}_0 \otimes \mathcal{F}_1$ and $X \uparrow^{fin}$. Then $\{\, \pi_0 x \mid x \in X \,\}$ and $\{\, \pi_1 x \mid x \in X \,\}$ are finitely compatible subsets of $\mathcal{F}_0$ and $\mathcal{F}_1$, respectively. Therefore $\bigcup_{x \in X} \pi_0 x \in \mathcal{F}_0$, $\bigcup_{x \in X} \pi_1 x \in \mathcal{F}_1$. Given $(\, e_0,\ e_1\,) \in \bigcup X$, there must be some $x \in X$ for which $(\, e_0,\ e_1\,) \in x$. By definition, therefore, $e_0 \in x_0$, $e_1 \in x_1$ for some $x_0 \in \mathcal{F}_0$, $x_1 \in \mathcal{F}_1$ with $x_0 \times x_1 \subseteq x$. Hence $x_0 \times x_1 \subseteq \bigcup X$. We have shown that $\bigcup X \in \mathcal{F}_0 \otimes \mathcal{F}_1$.

Finiteness. Let $x \in \mathcal{F}_0 \otimes \mathcal{F}_1$ and $(\, e_0,\ e_1\,) \in x$. Then there exist $x_0 \in \mathcal{F}_0$, $x_1 \in \mathcal{F}_1$ such that $e_0 \in x_0$, $e_1 \in x_1$, and $x_0 \times x_1 \subseteq x$. By the finiteness of $\mathcal{F}_0$, there exists $z_0 \in \mathcal{F}_0$ with $z_0 \subseteq x_0$, $e_0 \in z_0$, and $\mid z_0 \mid < \infty$. Similarly, there exists $z_1 \in \mathcal{F}_1$ with $z_1 \subseteq x_1$, $e_1 \in z_1$, and $\mid z_1 \mid < \infty$. We have, therefore, $(\, e_0,\ e_1\,) \in z_0 \times z_1 \subseteq x$ and $\mid z_0 \times z_1 \mid < \infty$. Also, it is obvious $z_0 \times z_1 \in \mathcal{F}_0 \otimes \mathcal{F}_1$.

Stability. Assume $X \subseteq \mathcal{F}_0 \otimes \mathcal{F}_1$, $X \neq \emptyset$, and $X \uparrow$ where $\mathcal{F}_0$ and $\mathcal{F}_1$ are finitary families. Clearly $\pi_0 X \uparrow$ and $\pi_1 X \uparrow$. By the stability of $\mathcal{F}_0$ and $\mathcal{F}_1$ we have $\bigcap \pi_0 X \in \mathcal{F}_0$, $\bigcap \pi_1 X \in \mathcal{F}_1$, i.e. $\pi_0 \bigcap X \in \mathcal{F}_0$, $\pi_1 \bigcap X \in \mathcal{F}_1$. Furthermore, if $(e_0,\ e_1) \in \bigcap X$ then $(e_0,\ e_1) \in x$ for all $x \in X$. Thus for any $x \in X$ there exist $u_x \in \mathcal{F}_0$, $v_x \in \mathcal{F}_1$ where $e_0 \in u_x$, $e_1 \in v_x$, and $u_x \times v_x \subseteq x$. From this we can see that e_0 must be in $\bigcap_{x \in X} u_x$ and e_1 in $\bigcap_{x \in X} v_x$. It is obvious that $\bigcap_{x \in X} u_x \times \bigcap_{x \in X} v_x \subseteq y$ for any $y \in X$. Therefore $\bigcap_{x \in X} u_x \times \bigcap_{x \in X} v_x \subseteq \bigcap X$. ∎

It is easy to see that $x_0 \times x_1 \in \mathcal{F}_0 \otimes \mathcal{F}_1$ for any $x_0 \in \mathcal{F}_0$ and any $x_1 \in \mathcal{F}_1$. From this one immediately conclude that the tensor product preserves fullness.

Proposition 7.3.9 Let $\mathcal{F}_0$ and $\mathcal{F}_1$ be finitary families. Then

$$x = \bigcup_{i \in I} u_i \times v_i \text{ iff } x \in \mathcal{F}_0 \otimes \mathcal{F}_1,$$

where $u_i \in \mathcal{F}_0$, $v_i \in \mathcal{F}_1$ for all $i \in I$ (I is not necessary finite).

Proof Suppose $x \in \mathcal{F}_0 \otimes \mathcal{F}_1$, where $\mathcal{F}_0$ and $\mathcal{F}_1$ are finitary families. By definition for each $(e_0,\ e_1) \in x$ there is $u_{(e_0, e_1)} \in \mathcal{F}_0$, $v_{(e_0, e_1)} \in \mathcal{F}_1$ such that $e_0 \in u_{(e_0, e_1)}$, $e_1 \in v_{(e_0, e_1)}$, and $u_{(e_0, e_1)} \times v_{(e_0, e_1)} \subseteq x$. It is clear that we have

$$x = \bigcup_{(e_0,\ e_1) \in x} u_{(e_0, e_1)} \times v_{(e_0, e_1)}.$$

The other half of the proof is trivial. ∎

Definition 7.3.8 Write $[\mathcal{F}_0 \multimap \mathcal{F}_1]$ for the collection of the linear maps $R : \mathcal{F}_0 \to \mathcal{F}_1$.

Proposition 7.3.10 $[\mathcal{F}_0 \multimap \mathcal{F}_1]$ is a finitary family.

Proof Finite completeness. Suppose $X \subseteq [\mathcal{F}_0 \multimap \mathcal{F}_1]$ is a finitary compatible subset. We want to show that $\bigcup X$ is a linear map. Let $x_0 \in \mathcal{F}_0$. Clearly

$$\Delta(\bigcup X)\ x_0 = \bigcup \{\ \Delta R\ x_0 \mid R \in X\ \}$$

and $\{\ (\Delta R)\, x_0 \mid R \in X\ \}$ is finitary compatible. So $\bigcup \{\ (\Delta R)\, x_0 \mid R \in X\ \} \in \mathcal{F}_1$. Let $(e_0,\ e_1)$, $(e_0',\ e_1) \in \bigcup X$ and $e_0,\ e_0' \in x_0$ for some $x_0 \in \mathcal{F}_0$. Then $(e_0,\ e_1) \in R_0$, $(e_0',\ e_1) \in R_1$ for some $R_0,\ R_1 \in X$. Since X is finitary compatible, there is a $R \in X$ which dominates both R_0 and R_1, i.e. $R_0 \subseteq R$ and $R_1 \subseteq R$. Hence $e_0 = e_0'$, by the definition of linear maps.

The finiteness of $[\mathcal{F}_0 \multimap \mathcal{F}_1]$ is obvious. Let us check stability. Suppose $X \subseteq [\mathcal{F}_0 \multimap \mathcal{F}_1]$ is a non-empty, compatible subset. It is easy to see that, for any $x_0 \in \mathcal{F}_0$,

$$(\Delta[\bigcap X])\, x_0 \subseteq \bigcap\{\, (\Delta R)\, x_0 \mid R \in X \,\}.$$

On the other hand,

$$\begin{aligned}
e_1 \ &\in \bigcap\{\, (\Delta R)\, x_0 \mid R \in X \,\} \\
&\Longrightarrow \forall R \in X \exists e_0^R \in x_0.\ (e_0^R,\ e_1) \in R \\
&\Longrightarrow \forall R \in X.\ (e_0^R,\ e_1) \in \bigcup X \\
&\Longrightarrow \forall R, S \in X.\ e_0^R = e_0^S \qquad (\ \bigcup X \text{ is a linear map }) \\
&\Longrightarrow \forall S \in X.\ (e_0^R,\ e_1) \in S \\
&\Longrightarrow (e_0^R,\ e_1) \in \bigcap X \\
&\Longrightarrow e_1 \in (\Delta[\bigcap X])\, x_0
\end{aligned}$$

Therefore

$$(\Delta[\bigcap X])\, x_0 = \bigcap\{\, (\Delta R)\, x_0 \mid R \in X \,\},$$

which implies $\Delta(\bigcap X)\, x_0 \in \mathcal{F}_1$ since $\{\, (\Delta R)\, x_0 \mid R \in X \,\}$ is a non-empty, compatible family of $\mathcal{F}_1$ and $\bigcap\{\, Rx_0 \mid R \in X \,\} \in \mathcal{F}_1$. It is trivial that $(e_0,\ e_1)$, $(e_0',\ e_1) \in \bigcap X$ and $e_0,\ e_0' \in x_0$ implies $e_0 = e_0'$. As each $R \in X$ is a linear map we have, for each $(e_0,\ e_1) \in \bigcap X$ and each $R \in X$, some finite, linear map $S_{R,e_0,e_1} \subseteq R$ which contains $(e_0,\ e_1)$. We have also $(e_0,\ e_1) \in \bigcap\{\, S_{R,e_0,e_1} \mid R \in X \,\}$ and, similar to the proof above,

$$\forall x_0 \in \mathcal{F}_0.\ (\Delta[\bigcap\{\, S_{R,e_0,e_1} \mid R \in X \,\}])\, x_0 \in \mathcal{F}_1.$$

∎

Example 7.3.2 shows that he construction of linear maps does not preserve fullness. However, this is not a problem since we can remove those events which never appear in any configuration and get a full finitary family of linear maps.

Proposition 7.3.11 (Monodial closedness) For finitary families $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$,

$$[\mathcal{F}_0 \otimes \mathcal{F}_1 \multimap \mathcal{F}_2] \cong [\mathcal{F}_0 \multimap [\mathcal{F}_1 \multimap \mathcal{F}_2]],$$

where the isomorphism is given by

$$\theta :\ (\,(\,e_0,\ e_1\,),\ e_2\,) \longmapsto (\,e_0,\ (\,e_1\ e_2\,)\,).$$

Note the effect of θ is simply regrouping the brackets.

Proof Let $z \in [\mathcal{F}_0 \otimes \mathcal{F}_1 \multimap \mathcal{F}_2]$. We show that $\theta z \in [\mathcal{F}_0 \multimap [\mathcal{F}_1 \multimap \mathcal{F}_2]]$. Given $x_0 \in \mathcal{F}_0$, we have

$$(\Delta\theta z)\, x_0 = \{\, (e_1, e_2) \mid ((e_0, e_1), e_2) \in z \;\&\; e_0 \in x_0 \,\}.$$

Thus for any $x_1 \in \mathcal{F}_1$,

$$(\Delta(\Delta\theta z)\, x_0)\, x_1 = \{\, e_2 \mid ((e_0, e_1), e_2) \in z \;\&\; e_0 \in x_0 \;\&\; e_1 \in x_1 \,\}.$$

However,

$$\{\, e_2 \mid ((e_0, e_1), e_2) \in z \;\&\; e_0 \in x_0 \;\&\; e_1 \in x_1 \,\} = (\Delta z)\,(x_0 \times x_1)$$

and $x_0 \times x_1 \in \mathcal{F}_0 \otimes \mathcal{F}_1$. Therefore $(\Delta z)\,(x_0 \times x_1) \in \mathcal{F}_2$, from which it follows that $(\Delta(\Delta\theta z)\, x_0)\, x_1 \in \mathcal{F}_2$. Suppose (e_1, e_2), $(e_1', e_2) \in (\Delta\theta z)\, x_0$ and $e_1, e_1' \in x_1 \in \mathcal{F}_1$. This implies the existence of $e_0,\ e_0' \in x_0 \in \mathcal{F}_0$ such that $((e_0,\ e_1),\ e_2) \in z$ and $((e_0',\ e_1'),\ e_2) \in z$. Accordingly $e_0 = e_0'$, $e_1 = e_1'$ since z is a linear map and

$$(e_0,\ e_1), (e_0',\ e_1') \in x_0 \times x_1 \;\&\; x_0 \times x_1 \in \mathcal{F}_0 \otimes \mathcal{F}_1.$$

Now let $(e_1, e_2) \in (\Delta\theta z)\, x_0$. There is some $e_0 \in z$ for which $((e_0,\ e_1),\ e_2) \in z$. For z is a linear map, there is a linear map $z' \subseteq^{fin} z$ such that $((e_0,\ e_1),\ e_2) \in z'$. Therefore $(e_1,\ e_2) \in (\Delta\theta z')\, x_0 \subseteq^{fin} (\Delta\theta z)\, x_0$ where $(\Delta\theta z')\, x_0$, similar to $(\Delta\theta z)\, x_0$, has the property that $\forall x_1 \in \mathcal{F}_1$, $(\Delta(\Delta\theta z')\, x_0)\, x_1 \in \mathcal{F}_2$. We have shown that $(\Delta\theta z)\, x_0 \in [\mathcal{F}_1 \multimap \mathcal{F}_2]$.

Assume $(e_0,\ (e_1,\ e_2))$, $(e_0',\ (e_1,\ e_2)) \in \theta z$ and $e_0,\ e_0' \in x_0$ for $x_0 \in \mathcal{F}_0$. Then $((e_0,\ e_1),\ e_2)$, $((e_0',\ e_1),\ e_2) \in z$. As $\mathcal{F}_1$ is full, there is $x_1 \in \mathcal{F}_1$ with $e_1 \in x_1$. So $e_0 = e_0'$ since (e_0, e_1), $(e_0',\ e_1) \in x_0 \times x_1 \in \mathcal{F}_0 \otimes \mathcal{F}_1$. If $(e_0,\ (e_1,\ e_2)) \in \theta z$ then $((e_0,\ e_1),\ e_2) \in z$. This implies there is some linear map $z' \subseteq^{fin} z$ for which $((e_0,\ e_1),\ e_2) \in z'$. Hence $(e_0,\ (e_1,\ e_2)) \in \theta z' \subseteq^{fin} \theta z$, where $(\Delta\theta z')\, x_1 \in \mathcal{F}_2$ for any $x_1 \in \mathcal{F}_1$, by a similar argument given above.

On the other hand, let $y \in [\mathcal{F}_0 \multimap [\mathcal{F}_1 \multimap \mathcal{F}_2]]$. It is to be shown that $\theta^{-1}y \in [\mathcal{F}_0 \otimes \mathcal{F}_1 \multimap \mathcal{F}_2]$. To this end let $x \in \mathcal{F}_0 \otimes \mathcal{F}_1$. Obviously

$$(\Delta\theta^{-1}y)\, x = \{\, e_2 \mid (e_0, (e_1, e_2)) \in y,\ (e_0, e_1) \in x \,\}.$$

By Proposition 7.3.5 , $x = \bigcup_{i\in I} u_i \times v_i$ where $u_i \in \mathcal{F}_0$, $v_i \in \mathcal{F}_1$ for $i \in I$. Hence

$$\begin{aligned}(\Delta\,\theta^{-1}\,y\,)\,x &= (\,\Delta\,\theta^{-1}\,y\,)(\,\bigcup_{i\in I} u_i \times v_i\,)\\ &= \bigcup_{i\in I}(\,\Delta\,\theta^{-1}\,y\,)\,u_i \times v_i\\ &= \bigcup_{i\in I}\{\,e_2 \mid (\,e_0,\,(\,e_1,\,e_2\,)\,) \in y,\ (\,e_0,\,e_1\,) \in u_i \times v_i\,\}\\ &= \bigcup_{i\in I}(\,\Delta(\Delta y)\,u_i)\,v_i\end{aligned}$$

However, we have, for each i, $(\,\Delta(\Delta y)\,u_i)\,v_i \subseteq (\,\Delta(\Delta y)\,\pi_0 x)\,\pi_1 x$ and $(\,\Delta(\Delta y)\,u_i)\,v_i \in \mathcal{F}_2$. So $\bigcup_{i\in I}(\,\Delta(\Delta y)\,u_i)\,v_i \in \mathcal{F}_2$, which means $\Delta(\theta^{-1}y\,)\,x \in \mathcal{F}_2$.

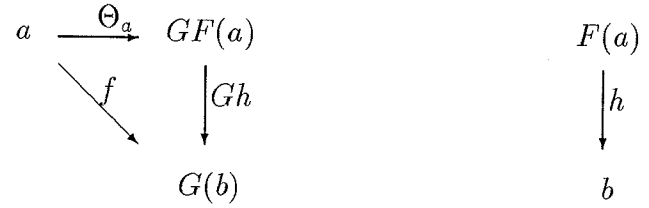
Assume $(\,(\,e_0,\,e_1\,),\,e_2\,)$, $(\,(\,e_0',\,e_1'\,),\,e_2\,) \in \theta^{-1}y$ and $(\,e_0,\,e_1\,)$, $(\,e_0',\,e_1'\,) \in x \in \mathcal{F}_0 \otimes \mathcal{F}_1$. Then $(\,e_0,\,(\,e_1,\,e_2\,)\,)$, $(\,e_0',\,(\,e_1',\,e_2\,)\,) \in y$ and $e_0,\ e_0' \in \pi_0 x \in \mathcal{F}_0$, $e_1,\ e_1' \in \pi_1 x \in \mathcal{F}_1$. Furthermore, $(e_1,\,e_2\,)$, $(e_1',\,e_2\,) \in \Delta(y)\,\pi_0 x$ and $e_1,\ e_1' \in \pi_1 x \in \mathcal{F}_1$. Therefore $e_1 = e_1'$ since $\Delta(y)\,\pi_0 x$ is a linear map. As a consequence $e_0 = e_0'$.

If $(\,(\,e_0,\,e_1\,),\,e_2\,) \in \theta^{-1}y$ then $(\,e_0,\,(\,e_1,\,e_2\,)\,) \in y$. We have, for some $y' \subseteq^{fin} y$, $(\,e_0,\,(\,e_1,\,e_2\,)\,) \in y'$ where y' is a linear map. Therefore $(\,(\,e_0,\,e_1\,),\,e_2\,) \in \theta^{-1}y'$, which is a finite subset of $\theta^{-1}y$. For this y' it is also true that $\Delta(\,\theta^{-1}y'\,)\,x \in \mathcal{F}_2$ for any $x \in \mathcal{F}_0 \otimes \mathcal{F}_1$. ∎

7.4 Relationships among the Categories

It is known that **DI**, $\mathbf{SEV}_s$, and $\mathbf{SF}_s$ are equivalent categories. It is also clear that **DL**, $\mathbf{SEV}_l$, and $\mathbf{SF}_l$ are equivalent categories. Apparently $\mathbf{SEV}_{syn}$ is a subcategory of $\mathbf{SEV}^*_{syn}$, which is a subcategory of $\mathbf{SEV}_l$. Again, $\mathbf{SEV}_l$ is a subcategory of $\mathbf{SEV}_s$. $\mathbf{COH}_l$ is a full-subcategory of $\mathbf{SF}_l$; $\mathbf{COH}_s$ is a full-subcategory of $\mathbf{SF}_s$. There are more interesting relationships between some of the categories. In this section we show that there is an adjunction between $\mathbf{COH}_l$ and $\mathbf{COH}_s$. We also show that there is a coreflection between **DL** and **FF**.

Recall one of the ways of determining an adjunction between two categories **A** and **B**. Two functors $F : \mathbf{A} \to \mathbf{B}$, $G : \mathbf{B} \to \mathbf{A}$ is an adjunction pair if for any object a of **A**, there is a morphism $\Theta_a : a \to GF(a)$ in **A** which is *universal* in the sense explained as follows. $\Theta_a : a \to GF(a)$ in **A** is universal if for any morphism $f : a \to G(b)$ in **A** with b in **B** there is a unique morphism $h : F(a) \to b$ in **B** such that the following diagram commutes:



In this situation we say F is the left adjoint of G and G is a right adjoint of F. When for each $a \in \mathbf{A}$, Θ_a is an isomorphism, then the adjunction is called a *coreflection*.

First consider the relationship between $\mathbf{COH}_l$ and $\mathbf{COH}_s$. There is the following construction on coherent families.

Definition 7.4.1 (Shriek) Let $\mathcal{F}$ be a coherent family. Its *shriek*, $!\mathcal{F}$, is a family of subsets which satisfies

$$x \in !\mathcal{F} \iff x \subseteq \{\, a \mid a \in \mathcal{F} \;\&\; |\, a \,| < \infty \,\} \;\&\; \bigcup x \in \mathcal{F}$$

It is obvious that if $\mathcal{F}$ is a coherent family then $!\mathcal{F}$ is still a coherent family. The requirement that x consists of the *finite* configurations of $\mathcal{F}$ ensures that there are countably many events in $!\mathcal{F}$.

Clearly the inclusion i is a functor form $\mathbf{COH}_l$ to $\mathbf{COH}_s$. It has a left adjoint !, which sends a morphism $\mathcal{F}_0 \xrightarrow{g} \mathcal{F}_1$ in $\mathbf{COH}_s$ to a morphism $!\mathcal{F}_0 \xrightarrow{!g} !\mathcal{F}_1$ in $\mathbf{COH}_l$, where ! is the shriek operation on coherent families and

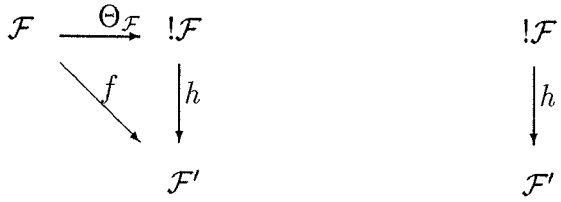
$$!g = Pt(\{\, (x,\, y) \mid |\, x \,| < \infty \;\&\; |\, y \,| < \infty \;\&\; y \subseteq g(x) \;\&\; \forall x' \subseteq x.\, y \subseteq g(x') \Longrightarrow x = x' \,\}).$$

for a stable function g. $\Theta_{\mathcal{F}} : \mathcal{F} \to !\mathcal{F}$ is given by $\Theta_{\mathcal{F}}(x) = \{\, y \mid y \subseteq^{fin} x \,\}$. By inspecting the relevant axioms one can convince himself that ! is indeed a functor.

We check that $\Theta_{\mathcal{F}} : \mathcal{F} \to !\mathcal{F}$ is universal. For any morphism $f : \mathcal{F} \to \mathcal{F}'$ in $\mathbf{COH}_s$, $h : !\mathcal{F} \to \mathcal{F}'$ is a morphism in $\mathbf{COH}_l$, where

$$h = Pt(\{\, (x,\, e) \in \bigcup !\mathcal{F} \times \bigcup \mathcal{F}' \mid (x,\, e) \in Cof\, f \,\}).$$

Note that x has two different types in the definition of h. The first x is considered as an event of $!\mathcal{F}$, the second is considered as a configuration in $\mathcal{F}$. Clearly $h \circ \Theta_{\mathcal{F}} = f$, and such h is unique because it is linear. Thus the following diagram gives the universal property of $\Theta_{\mathcal{F}} : \mathcal{F} \to !\mathcal{F}$ where we omitted the inclusion functor i.



We have already known that any stable function $f : \mathcal{F} \to \mathcal{F}'$ gives a linear function $Pt(\{(x, e) \in \bigcup !\mathcal{F} \times \bigcup \mathcal{F}' \mid (x, e) \in Cof\, g\})$ form $!\mathcal{F}$ to $\mathcal{F}'$. On the other hand, assume that $h : !\mathcal{F} \to \mathcal{F}'$ is a morphism in $\mathbf{COH}_l$. Then $Cof\, h \in !\mathcal{F} \multimap \mathcal{F}'$. Let

$$H = Pt(\{(x, e) \mid (\{x\}, e) \in Cof\, h\}).$$

It is easy to check that $\{(x, e) \mid (\{x\}, e) \in Cof\, h\}$ is a configuration in $\mathcal{F} \to \mathcal{F}'$, and

$$!\mathcal{F} \multimap \mathcal{F}' \cong \mathcal{F} \to \mathcal{F}'.$$

In summary, we have proved that

Theorem 7.4.1 $\mathbf{COH}_l \xrightarrow{i} \mathbf{COH}_s \xrightarrow{!} \mathbf{COH}_l$ determines an adjunction, with ! the left adjoint of i.

Now let us consider the relationship between **FF** and **DL**. Let Δ be a functor from **FF** to **DL** defined by

$$\begin{array}{ccc} \mathcal{F}_0 & \xrightarrow{\mathrm{R}} & \mathcal{F}_1 \\ \Big\downarrow & & \\ \Delta(\mathcal{F}_0) & \xrightarrow{\Delta(R)} & \Delta(\mathcal{F}_1) \end{array}$$

where $\Delta(\mathcal{F}_0)$ and $\Delta(\mathcal{F}_1)$ are dI-domains determined by $\mathcal{F}_0$ and $\mathcal{F}_1$ (Proposition 7.3.5), $\Delta(R)$ is the linear stable function determined by the linear map R (Definition 7.3.2).

There is also a functor Φ in the other direction $\mathbf{DL} \to \mathbf{FF}$ defined by

$$\begin{array}{ccc} D_0 & \xrightarrow{\mathrm{f}} & D_1 \\ \Big\downarrow & & \\ \Phi(D_0) & \xrightarrow{\Phi(f)} & \Phi(D_1) \end{array}$$

where $\Phi(D_0)$ (and similarly $\Phi(D_1)$) is a family of subsets of D_0^1, with D_0^1 the set of complete primes of D_0, such that $x \in \Phi(D_0)$ iff x is left closed (with respect to D_0^1) and pairwise compatible. $\Phi(f)$ is given after the following proposition.

Proposition 7.4.1 $\Phi(D)$ is a finitary family if D is a dI-domain.

Proof Finite completeness is straightforward.

Finiteness: Let $x \in \Phi(D)$ and $p \in x$. As p is a complete prime, $\{p' \mid p' \sqsubseteq p \ \& \ p' \in D^1\}$ is a finite set of complete primes. Clearly $\{p' \mid p' \sqsubseteq p \ \& \ p' \in D^1\} \in \Phi(D)$, $\{p' \mid p' \sqsubseteq p \ \& \ p' \in D^1\} \subseteq x$, and $p \in \{p' \mid p' \sqsubseteq p \ \& \ p' \in D^1\}$.

Stability: Suppose X is a non-empty, compatible subset of $\Phi(D)$. Clearly any two members of $\bigcap X$ are compatible and $\bigcap X$ is left closed because each member of X is. ∎

Proposition 7.4.2 Given $f : D \to E$, a linear stable function between dI-domains D, E. Define $\Phi f \subseteq D^1 \times E^1$ to be a subset such that $(p, q) \in \Phi f$ iff $f(p) \sqsupseteq q$ and $\forall p' \sqsubseteq p.\ (f(p') \sqsupseteq q \Longrightarrow p' = p)$. Then $\Phi f : \Phi(D) \to \Phi(E)$ is a linear map.

Proof Let $x \in \Phi D$. $(\Phi f)x = \{q \mid \exists p \in x.\ (p, q) \in \Phi f\}$. We have

$$\begin{aligned} q, q' \in (\Phi f)\, x &\Longrightarrow \exists p, p' \in x.\ (p, q), (p', q') \in \Phi f \\ &\Longrightarrow p \uparrow p' \\ &\Longrightarrow f(p) \uparrow f(p') \\ &\Longrightarrow q \uparrow q'. \end{aligned}$$

Let $q \sqsubseteq q' \in (\Phi f)x$, where q is a complete prime. There must be some $p' \in x$ such that $(p', q') \in \Phi f$. This means $f(p') \sqsupseteq q'$ and $\forall p \sqsubseteq p'.\ f(p) \sqsupseteq q' \Longrightarrow p = p'$. Let

$$p_0 = \bigsqcap \{p \mid p \sqsubseteq p' \ \& \ f(p) \sqsupseteq q\}.$$

We have

$$\begin{aligned} f(p_0) &= f(\bigsqcap \{p \mid p \sqsubseteq p' \ \& \ f(p) \sqsupseteq q\}) \\ &= \bigsqcap \{f(p) \mid p \sqsubseteq p' \ \& \ f(p) \sqsupseteq q\} \\ &\sqsupseteq q. \end{aligned}$$

Clearly $\forall p \sqsubseteq p_0.\ (f(p) \sqsupseteq q \Longrightarrow p = p_0)$. To check that p_0 is a complete prime suppose

$p_0 = \bigsqcup_{i=1}^{n} p_i$ where p_i's are complete prime.

$$\begin{aligned} f(p_0) \sqsupseteq q &\Longrightarrow f(\bigsqcup_{i=1}^{n} p_i) \sqsupseteq q \\ &\Longrightarrow \bigsqcup_{i=1}^{n} f(p_i) \sqsupseteq q \\ &\Longrightarrow \exists i.\ f(p_i) \sqsupseteq q \\ &\Longrightarrow p_i = p_0 \end{aligned}$$

Hence $(p_0, q) \in \Phi f$. Also $p_0 \sqsubseteq p'$ since $f(p') \sqsupseteq q' \sqsupseteq q$, which implies $p_0 \in x$. Therefore $q \in (\Phi f)x$, and so $(\Phi f)x \in \Phi E$.

If (p, q), $(p', q) \in \Phi f$ and $p,\ p' \in x$ then $p \uparrow p'$ and so $f(p \sqcap p') = f(p) \sqcap f(p') \sqsupseteq q$ by the stability of f. We have $p \sqcap p' = \bigsqcup_{i=1}^{n} p_i$ where p_i's are complete primes. Hence $f(p_i) \sqsupseteq q$ for some i, since q is a complete prime. So $p_i = p = p'$.

For any $(p, q) \in \Phi f$, consider Φf_p^q where $f_p^q = \lambda t.\, f(t \sqcap p) \sqcap q$. Similar to the proof given above we have $(\Phi f_p^q)x \in \Phi E$ for any $x \in \Phi D$. It is easy to see that $(p, q) \in \Phi f_p^q$. Let $(p', q') \in \Phi f_p^q$, i.e. $f_p^q(p') \sqsupseteq q'$ and $\forall p'' \sqsubseteq p'.\ f_p^q(p'') \sqsupseteq q' \Longrightarrow p'' = p'$. We have $f(p' \sqcap p) \sqcap q \sqsupseteq q'$ and $\forall p'' \sqsubseteq p'.\ f(p'' \sqcap p) \sqcap q \sqsupseteq q' \Longrightarrow p'' = p'$, which implies $f(p') \sqsupseteq q'$. Also, $p'' \sqsubseteq p'$ and $f(p'') \sqsupseteq q'$ implies

$$\begin{aligned} & f(p'' \sqcap p) \sqcap q \\ &= f(p'' \sqcap (p' \sqcap p)) \sqcap q \\ &= f(p'') \sqcap f(p' \sqcap p) \sqcap q \ \ (\text{ since f is stable and } p'' \uparrow (p' \sqcap p)) \\ &\sqsupseteq q' \sqcap q' \\ &= q'. \end{aligned}$$

Therefore $p'' = p'$ and so $(p', q') \in \Phi f$. Finally we check that Φf_p^q is a finite set. Clearly $q \sqsupseteq q'$ for each $(p', q') \in \Phi f_p^q$. Taking $p'' = p' \sqcap p$, we have $p'' \sqsubseteq p'$ and $f(p'' \sqcap p) \sqcap q \sqsupseteq f(p' \sqcap p) \sqcap q \sqsupseteq q'$. Therefore $p'' = p'$, i.e. $p' \sqsubseteq p$. As there are only finitely many complete primes below q and all the p''s are below p (hence compatible), we conclude that Φf_p^q is finite. ∎

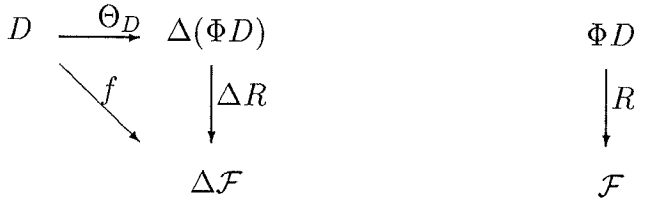
Proposition 7.4.3 For any dI-domain D, $D \cong \Delta(\Phi D)$.

Proof Define $\Theta_D : D \to \Delta(\Phi D)$ by letting $\Theta_D(d) = \{ p \in D^1 \mid p \sqsubseteq d \}$ for $d \in D$ and $\theta_D : \Delta(\Phi D) \to D$ by $\theta_D(x) = \bigsqcup x$. It is trivial that $\Theta_D \theta_D = \mathbf{1}_{\Delta(\Phi D)}$ and $\theta_D \Theta_D = \mathbf{1}_D$.

∎

Theorem 7.4.2 $\Phi : \mathbf{DL} \to \mathbf{FF}$ and $\Delta : \mathbf{FF} \to \mathbf{DL}$ is a coreflection pair.

Proof Given $D \in \mathbf{DL}$, $\mathcal{F} \in \mathbf{FF}$ and an arrow $f : D \to \Delta\mathcal{F}$ of $\mathbf{DL}$, we need to show that there exists a unique $R : \Phi D \to \mathcal{F}$ which makes the following diagram commute:



Existence: Define $R : \Phi D \to \mathcal{F}$ to be such that $(p, e) \in R$ iff

$$\exists x \in \mathcal{F}.\ e \in x \& f(p) \sqsupseteq \lceil e \rceil_x \& (p' \sqsubseteq p \ \& \ f(p') \sqsupseteq \lceil e \rceil_x \Longrightarrow p = p').$$

We show that R is a linear map. Let $u \in \Phi D$, $\Delta(R)\, u = \{ e \mid \exists p \in u.\ (p, e) \in R \}$. To show $\Delta(R)\, u \in \mathcal{F}$ it is enough to prove that

$$\{ e \mid \exists p \in u.\ (p, e) \in R \} = \bigcup \{ \lceil e \rceil_{x_e} \mid \exists p \in u.\ (p, e) \in R \}$$

(x_e's are those x's appear in the definition of R) since the right hand side of the above equation is finitely compatible, using the compatibility of u. Suppose $e \in \Delta(R)\, u$. Then $\exists p \in u.\ (p, e) \in R$, or $\exists x_e \in \mathcal{F}.\ e \in x_e \& f(p) \sqsupseteq \lceil e \rceil_{x_e} \& (p' \sqsubseteq p \& f(p') \sqsupseteq \lceil e \rceil_{x_e} \Longrightarrow p = p')$. Let $e' \leq_{x_e} e$. Clearly $f(p) \sqsupseteq \lceil e' \rceil_{x_e}$. Let

$$p_0 = \bigsqcap \{ p'' \mid p'' \sqsubseteq p \& \ f(p'') \sqsupseteq \lceil e' \rceil_{x(e)} \}.$$

It is easy to show that $p_0 \sqsubseteq p$ and $(p_0,\ e') \in R$. Hence $\lceil e \rceil_{x_e} \subseteq \Delta R\, u$ for any $e \in \Delta R\, u$, which implies the above equation.

We have

$$\begin{aligned}
(p, e),\ (p', e) \in R \ \& \ p,\ p' \in x_0 &\Longrightarrow f(p) \sqsupseteq \lceil e \rceil_x \ \& \ f(p') \sqsupseteq \lceil e \rceil_{x'} \ \& \ p \uparrow p' \\
&\Longrightarrow f(p) \uparrow f(p') \\
&\Longrightarrow \lceil e \rceil_x \uparrow \lceil e \rceil_{x'} \\
&\Longrightarrow \lceil e \rceil_x = \lceil e \rceil_{x'}.
\end{aligned}$$

From this it is easy to deduce $p = p'$ using the fact that $\lceil e \rceil_x$ is a complete prime and f is stable, linear.

Given $(p, e) \in R$, we have, for some $x \in \mathcal{F}$, $e \in x$ and $f(p) \sqsupseteq \lceil e \rceil_x$. Clearly $R \cap [p \downarrow \times \lceil e \rceil_x]$ is a finite set and $(p, e) \in R \cap [p \downarrow \times \lceil e \rceil_x]$, where $p \downarrow = \{ p' \mid p' \in D^1 \ \& \ p' \sqsubseteq p \}$. We need to show that for any $u \in \Phi D$,

$$\{ e' \mid \exists p' \in u. (p', e') \in R \cap [p \downarrow \times \lceil e \rceil_x] \} \in \mathcal{F}.$$

It is obvious that

$$\{ e' \mid \exists p' \in u. (p', e') \in R \cap [p \downarrow \times \lceil e \rceil_x] \} \subseteq \lceil e \rceil_x$$

and

$$\{ e' \mid \exists p' \in u. (p', e') \in R \cap [p \downarrow \times \lceil e \rceil_x] \} \subseteq \Delta R \{ p' \mid p' \in u \ \& p' \sqsubseteq p \}.$$

On the other hand, $\forall e_0 \in \lceil e \rceil_x \cap R\{ p' \mid p' \in u \ \& p' \sqsubseteq p \}$, there exists $p' \in u$, $p' \sqsubseteq p$ such that $(p', e_0) \in R$, where $e_0 \in \lceil e \rceil_x$. Therefore $(p', e_0) \in R \cap [p \downarrow \times \lceil e \rceil_x]$. We have shown that

$$\{ e' \mid \exists p' \in u. (p', e') \in R \cap [p \downarrow \times \lceil e \rceil_x] \} = \lceil e \rceil_x \cap R\{ p' \mid p' \in u \& p' \sqsubseteq p \}.$$

Now $(p, e) \in R$ implies $f(p) \sqsupseteq \lceil e \rceil_x$. Also,

$$\begin{aligned} e_0 \in R\{ p' \mid p' \in u \& p' \sqsubseteq p \} &\Longrightarrow \exists p' \in u. p' \sqsubseteq p \ \& \ (p', e_0) \in R \\ &\Longrightarrow \exists x_0. e_0 \in x_0 \ \& \ f(p') \sqsupseteq \lceil e_0 \rceil_{x_0} \\ &\Longrightarrow e_0 \in f(p) \qquad (f(p') \sqsubseteq f(p)). \end{aligned}$$

Hence $R\{ p' \mid p' \in u \ \& p' \sqsubseteq p \} \sqsubseteq f(p)$. As $\lceil e \rceil_x \uparrow R\{ p' \mid p' \in u \ \& p' \sqsubseteq p \}$,

$$\lceil e \rceil_x \cap R\{ p' \mid p' \in u \ \& p' \sqsubseteq p \} \in \mathcal{F}.$$

To summarise, we have shown that R is a linear map.

Commutativity: To show that the diagram commutes let $f(d) = x$, where $d \in D$, $x \in \Delta\mathcal{F}$. $\Theta_D(d) = \{ p \mid p \in D^1 \ \& \ p \sqsubseteq d \}$. We claim that $\Delta(R\Theta_D(d)) = x$, required by commutativity. Let $e \in x$. Clearly $f(d) = x \sqsupseteq \lceil e \rceil_x$. Set $p_0 = \bigsqcap \{ p \mid p \in D^1 \ \& \ f(p) \sqsupseteq \lceil e \rceil_x \}$. We have $f(p_0) \sqsupseteq \lceil e \rceil_x$ and $\forall p \sqsubseteq p_0. f(p) \sqsupseteq \lceil e \rceil_x \Longrightarrow p = p_0$. Therefore $(p_0, e) \in R$

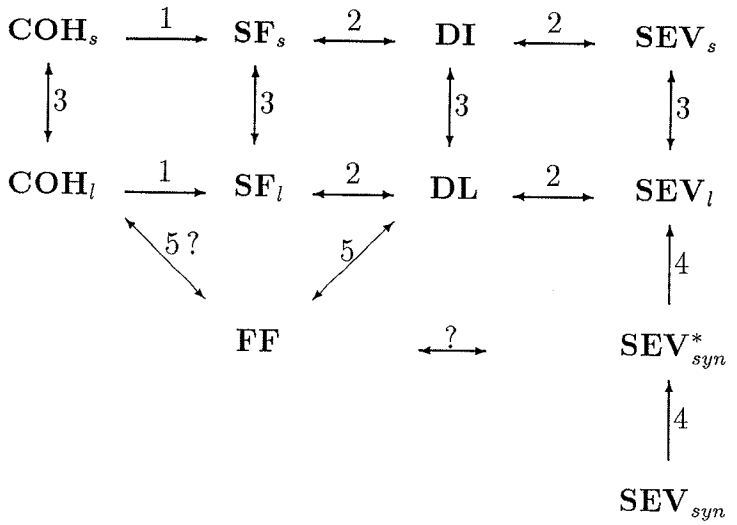
and $p_0 \sqsubseteq d$, and so $e \in \Delta(R\Theta_D(d))$. If $e \in \Delta(R\Theta_D(d))$ then $\exists p \sqsubseteq d.\ (p, e) \in R$. Hence $e \in x_0$ & $f(p) \sqsupseteq \lceil e \rceil_{x_0}$ for some x_0. Obviously this implies $e \in f(d)$ $(= x)$.

Uniqueness: Assume R' is a linear map which also makes the diagram commute. Let $(p', e') \in R'$ and $f(p') = x'$ $(= \Delta(R'\Theta_D(p'))$). Clearly $e' \in x'$ and $\forall p'' \sqsubseteq p'$, $f(p'') \sqsupseteq \lceil e' \rceil_{x'}$ implies $\Delta(R'\Theta_D(p'')) \sqsupseteq \lceil e' \rceil_{x'}$. Hence $e' \in \Delta(R'\Theta_D(p''))$, which means $(p_0, e') \in R'$ for some $p_0 \sqsubseteq p''$. This is only possible when $p_0 = p'$, as $p_0 \uparrow p'$. Thus $p'' = p'$, too, which implies $(p', e') \in R$. Therefore $R' \subseteq R$. On the other hand, suppose $(p, e) \in R$. We have

$$\Delta(R'\Theta_D(p)) = f(p) = \Delta(R\Theta_D(p)).$$

$e \in \Delta(R'\Theta_D(p))$ since $e \in \Delta(R\Theta_D(p))$. Hence $(p', e) \in R'$ for some $p' \sqsubseteq p$. However $R' \subseteq R$. So $(p', e) \in R$, which implies $p' = p$. ∎

Winskel told me that there is an adjunction between **DI** and **DL**. It is likely that there is a coreflection between **FF** and $\mathbf{COH}_l$. The relationship between **FF** and $\mathbf{SEV}^*_{syn}$, however, is not clear. $\mathbf{COH}_s$, $\mathbf{SF}_s$, **DI**, and $\mathbf{SEV}_s$ are cartesian closed. $\mathbf{COH}_l$, $\mathbf{SF}_l$, **DL**, $\mathbf{SEV}_l$, and **FF** are monoidal closed. I do not know whether or not $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$ are monoidal closed. We summarise the relationships among the categories by the following diagram:



where the labels have interpretations given below:

1 : full-subcategory;
2 : equivalent;
3 : adjunction;
4 : subcategory;
5 : coreflection;
? : unknown.

Chapter 8

Stable Neighbourhoods

Scott topology plays an essential role for the logic of domains developed in Chapter 5. One of the reasons for it is that Scott open sets characterise continuous functions. For dI-domains, is there a similar notion of 'open sets' which characterise stable functions? The answer is yes and *stable neighbourhoods* play the role of open sets.

This chapter introduces stable neighbourhoods. Through extensive study, especially with respect to the various constructions in the categories presented in the previous chapter, stable neighbourhoods are demonstrated to be fundamental, rich and general in the stable world. In particular, it is shown that, although stable neighbourhoods do not necessarily form a topology, they determine the associated dI-domain. Constructions on stable neighbourhoods are introduced in the categories **DI**, $\mathbf{COH}_l$, and $\mathbf{COH}_s$. These constructions show how stable neighbourhoods of a higher type can be built up from those of the component types. Stable neighbourhoods are important for the logic of dI-domain since they will be used to interpret the assertions. The constructions on stable neighbourhoods suggest proof rules for the logic of dI-domains.

The contents of this chapter is organised as follows. Section 1 introduces stable neighbourhoods and shows that stable neighbourhoods specify not only the stable functions but also the stable order. Section 2 gives a characterisation of the complete primes **DI**. Section 3 studies the stable-neighbourhood constructions in **DI**. Section 4 studies the stable-neighbourhood constructions in $\mathbf{COH}_l$ and $\mathbf{COH}_s$. Section 5 studies the stable-neighbourhood constructions in $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$.

8.1 Stable Neighbourhoods

Scott open sets play an important role in domain theory. They are also essential to the logic of **SFP** domains, the development of which is based on the view regarding open sets as properties. This is because Scott open sets have many nice properties, which include

- a function is continuous iff the inverse image of an open set is open;
- for continuous functions f and g, $f \sqsubseteq g$ iff $f^{-1}(O) \subseteq g^{-1}(O)$ for all open set O;

• Scott open sets form a topology.

Of course, when it comes to dI-domains we cannot hope Scott open sets do the same job. But are there any kind of 'open sets' of dI-domains which play the same role as Scott open sets do for domains?

The first fact we notice is that, if there is any class of sets of dI-domains which has a property corresponding to the first one stated above, those sets do not necessary form a topology.

Consider the stable functions from $\mathcal{O}^2$ to $\mathcal{O}$. Suppose there were such a topology. Then the topology on $\mathcal{O}$ must contain $\{\top\}$ as an open set: otherwise we may get non-monotonic functions. The inverse image of the stable function

$$(\top, \bot) \longmapsto \top, \quad (\bot, \top) \longmapsto \bot$$

on $\{\top\}$ is $\{(\top, \top), (\top, \bot)\}$ and the inverse image of the stable function

$$(\bot, \top) \longmapsto \top, \quad (\top, \bot) \longmapsto \bot$$

on $\{\top\}$ is $\{(\top, \top), (\bot, \top)\}$. Hence their union $\{(\top, \top), (\bot, \top), (\top, \bot)\}$ would be again an open set. That means the topology on $\mathcal{O}^2$ coincides with the Scott topology, which allows the non-stable 'parallel-or':

$$(\bot, \top) \longmapsto \top$$

$$(\top, \bot) \longmapsto \top$$

$$(\bot, \bot) \longmapsto \bot$$

Therefore the required topology does not exist.

It is still meaningful, however, to ask whether there is any class of sets which characterise stable functions in the following sense:

- a function is stable iff the inverse image of a set in this class is still a set in the class;
- for stable functions f, g, $f \sqsubseteq_s g$ iff $f^{-1}(O) \sqsubseteq g^{-1}(O)$ for all set O in this class, where $\sqsubseteq$ is a suitable order.

The answer is *yes*.

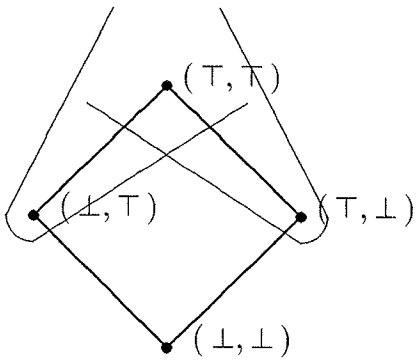
Let D be a Scott domain. We have $[D \to \mathcal{O}] \cong \Omega(D)$, *i.e.*, the continuous functions $[D \to \mathcal{O}]$ with the pointwise order are isomorphic to Scott open sets of D, with set inclusion as the order. The isomorphism is given by $f \longmapsto f^{-1}(\top)$. Now consider a dI-domain D and a stable function $f : D \to \mathcal{O}$. $f^{-1}\{\top\}$ is a Scott-open set as f is continuous. If $x \uparrow y$ and $x, y \in f^{-1}\{\top\}$ then $x \sqcap y \in f^{-1}\{\top\}$ since stable function preserves meets of compatible elements. This simple analysis leads us to

Definition 8.1.1 Let D be a dI-domain. U is a stable neighbourhood of D if

- U is Scott-open;
- $(x \uparrow y \;\&\; x, y \in U) \Rightarrow x \sqcap y \in U$.

Write the set of stable neighbourhoods of a dI-domain D as $\mathbf{SN}(D)$. $\mathbf{SN}(D)$ does not necessarily form a topology. It is closed under finite intersections but not unions.

Example 8.1.1 In $\mathcal{O}^2$, $\{(\bot\ ,\ \top)\ ,(\top\ ,\ \top)\}$ and $\{(\top\ ,\ \bot),\ (\top\ ,\ \top)\}$ are stable neighbourhoods but not their union.



Proposition 8.1.1 $(\mathbf{SN}(D), \subseteq)$ is a lattice.

Proof Let $U, V \in \mathbf{SN}(D)$. $U \cap V$ is Scott-open. $x \uparrow y \;\;\&\;\; x,\ y \in U \cap V$ implies $x \sqcap y \in U \;\&\; x \sqcap y \in V$. Therefore $U \cap V \in \mathbf{SN}(D)$, i.e., $U \sqcap V = U \cap V$. To show that $\mathbf{SN}(D)$ has joins, we first introduce a binary operation $\diamond$ between two open sets A and B:

$$A \diamond B =_{def} \{d \mid \exists x, y \in A \cup B.\ (x \uparrow y \;\&\; d \sqsupseteq x \sqcap y)\}.$$

It is easy to see that A and B open implies $A \diamond B$ open. Now for $U, V \in \mathbf{SN}(D)$, let

$$K_0 = U \diamond V, \quad K_1 = K_0 \diamond K_0, \quad \ldots \quad K_n = K_{n-1} \diamond K_{n-1}, \ldots$$

We claim that $\bigcup_{i \in \omega} K_i = U \sqcup V$. It is enough to check that $\bigcup_{i \in \omega} K_i$ is a stable neighbourhood,

and it actually is:

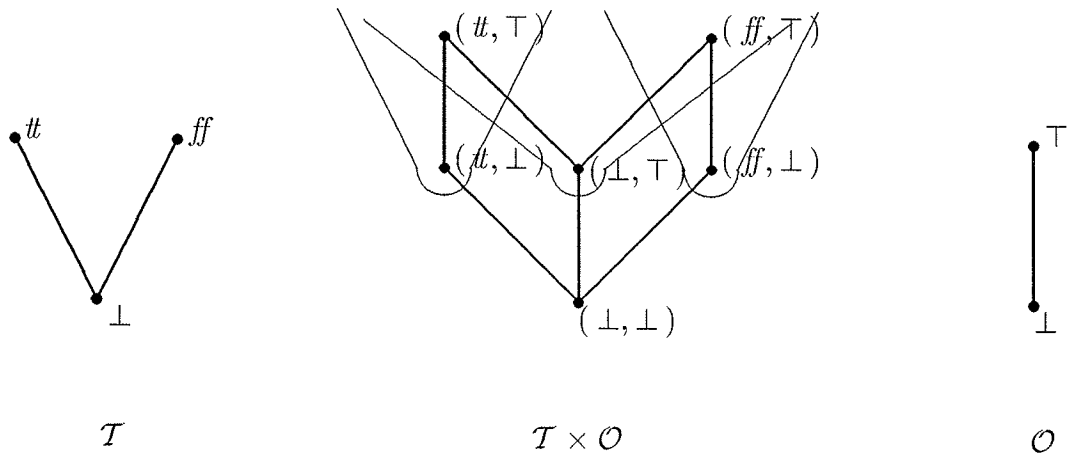
$$\begin{aligned} d \uparrow d' \ \& \ d, d' \in \bigcup_{i\in\omega} K_i \Rightarrow \quad & \exists m, n \in \omega.\, d \in K_n \ \& \ d' \in K_m \\ \Rightarrow \quad & d \in K_{n+m} \ \& \ d' \in K_{n+m} \\ \Rightarrow \quad & d \sqcap d' \in K_{n+m+1} \subseteq \bigcup_{i\in\omega} K_i. \end{aligned}$$

∎

$\mathbf{SN}(D)$ is, moreover, a complete lattice. The $\diamond$ operation can be extended to an arbitrary number of stable neighbourhoods and $\bigcup K_i$ will be their least upper bound. Then the meet of an arbitrary number of stable neighbourhoods is the join of all the stable neighbourhoods contained in their intersection.

However, $\mathbf{SN}(D)$ is not distributive, as the following example shows. Note that meet and join of $\mathbf{SN}(D)$ may not be the same as intersection and union of sets, so we write $\sqcap$ and $\sqcup$, with subscript $\mathbf{SN}(D)$ omitted.

Example 8.1.2 Consider $\mathcal{T} \times \mathcal{O}$.



Let

$$\begin{aligned} A &= \{\, (tt, \bot),\ (tt, \top) \,\}, \\ B &= \{\, (\bot, \top),\ (tt, \top),\ (ff, \top) \,\}, \\ C &= \{\, (ff, \bot),\ (ff, \top) \,\}. \end{aligned}$$

Then

$$A \sqcap (B \sqcup C) = A$$

but

$$(A \sqcap B) \sqcup (A \sqcap C) = \{\, (tt, \top),\ (ff, \top) \,\} \neq A.$$

Note $A, B \in \mathbf{SN}(D)$ does not imply $A \diamond B \in \mathbf{SN}(D)$.

If $f : D \to \mathcal{O}$ is a stable function, then clearly $f^{-1}(\top)$ is a stable neighbourhood. On the other hand, suppose U is a stable neighbourhood of D. Then $F(U)$ is a stable function, where

$$F(U)(x) =_{def} \begin{cases} \top & \text{if } x \in U \\ \bot & \text{if } x \in (D \setminus U) \end{cases}$$

However, set inclusion on stable neighbourhoods does not determine the stable order. In $[\mathcal{O} \to_s \mathcal{O}]$, for example, two stable functions $\lambda x.x$ and $\lambda x.\top$ have the property $\forall U \in \mathbf{SN}(\mathcal{O}).\ (\lambda x.x)^{-1}(U) \subseteq (\lambda x.\top)^{-1}(U)$, but we do not have $\lambda x.x \sqsubseteq_s \lambda x.\top$.

Suppose U, V are stable neighbourhoods of D such that $F(U) \sqsubseteq_s F(V)$, where $F : \mathbf{SN}(D) \to [D \to_s \mathcal{O}]$ is defined in the previous paragraph. Then for any x, y in D, $x \sqsubseteq y$ implies $F(U)(x) = F(U)(y) \sqcap F(V)(x)$. Obviously $x \in V$ when $x \in U$, *i.e.*, $U \subseteq V$. Moreover, whenever $x \sqsubseteq y \in U$ but $x \notin U$, it must also be true that $x \notin V$. This means a minimal point of U must also be a minimal point of V.

Definition 8.1.2 Let D be a dI-domain. The set of minimal points of $U \in \mathbf{SN}(D)$, write μU, consists of $m \in U$ such that $\forall x \sqsubseteq m.\ x \in U \Rightarrow x = m$. For $U, V \in \mathbf{SN}(D)$, U *minimally less than* V, write $U \sqsubseteq_\mu V$, if $\mu U \subseteq \mu V$.

Clearly $\sqsubseteq_\mu$ is a equivalence relation. $U \sqsubseteq_\mu V$ implies $U \subseteq V$ but not vice versa. Notice that if $U \sqsubseteq_\mu W$ and $U \subseteq V \subseteq W$, then $U \sqsubseteq_\mu V$. $U \sqsubseteq_\mu V$ iff $\exists W \in \mathbf{SN}(D).\ V = U \cup W\ \&\ U \cap W = \emptyset$. Every minimal point of a stable neighbourhood is a finite element. The following two propositions are immediate.

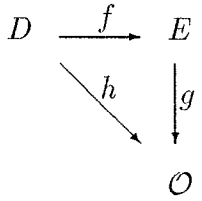
Proposition 8.1.2 Let D be a dI-domain. We have $[D \to_s \mathcal{O}] \cong \mathbf{SN}(D)$, with the stable order on $[D \to_s \mathcal{O}]$ and $\sqsubseteq_\mu$ on $\mathbf{SN}(D)$, where the isomorphism is given by

$$f \longmapsto f^{-1}(\top).$$

Proposition 8.1.3 $U \in \mathbf{SN}(D)$ implies there is some $K \subseteq D^0$, a pairwise incompatible set, such that

$$U = \bigcup \{\, k\uparrow \mid k \in K \,\}.$$

Suppose $f : D \to E$ is stable. For any $g : E \to \mathcal{O}$, there is a unique stable function $h : D \to \mathcal{O}$ which makes the following diagram commute.



This implies for any $g : E \to \mathcal{O}$, $f^{-1}(\, g^{-1}(\top)\,) = h^{-1}(\top) \in \mathbf{SN}(D)$, or, for any $U \in \mathbf{SN}(E)$, $f^{-1}(U) \in \mathbf{SN}(D)$, by Proposition 8.1.2. In general, we have

Theorem 8.1.1 Let D, E be dI-domains. $f : D \to E$ is stable iff

$$\forall U \in \mathbf{SN}(E).\ f^{-1}(U) \in \mathbf{SN}(D).$$

Proof ($\Rightarrow$) : As shown above.

($\Leftarrow$): Such f is monotonic: Let $d, d' \in D$ and $d' \sqsubseteq d$. For any finite e in E, if $e \sqsubseteq f(d')$ then $d' \in f^{-1}(e{\uparrow})$. $f^{-1}(e{\uparrow}) \in \mathbf{SN}(D)$ as $e{\uparrow} \in \mathbf{SN}(E)$. So $d \in f^{-1}(e{\uparrow})$, i.e., $f(d) \sqsupseteq e$. Therefore $f(d') \sqsubseteq f(d)$.

f preserves directed sups. Assume X is a directed set of D. $e \sqsubseteq f(\bigsqcup X)$ implies $\bigsqcup X \in f^{-1}(e{\uparrow})$, which is open. Hence $\exists x \in X$ such that $x \in f^{-1}(e{\uparrow})$, i.e., $f(x) \sqsupseteq e$. Therefore $f(\bigsqcup X) \sqsubseteq \bigsqcup f(X)$. Thus f is continuous.

To see f is stable notice we have, for any $e' \in E^0$,

$$\begin{aligned} x \uparrow y \ \&\ x, y \in D \ \&\ e' \sqsubseteq f(x) \sqcap f(y) \\ \Rightarrow\ & x \in f^{-1}(e'{\uparrow}) \ \&\ y \in f^{-1}(e'{\uparrow}) \\ \Rightarrow\ & x \sqcap y \in f^{-1}(e'{\uparrow}) \\ \Rightarrow\ & f(x \sqcap y) \sqsupseteq e'. \end{aligned}$$

Therefore $f(x \sqcap y) = f(x) \sqcap f(y)$. ∎

Suppose f, $f' : D \to E$ are stable functions. If $f \sqsubseteq_s f'$ then clearly for any $g : E \to \mathcal{O}$, $g \circ f \sqsubseteq_s g \circ f'$. By Proposition 8.1.2 we have, for any $g : E \to \mathcal{O}$, $f^{-1}(\, g^{-1}(\top)\,) \sqsubseteq_\mu (f')^{-1}(\, g^{-1}(\top)\,)$, or, for any $U \in \mathbf{SN}(E)$, $f^{-1}(U) \sqsubseteq_\mu (f')^{-1}(U)$. The other way round is also true. We have

Theorem 8.1.2 Let f, g be members of $[D \to_s E]$. $f \sqsubseteq_s g$ iff

$$\forall U \in \mathbf{SN}(E).\ f^{-1}(U) \sqsubseteq_\mu g^{-1}(U).$$

Proof It is enough to prove sufficiency. For any $x \in D$ and $d \sqsubseteq f(x)$ in E^0, We have $d\uparrow \in \mathbf{SN}(E)$ and $f^{-1}(d\uparrow) \subseteq g^{-1}(d\uparrow)$. Therefore $d \sqsubseteq g(x)$ and hence $f \sqsubseteq g$. Let $x \sqsubseteq y$ in D, $d' \in E^0$, and $y \sqsupseteq y_0 \in \mu f^{-1}(d'\uparrow)$. By assumption, $y_0 \in \mu g^{-1}(d'\uparrow)$.

$$\begin{aligned} d' \sqsubseteq f(y) \sqcap g(x) \Rightarrow & \quad x \in g^{-1}(d'\uparrow) \ \& \ y \in f^{-1}(d'\uparrow) \\ \Rightarrow & \quad y_0 \uparrow x \ \& \ y_0 \sqcap x \in g^{-1}(d'\uparrow) \\ \Rightarrow & \quad y_0 \sqcap x = y_0 \ \ (\text{ as } y_0 \in \mu g^{-1}(d'\uparrow)\ ) \\ \Rightarrow & \quad x \sqsupseteq y_0 \\ \Rightarrow & \quad f(x) \sqsupseteq f(y_0) \sqsupseteq d'. \end{aligned}$$

Therefore $f(x) \sqsupseteq f(y) \sqcap g(x)$. In other words, $f \sqsubseteq_s g$. ∎

We remark that if $f \sqsubseteq_s g$, and $x \in \mu g^{-1}(A)$, where A is a stable neighbourhood and $x \in f^{-1}(A)$, then it must be true that $x \in \mu f^{-1}(A)$.

Definition 8.1.3 K is a compact stable neighbourhood of $\mathbf{SN}(D)$ if K is a compact Scott open set and a stable neighbourhood. Write $\mathbf{KSN}(D)$ for the set of compact stable neighbourhoods of $\mathbf{SN}(D)$. Q is a *prime* stable neighbourhood of $\mathbf{SN}(D)$ if $\exists d \in D^0.\ P = d\uparrow$. P is a *very prime* stable neighbourhood of $\mathbf{SN}(D)$ if $\mu P \subseteq D^1$, *i.e.*, the minimal points of P are all complete primes. Write $\mathbf{PSN}(D)$ for the set of *very prime* stable neighbourhoods of $\mathbf{SN}(D)$.

Similar to Theorem 8.1.1 one can show that $f : D \to E$ is linear, stable iff

$$\forall U \in \mathbf{PSN}(E).\ f^{-1}(U) \in \mathbf{PSN}(D).$$

Proposition 8.1.4

$$A \sqsubseteq_\mu B \ \& \ S \sqsubseteq_\mu T \ \Rightarrow \ (A \cap S) \sqsubseteq_\mu (B \cap T)$$

where $A, B, S, T \in \mathbf{SN}(D)$.

Proof We have, for some A', $S' \in \mathbf{SN}(D)$, $B = A \cup A'$, $A \cap A' = \emptyset$, $T = S \cup S'$, $S \cap S' = \emptyset$. It follows that

$$B \cap T = (A \cap S) \cup \big[(A \cap S') \cup (A' \cap S) \cup (A' \cap S')\big],$$

where $(A \cap S) \cap \big[(A \cap S') \cup (A' \cap S) \cup (A' \cap S')\big] = \emptyset$ and $(A \cap S') \cup (A' \cap S) \cup (A' \cap S') \in \mathbf{SN}(D)$. Therefore $(A \cap S) \sqsubseteq_\mu (B \cap T)$. ∎

8.2 Complete Primes in $[D \to_s E]$

Let x be a finite element of a dI-domain D. Clearly $\{ y \in D \mid y \sqsupseteq x \}$ is a stable neighbourhood. Proposition 8.1.3 states that all stable neighbourhoods are disjoint unions of such basic stable neighbourhoods. Thus stable neighbourhoods are determined by the finite elements of the domain. For dI-domains are prime algebraic (Lemma 7.1.4), complete primes of a dI-domain fully determine the stable neighbourhoods. Therefore one way of understanding the stable neighbourhoods is via the complete primes.

If we know the complete primes of D and E, it is easy to get the complete primes of $D \times E$, $D_\perp$ and $D + E$. However, the complete primes in $[D \to_s E]$ are non-trivial and interesting. This section studies the structures of complete primes (hence finite elements, too) in the stable function space $[D \to_s E]$.

We know that if D and E are Scott domains, then the function space $D \to E$ is built up from step functions. In other words, the finite elements of $D \to E$ can be represented as step functions. For dI-domains, we can still use

Definition 8.2.1 Let D, E be dI-domains. A one-step function is a function f defined as

$$[a, b](x) = \begin{cases} b & \text{if } x \sqsupseteq a \ , \\ \perp & \text{otherwise} \end{cases}$$

where $a \in D^0$, $b \in E^0$.

Theorem 8.1.2 tells us that the stable order is determined by the minimal points for a stable function f to assume an value. Thus it is desired to read $[a, b]$ as a function f for which 'a is a minimal point for f to assume a value greater than or equal to b.' This immediately makes functions of the form $[a, b]$ ambiguous because Definition 8.2.1 only specifies *one* of such functions! To totally specify a stable function in this way we are led to

Definition 8.2.2 Let D, E be dI-domains. A finite set

$$\{ (a_i, b_i) \mid i \in I \} \subseteq D^0 \times E^1$$

is called *stable joinable* if

- $\forall J \subseteq I. \{ a_i \mid i \in J \} \uparrow \Rightarrow \{ b_i \mid i \in J \} \uparrow$,
- $a_i \uparrow a_j \ \& \ (b_i = b_j) \ \Rightarrow \ (a_i = a_j)$,
- $\forall b \in E^1. \ b_i \sqsupseteq b \ \Rightarrow \ \exists j. \ b_j = b \ \& \ a_i \sqsupseteq a_j$.

Let

$$\bigsqcup_{i\in I}[\,a_i, b_i\,] =_{def} \lambda x.\bigsqcup_{i\in I}[\,a_i, b_i\,](x),$$

the pointwise join of one-step functions. When

$$\{\,(a_i,\, b_i) \mid i \in I\,\} \subseteq D^0 \times E^1$$

is stable joinable,

$$\bigsqcup_{i\in I}[\,a_i, b_i\,]$$

is called a *step function.*

The first condition of Definition 8.2.2 implies *consistency*, so that $\bigsqcup_{i\in I}[\,a_i, b_i\,]$ really defines a function. The second condition expresses the *minimal* property, which means, when $[\,a_i,\, b_i\,]$ appears as a constituent of a step function, a_i is minimal for b_i. The third condition insists on *completeness*, in the sense that when a minimal value with respect to b_i is specified, all the b's below b_i must also have their minimal values specified.

By the conditions given in Definition 8.2.2, a sigleton set $\{\,(a,\, b)\,\}$ is usually *not* stable joinable. Some care is needed here for the notations. According to the definition $\bigsqcup_{i\in I}[\,a_i, b_i\,]$ is a pointwise join of one-step functions $[\,a_i, b_i\,]$. It is not necessarily the case that $\bigsqcup_{i\in I}[\,a_i, b_i\,]$ is a join of $[\,a_i, b_i\,]$'s under the stable order.

Proposition 8.2.1 Let D, E be dI-domains and $\{\,(a_i,\, b_i) \mid i \in I\,\}$ stable joinable. Then $\bigsqcup_{i\in I}[\,a_i, b_i\,]$ is a stable function.

Proof Suppose $\{\,(a_i,\, b_i) \mid i \in I\,\}$ is stable joinable. Obviously $\bigsqcup_{i\in I}[\,a_i, b_i\,]$ is continuous. To check stability let $x, y \in D$ and $x \uparrow y$. Suppose

$$p \sqsubseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](x) \sqcap \bigsqcup_{i\in I}[\,a_i, b_i\,](y)$$

where $p \in E$ is a complete prime. We have, for some $i,\ j$, $p \sqsubseteq b_i$, $a_i \sqsubseteq x$ and $p \sqsubseteq b_j$, $a_i \sqsubseteq y$. By Definition 8.2.2, there exists $s,\ t$ such that $b_s = p$, $a_s \sqsubseteq a_i$ and $b_t = p$, $a_t \sqsubseteq a_j$. $a_s = a_t$ as $a_s \uparrow a_t$ and $b_s = b_t$. Therefore $a_s = a_t \sqsubseteq x \sqcap y$ and

$$p \sqsubseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](x \sqcap y).$$

Since E is prime algebraic (Lemma 7.1.4),

$$\bigsqcup_{i\in I}[\,a_i, b_i\,](x \sqcap y) \sqsupseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](x) \sqcap \bigsqcup_{i\in I}[\,a_i, b_i\,](y),$$

which implies stability. ∎

Proposition 8.2.2 Suppose D, E are dI-domains and

$$\{ (a_i, b_i) \mid i \in I \} \subseteq D^0 \times E^1$$

stable joinable. Then for any $j \in I$ we have $a_j \in \mu f^{-1}(b_j\uparrow)$, where

$$f = \bigsqcup_{i \in I} [a_i, b_i].$$

Proof We have

$$\begin{aligned} f(a_j) &= \bigsqcup \{ b_i \mid a_i \sqsubseteq a_j \} \\ &\sqsupseteq b_j. \end{aligned}$$

Hence $a_j \in f^{-1}(b_j\uparrow)$. Let $y \sqsubseteq a_j$ and $f(y) \sqsupseteq b_j$, i.e. $\bigsqcup \{ b_i \mid a_i \sqsubseteq y \} \sqsupseteq b_j$. Since b_j is a complete prime, $b_i \sqsupseteq b_j$ for some i with $a_i \sqsubseteq y$. By Definition 8.2.2, there is some k such that $b_k = b_j$ and $a_k \sqsubseteq a_i$, which implies $a_k = a_j$ since $a_k \uparrow a_j$. Hence $y = a_j$. This means $a_j \in \mu f^{-1}(b_j\uparrow)$. ∎

There is a fact which I think worth remarking. Suppose f is a stable function. If $a \in \mu f^{-1}(b\uparrow)$, $a' \in \mu f^{-1}(b'\uparrow)$, where $a \uparrow a'$, then $a \sqcup a' \in \mu f^{-1}(b \sqcup b')\uparrow$. To derive this, note that $a_0 \in \mu f^{-1}(b \sqcup b')\uparrow$, where

$$a_0 = \bigsqcap \{ x \mid f(x) \sqsupseteq b \sqcup b' \},$$

by the stability of f. Clearly $a_0 \sqsubseteq a \sqcup a'$. But $f(a_0) \sqsupseteq b$ and $f(a_0) \sqsupseteq b'$, thus $a_0 \sqsupseteq a$, $a_0 \sqsupseteq a'$. Hence $a_0 \sqsupseteq a \sqcup a'$.

Proposition 8.2.3 Let D, E be dI-domains and

$$\{ (a_i, b_i) \mid i \in I \} \subseteq D^0 \times E^1$$

$$\{ (a'_j, b'_j) \mid i \in J \} \subseteq D^0 \times E^1$$

stable joinable. Then

$$\bigsqcup_{i \in I} [a_i, b_i] \sqsubseteq_s \bigsqcup_{j \in J} [a'_j, b'_j] \iff$$

$$\{ (a_i, b_i) \mid i \in I \} \subseteq \{ (a'_j, b'_j) \mid j \in J \}.$$

Proof ($\Rightarrow$): Suppose $\bigsqcup_{i \in I} [a_i, b_i] \sqsubseteq_s \bigsqcup_{j \in J} [a'_j, b'_j]$. For any $k \in I$,

$$\bigsqcup_{i \in I} [a_i, b_i](a_k) \sqsubseteq \bigsqcup_{j \in J} [a'_j, b'_j](a_k).$$

Therefore,

$$b_k \sqsubseteq \bigsqcup\{\ b'_j \mid a'_j \sqsubseteq a_k\ \}.$$

However, b_k is a complete prime. Hence $b_k \sqsubseteq b'_j$ for some j, with $a'_j \sqsubseteq a_k$. By Definition 8.2.2, there is some r such that $b_k = b'_r$ and $a'_r \sqsubseteq a'_j$. By Proposition 8.2.2

$$a_k \in \mu(\bigsqcup_{i\in I}[\,a_i, b_i\,])^{-1}(b_k\uparrow).$$

Hence

$$a_k \in \mu(\bigsqcup_{j\in J}[\,a'_j, b'_j\,])^{-1}(b'_r\uparrow),$$

using Theorem 8.1.2. However $a'_r \sqsubseteq a'_j \sqsubseteq a_k$ and

$$a'_r \in \mu(\bigsqcup_{j\in J}[\,a'_j, b'_j\,])^{-1}(b'_r\uparrow),$$

which is only possible when $a_k = a'_r$.

($\Leftarrow$): Assume $x,\ y \in D$, $x \sqsubseteq y$ and

$$p \sqsubseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](y) \sqcap \bigsqcup_{j\in J}[\,a'_j, b'_j\,](x),$$

where p is a complete prime of E. There must be $i,\ j$ such that $p \sqsubseteq b_i,\ a_i \sqsubseteq y$ and $p \sqsubseteq b'_j,\ a'_j \sqsubseteq x$. By Definition 8.2.2 there is some $t,\ r$ for which $b_t = p,\ a_t \sqsubseteq a_i$ and $b'_r = p,\ a'_r \sqsubseteq a'_j$. However $(a_t,\ b_t) \in \{\ (a'_j,\ b'_j) \mid j \in J\ \}$, $a_t \sqsubseteq y$, and $a'_r \sqsubseteq y$, so $a_t = a'_r$, which implies $a_t \sqsubseteq x$. This means

$$p \sqsubseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](x).$$

Since E is prime algebraic (Lemma 7.1.4), we have

$$\bigsqcup_{i\in I}[\,a_i, b_i\,](x) \sqsupseteq \bigsqcup_{i\in I}[\,a_i, b_i\,](y) \sqcap \bigsqcup_{j\in J}[\,a'_j, b'_j\,](x).$$

The other direction of the inequality is trivial. ∎

Since the stable order is stronger than the pointwise order, it is easy to see that step functions are finite elements in the stable function space. One can prove they are all the finite elements in the function space by brute-force by showing that every stable function is the limit of a chain of step functions. However, we would like to save some energy by using another form of finite elements suggested by T. Coquand.

Definition 8.2.3 Let $f \in D \to_s E$. Define

$$[a, b, f] = \lambda x. f(a \sqcap x) \sqcap b$$

where $a \in D^0$, $b \in E^0$.

Clearly we have

Lemma 8.2.1 $[a, b, f]$ is stable.

Lemma 8.2.2 $[a, b, f] \sqsubseteq_s f$.

Proof This is because for $x \sqsubseteq y$ in D,

$$\begin{aligned}
[a, b, f](y) \sqcap f(x) &= f(a \sqcap y) \sqcap b \sqcap f(x) \\
&= f(a \sqcap y \sqcap x) \sqcap b \\
&= f(a \sqcap x) \sqcap b \\
&= [a, b, f](x).
\end{aligned}$$

∎

Lemma 8.2.3 Any function in $[D \to_s E]$ which is stably less than $[a, b, f]$ is uniquely determined by its value at a.

Proof Suppose $h \sqsubseteq_s [a, b, f]$. $a \sqcap x \sqsubseteq x$ and $a \sqcap x \sqsubseteq a$ implies

$$h(a \sqcap x) = [a, b, f](a \sqcap x) \sqcap h(a)$$

and

$$h(a \sqcap x) = [a, b, f](a \sqcap x) \sqcap h(x).$$

Therefore

$$h(x) = h(a) \sqcap [a, b, f](x).$$

∎

Lemma 8.2.3 asserts that there are only finitely many stable functions below $[a, b, f]$.

Proposition 8.2.4 Every finite element of $D \to_s E$ is of the form $\bigsqcup_{i \in I} [a_i, b_i, f]$, where f ranges over stable functions and a_i's and b_i's range over finite elements of D and E, respectively.

Proof First note that each $[a, b, f]$ is finite. This is because assuming T is a directed set of stable functions in $D \to_s E$ such that

$$[a, b, f] \sqsubseteq_s \bigsqcup_{t \in T} t,$$

we have, by Lemma 7.1.2, that $[a, b, f](a) \sqsubseteq \bigsqcup_{t \in T} t(a)$. However, $[a, b, f](a) = f(a) \sqcap b$, which is finite as b is. Hence there exists $t_0 \in T$ such that $t_0(a) \sqsupseteq [a, b, f](a)$. For any $x \in D$,

$$t_0(a) \sqcap [a, b, f](a \sqcap x) = [a, b, f](a \sqcap x) \sqcap t_0(a \sqcap x),$$

by Lemma 7.1.1. But $t_0(a) \sqcap [a, b, f](a \sqcap x) = [a, b, f](x)$ and $[a, b, f](a \sqcap x) \sqcap t_0(a \sqcap x) \sqsubseteq t_0(x)$. Therefore $[a, b, f](x) \sqsubseteq t_0(x)$. Using Corollary 7.1.1 we see that $t_0 \sqsupseteq_s [a, b, f]$. Therefore $[a, b, f]$ is finite. Remember the sup of finitely many finite elements is finite. Also it is easy to see that $f = \bigsqcup \{ [a, b, f] \mid a \in D^0 \,\&\, b \in E^0 \}$. Hence the conclusion follows. ∎

Theorem 8.2.1 Let D, E be dI-domains. A function in $D \to_s E$ is a finite element iff it is equal to a step function determined by a finite stable joinable set.

Proof Obviously step functions are finite elements since they are as Scott functions under the pointwise order. We show that every finite element in $[D \to_s E]$ is equal to a step function. Proposition 8.2.4 concludes that every finite element in $[D \to_s E]$ is of the form

$$\bigsqcup_{1 \le i \le n} [a_i, b_i, f].$$

Therefore it is enough to prove that $\bigsqcup_{1 \le i \le n} [a_i, b_i, f]$ is equal to a step function. Write, for each $1 \le i \le n$, $\{ d \in E^1 \mid d \sqsubseteq f(a_i) \sqcap b_i \}$ as $\{ d_{i1}, d_{i2}, \cdots, d_{ik_i} \}$. Let

$$c_{ij} = \bigsqcap \{ x \sqsubseteq a_i \mid f(x) \sqcap b_i \sqsupseteq d_{ij} \}$$

for every $1 \le j \le k_i$. Let

$$\bigsqcup_{1 \le i \le n} \bigsqcup_{1 \le j \le k_i} [c_{ij}, d_{ij}] = \lambda x. \bigsqcup \{ d_{ij} \mid c_{ij} \sqsubseteq x \}.$$

$\bigsqcup_{1 \le i \le n} \bigsqcup_{1 \le j \le k_i} [c_{ij}, d_{ij}] = \bigsqcup_{1 \le i \le n} [a_i, b_i, f]$ because for each i

$$\begin{aligned} \bigsqcup \{ d_{ij} \mid c_{ij} \sqsubseteq x \} &\sqsubseteq \bigsqcup \{ f(c_{ij}) \sqcap b_i \mid c_{ij} \sqsubseteq x \} \\ &\sqsubseteq [a_i, b_i, f](x) \end{aligned}$$

and for any prime $d \sqsubseteq f(x \sqcap a_i) \sqcap b_i$ there exists s such that $d = d_{is}$ and $c_{is} \sqsubseteq x \sqcap a_i$. It is enough to check that

$$\bigcup_{1 \le i \le n} \{ (c_{ij}, d_{ij}) \mid 1 \le j \le k_i \}$$

satisfies the three requirements for a stable joinable set. Assume $I \subseteq \bigcup_{1 \le i \le n} \{ (i, j) \mid 1 \le j \le k_i \}$, and for all $(i, j) \in I$. $c_{ij} \sqsubseteq a$ for some $a \in D$. Then for each $(i, j) \in I$ we have $d_{ij} \sqsubseteq f(c_{ij} \sqcap a_i) \sqcap b_i \sqsubseteq f(a)$. Hence the first condition holds. The second and third conditions hold because of the way we defined c_{ij}. ∎

It is not difficult to observe further that

Proposition 8.2.5 The complete primes of $[D \to_s E]$, with D, E dI-domains, are of the form $[a, p, f]$, where f is a stable function, a is a finite element of D and p is a complete prime of E such that $p \sqsubseteq f(a)$.

Proof That each $[a, p, f]$ is a complete prime can be seen from the proof of Proposition 8.2.4, making use of the fact that p is a complete prime in E. They are all the complete primes because for any $[a, b, f]$, $[a, b, f] = \bigsqcup_{j \in J} [a, p_j, f]$, where p_j's are complete primes such that $\bigsqcup_{j \in J} p_j = f(a) \sqcap b$. ∎

Proposition 8.2.6 A stable joinable set

$$\{ (a_i, b_i) \mid i \in I \}$$

determines a complete prime $\bigsqcup_{i \in I} [a_i, b_i]$ iff there exists k such that $\forall i \in I.\, a_i \sqsubseteq a_k \;\&\; b_i \sqsubseteq b_k$.

Proof Use Proposition 8.2.5. ∎

8.3 Constructions for Stable Neighbourhoods in DI

In the previous section we studied how to get the stable neighbourhoods via the complete primes. In this section we study a more direct way to get the stable neighbourhoods. We introduce constructions to get stable neighbourhoods of a higher type from those of the type constituents in the category **DI**.

A dI-domain can be seen as a collection of computations of certain type. The stable neighbourhoods of the dI-domain can be taken as properties about the computations. Constructions on dI-domains can be seen as ways to combine computations together.

Suppose x is a computation of type D, having property A, written $x \models A$ and y is a computation of type E, having property B, written $y \models B$. If we combine the computations x of D and y of E together to get a computation $(x \text{ op } y)$ of type $[D \text{ op } E]$ (here op is some domain construction like sum, product, or stable function space), can we deduce some property of $(x \text{ op } y)$ from the facts $x \models A$ and $y \models B$? To answer this question one is lead to constructions $(A \text{ op } B)$ on properties A and B so that from $x \models A$ and $y \models B$ one deduces $(x \text{ op } y) \models (A \text{ op } B)$.

There can be many different ways to combine a stable neighbourhood A of D and a stable neighbourhood B of E together to get a stable neighbourhood $(A \text{ op } B)$ of $[D \text{ op } E]$. But the following are some basic requirements for constructions on stable neighbourhoods as I see.

- if A is a stable neighbourhood of D and B is a stable neighbourhood of E, $(A \text{ op } B)$ should be a stable neighbourhood of $[D \text{ op } E]$.
- through finite intersection of stable neighbourhoods of the form $(A \text{ op } B)$ it should be possible to get all prime stable neighbourhood of $[D \text{ op } E]$.
- for any $x \in D$, $y \in E$, $x \in A$ and $y \in B$ implies $(x \text{ op } y) \in (A \text{ op } B)$.

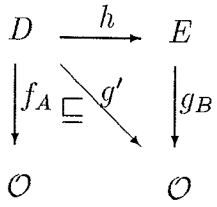
The first condition requires that $(A \text{ op } B)$ is well-defined. The second condition states that stable neighbourhoods of the form $(A \text{ op } B)$ are expressive enough–we can get all the compact stable neighbourhoods by using finite intersection and finite union. The third condition ensures that the way we combine the stable neighbourhoods captures the way we combine the computations.

We make a remark at the beginning that *all the constructions introduced in this section and next section have the above three properties satisfied.* However some of them are obvious and we do not always explicitly check them all.

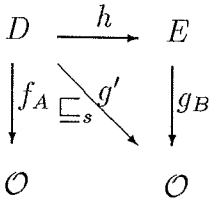
Suppose D and E are dI-domains. Similar to **SFP** objects, it is easy to get the stable neighbourhoods of $D + E$, $D \times E$ and $D_\perp$ from those of D and E with the three requirements satisfied. But how can we get the stable neighbourhoods of $[D \to_s E]$ from those of D and E directly?

Let us first have a look at how we dealt with this problem for the Scott topology. If

A and B are compact Scott open sets of D and E, respectively, then $A \to B = \{\ h : D \to E \mid A \subseteq h^{-1}(B)\ \}$ is a compact Scott open set of $[D \to E]$. There is another way to look at this. Since A and B are Scott open sets, they correspond to some functions $f_A : D \to \mathcal{O}$, $g_B : E \to \mathcal{O}$. Set inclusion on open sets determines the pointwise order, hence $h \in A \to B$ iff $f_A \subseteq g_B \circ h$ (see the diagram below).



This suggests that, for dI-domains, we should use the diagram



where the pointwise order is replaced by the stable order. This suggests the definition $h \in A \to B$ iff $f_A \sqsubseteq_s g_B \circ h$. By the analysis given just before Theorem 8.1.2, $f_A \sqsubseteq_s g_B \circ h$ iff $A \sqsubseteq_\mu h^{-1}(B)$. The following definition is reasonable.

Definition 8.3.1 Let D, E be dI-domains, $A \in \mathbf{KSN}(D)$ and $B \in \mathbf{KSN}(E)$. Define

$$A \to B = \{f \in D \to_s E \mid A \sqsubseteq_\mu f^{-1}B\}.$$

Following the view mentioned at the beginning of this section, let us think of a stable function f as a computation of type $[D \to_s E]$ which consumes some information of type D and produces some information of type E (here we can identify the computations of type D and E as data, or information, of type D and E, respectively). What does it mean intuitively for a computation of $[D \to_s E]$ to have a property $A \to B$, where A is a property of type D and B is a property of type E? The properties appropriate for stable functions are those which are determined by a set of incompatible minimal information. We can say that f has property $A \to B$ if f can produce some information with property B from any input information with property A and, moreover, a minimal information of property A is also a minimal information for f to produce some information with property B. We can also say that f has property $A \to B$ if whenever f can produce an output

(information) with property B, there is always some minimal input information x for f to do so. If this minamal information x happens to be consistent with property A, then it must also be a minimal information of property A.

Proposition 8.3.1 Let $\{(a_i, b_i) \mid i \in I\}$ be a stable joinable set. Then $f \in (a_j\uparrow \to b_j\uparrow)$ for any $j \in I$, where f is an abbreviation for the step function $\bigsqcup_{i\in I}[a_i, b_i]$ determined by the stable joinable set.

Proof Directly follows from Proposition 8.2.2. ∎

Proposition 8.3.2 Let A, B, C, D be stable neighbourhoods and a a finite element. Then with appropriate types we have

$$A \cap B = \emptyset \Longrightarrow (A \cup B) \to C = (A \to C) \cap (B \to C),$$

$$A \cap B = \emptyset \Longrightarrow a\uparrow \to (A \cup B) = (a\uparrow \to A) \cup (a\uparrow \to B),$$

$$(A \to C) \cap (B \to D) \subseteq (A \cap B) \to (C \cap D).$$

Proof Only the last inequality needs verification. Let $f \in (A \to C) \cap (B \to D)$. We have $A \sqsubseteq_\mu f^{-1}C$ and $B \sqsubseteq_\mu f^{-1}D$. By Proposition 8.1.5, $A \cap B \sqsubseteq_\mu (f^{-1}C) \cap (f^{-1}D)$. But $(f^{-1}C) \cap (f^{-1}D) = f^{-1}(C \cap D)$. Therefore the desired inequality follows. ∎

The third conclusion of Proposition 8.3.2 is, in fact, a generalisation of the remark given after the proof of Proposition 8.2.2.

Because $f \in A \to B$ iff $A \sqsubseteq_\mu f^{-1}B$, the rule

$$A' \supseteq A \;\&\; B \subseteq B' \Longrightarrow (A \to B) \subseteq (A' \to B')$$

is no longer valid for dI-domains and stable neighbourhoods.

Theorem 8.3.1 Assume $A \in \mathbf{KSN}(D)$ and $B \in \mathbf{KSN}(E)$. Then

$$(A \to B) \in \mathbf{KSN}([D \to_s E]).$$

We have to restrict A and B to compact open sets. Otherwise $(A \to B)$ can be a non-open set, for the same reason as explained for Scott domains in Chapter 6.

Proof First we prove that $(A \to B)$ is Scott open. It is a direct consequence of Theorem 8.1.2 that $(A \to B)$ is upwards closed. Suppose

$$f_0 \sqsubseteq_s f_1 \sqsubseteq_s \cdots \sqsubseteq_s f_n \sqsubseteq_s \cdots$$

is a chain and $\bigsqcup_{i\in\omega} f_i \in (A \to B)$. By definition $\mu A \subseteq \mu(\bigsqcup_{i\in\omega} f_i)^{-1}B$. As A is compact, μA is finite. Let

$$\mu A = \{ a_1, a_2, \cdots, a_m \}.$$

$a_j \in \mu(\bigsqcup_{i\in\omega} f_i)^{-1}B$ implies, by Lemma 8.2.2, $\bigsqcup_{i\in\omega} f_i(a_j) \in B$. There exists I_j such that $f_{I_j}(a_j) \in B$, as B is open. Let $n = \max\{ I_j \mid 1 \leq j \leq m \}$. We have $f_n(a_j) \in B$ for all $1 \leq j \leq m$. It is then easy to see that $\mu A \subseteq \mu f_n^{-1}B$, i.e. $f_n \in (A \to B)$. Namely, $(A \to B)$ is open.

Assume $f, g \in (A \to B)$ and $f \uparrow g$. By Theorem 8.1.2 we have

$$\mu(f \sqcap g)^{-1}(B) \subseteq \mu f^{-1}(B) \cap \mu g^{-1}(B).$$

On the other hand,

$$\begin{aligned} x \in & \mu f^{-1}(B) \cap \mu g^{-1}(B) \\ & \Rightarrow\ f(x) \in B \ \&\ g(x) \in B \\ & \Rightarrow\ f(x) \sqcap g(x) \in B \\ & \Rightarrow\ (f \sqcap g)(x) \in B \qquad \text{(Lemma 8.2.3)} \\ & \Rightarrow\ x \in \mu(f \sqcap g)^{-1}(B). \end{aligned}$$

Hence

$$\mu(f \sqcap g)^{-1}(B) \ =\ \mu f^{-1}(B) \cap \mu g^{-1}(B).$$

Now

$$\begin{aligned} f \in (A \to B) \ \&\ g \in (A \to B) \ &\Rightarrow\ A \sqsubseteq_\mu f^{-1}(B) \ \&\ A \sqsubseteq_\mu g^{-1}(B) \\ &\Rightarrow\ \mu A \subseteq \mu f^{-1}(B) \cap \mu g^{-1}(B) \\ &\Rightarrow\ \mu A \subseteq \mu(f \sqcap g)^{-1}(B). \end{aligned}$$

Therefore $f \sqcap g \in (A \to B)$, and $(A \to B)$ is a stable neighbourhood.

To show that $(A \to B)$ is compact we first prove that stable neighbourhoods of the form $(a\!\uparrow \to b\!\uparrow)$ are compact, where $a \in D^0$, $b \in E^0$. Let $Q_a = \{ c \in D \mid c \sqsubseteq a \}$, $P_b = \{ p \in E^1 \mid p \sqsubseteq b \}$, and

$$F_a^b = \{ f \mid f \text{ is a step function } \ \&\ f \subseteq Q_a \times P_b \ \&\ a \in \mu f^{-1}(b\!\uparrow) \}.$$

We claim that

$$(a\!\uparrow \to b\!\uparrow) = \bigcup_{g \in F_a^b} g\!\uparrow.$$

Obviously

$$(a\uparrow \to b\uparrow) \supseteq \bigcup_{g \in F_a^b} g\uparrow.$$

On the other hand, let $f \in (a\uparrow \to b\uparrow)$. Clearly $[a, b, f] \in F_a^b$. Hence $f \in \bigcup_{g \in F_a^b} g\uparrow$ since $f \sqsupseteq_s [a, b, f]$. Therefore $(a\uparrow \to b\uparrow)$ is compact.

Write $A = \bigcup_{i \in I}(a_i\uparrow)$, $B = \bigcup_{j \in J}(b_j\uparrow)$, where I and J are finite and a_i's are pairwise incompatible, b_j's are pairwise incompatible. It is easy to see that

$$\begin{aligned} A \to B &= (\bigcup_{i \in I} a_i\uparrow) \to (\bigcup_{j \in J} b_j\uparrow) \\ &= \bigcap_{i \in I}(a_i\uparrow \to \bigcup_{j \in J} b_j\uparrow) \\ &= \bigcap_{i \in I}[\bigcup_{j \in J}(a_i\uparrow \to b_j\uparrow)]. \end{aligned}$$

Hence $A \to B$ is compact. ∎

From this theorem we can also see that it is possible to get all the compact stable neighbourhoods of $[D \to_s E]$ by finite union and intersection of stable neighbourhoods of the form $A \to B$, where A, B are compact stable neighbourhoods of D and E, respectively. In particular we can get stable neighbourhoods whose minimal point is a single stable function. We also have, for $f, g \in [D \to_s E]$, $f \sqsubseteq_s g$ iff $f \in (A \to B)$ implies $g \in (A \to B)$ for all $A \in \mathbf{SN}(D)$, $B \in \mathbf{SN}(E)$.

Proposition 8.3.3 Let $a \in D^0$ and $b, c \in E^0$, where D, E are dI-domains.

$$c \sqsubseteq b \Longrightarrow (a\uparrow \to b\uparrow) \subseteq \bigcup_{a' \sqsubseteq a} (a'\uparrow \to c\uparrow).$$

Note that if $a' \neq a''$ and $a' \uparrow a''$ then $(a'\uparrow \to c\uparrow) \cap (a''\uparrow \to c\uparrow) = \emptyset$. Hence actually $\bigcup_{a' \sqsubseteq a}(a'\uparrow \to c\uparrow)$ is a stable neighbourhood of $[D \to_s E]$.

Proof Suppose $f \in (a\uparrow \to b\uparrow)$. Then $a \in \mu f^{-1} b\uparrow$. We have $f(a) \sqsupseteq c$. Let $a'' = \bigsqcap \{ x \mid x \sqsubseteq a \ \& \ f(x) \sqsupseteq c \}$. Clearly $a'' \sqsubseteq a$ and $a'' \in \mu f^{-1} c\uparrow$. Hence $f \in \bigcup_{a' \sqsubseteq a}(a'\uparrow \to c\uparrow)$. ∎

Theorem 8.3.2 Let $\{(a_i, b_i) \mid i \in I\}$ be a stable joinable set. Then

$$\bigcap_{i \in I}(a_i\uparrow \to b_i\uparrow) = \Big[\bigsqcup_{i \in I}[a_i, b_i]\Big]\uparrow .$$

This theorem says that in the stable function space, if we take the intersection of the

stable neighbourhoods $(a_i\uparrow \to b_i\uparrow)$, $i \in I$, we get a stable neighbourhood consisting of all the stable functions which dominate the step function $\bigsqcup_{i\in I}[a_i, b_i]$ under the stable order.

Proof We know from Proposition 8.2.2 that

$$\bigsqcup_{i\in I}[a_i, b_i] \in \bigcap_{i\in I}(a_i\uparrow \to b_i\uparrow).$$

It is enough to show that $\bigsqcup_{i\in I}[a_i, b_i]$ is less than any other stable function in $\bigcap_{i\in I}(a_i\uparrow \to b_i\uparrow)$. Let g be a stable function in $\bigcap_{i\in I}(a_i\uparrow \to b_i\uparrow)$. For any $i \in I$, $g(a_i) \sqsupseteq b_i$. Therefore for any x in D

$$\begin{aligned}\bigsqcup\{ b_k \mid a_k \sqsubseteq x \} &\sqsubseteq \bigsqcup\{ g(a_k) \mid a_k \sqsubseteq x \} \\ &\sqsubseteq g(x),\end{aligned}$$

i.e.,

$$\bigsqcup_{i\in I}[a_i, b_i](x) \sqsubseteq g(x).$$

Suppose $x, y \in D$ and $x \sqsubseteq y$. Let $p \sqsubseteq \bigsqcup\{ b_j \sqcap g(x) \mid a_j \sqsubseteq y \}$, where p is a complete prime. $p \sqsubseteq b_j \sqcap g(x)$ for some j. Therefore, there exists s such that $p = b_s$ and $a_j \sqsupseteq a_s$. $g(a_s \sqcap x) = g(a_s) \sqcap g(x) \sqsupseteq b_s$. This implies, as $g \in (a_s\uparrow \to b_s\uparrow)$, $a_s \sqcap x = a_s$, or $a_s \sqsubseteq x$. Hence $p \sqsubseteq \bigsqcup\{ b_i \mid a_i \sqsubseteq x \}$. By prime algebraicity of E,

$$\begin{aligned}g(x) \sqcap \bigsqcup\{ b_i \mid a_i \sqsubseteq y \} &= \bigsqcup\{ b_j \sqcap g(x) \mid a_j \sqsubseteq y \} \\ &\sqsubseteq \bigsqcup\{ b_i \mid a_i \sqsubseteq x \}.\end{aligned}$$

Now it is easy to see that

$$\bigsqcup_{i\in I}[a_i, b_i] \sqsubseteq_s g.$$

∎

8.4 Stable-Neighbourhood Constructions in $\mathbf{COH}_l$

This section studies the stable-neighbourhood constructions in $\mathbf{COH}_l$. Since the sum and product constructions are easy, by introducing the shriek construction on stable neighbourhood we get the constructions in $\mathbf{COH}_s$ also. We introduce stable-neighbourhood constructions corresponding to the tensor product, the stable function space, the linear function space, and the shriek operation. We show that all these constructions preserve compactness. We also give equations which indicate how those constructions interact with unions and intersections.

Theorem 8.4.1 Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent spaces and $A \in \mathbf{KSN}(\mathcal{F}_0)$, $B \in \mathbf{KSN}(\mathcal{F}_1)$. Then $A \otimes B \in \mathbf{KSN}(\mathcal{F}_0 \otimes \mathcal{F}_1)$, where

$$A \otimes B =_{def} \{\, x \in \mathcal{F}_0 \otimes \mathcal{F}_1 \mid \exists x_0 \in A,\, x_1 \in B.\, x_0 \times x_1 \subseteq^{fin} x \,\}.$$

Proof Clearly $A \otimes B$ is upwards closed. Suppose

$$u_0 \subseteq u_1 \subseteq \cdots \subseteq u_n \subseteq \cdots$$

is a chain in $\mathcal{F}_0 \otimes \mathcal{F}_1$ such that $\bigcup_{i\in\omega} u_i \in A \otimes B$. This implies $z_0 \times z_1 \subseteq \bigcup_{i\in\omega} u_i$ for some $z_0 \in A$, $z_1 \in B$, where z_0 and z_1 are finite. From $z_0 \subseteq \pi_0 \bigcup_{i\in\omega} u_i$ and $z_1 \subseteq \pi_1 \bigcup_{i\in\omega} u_i$ we derive that for some i, $z_0 \subseteq \pi_0 u_i$, $z_1 \subseteq \pi_1 u_i$, hence $z_0 \times z_1 \subseteq u_i$. Thus $u_i \in A \otimes B$. So $A \otimes B$ is open.

Assume $x, y \in A \otimes B$ and $x \uparrow y$. Then there exist $x_0, x_1 \in A$, $y_0, y_1 \in B$ such that $x_0 \times y_0 \subseteq^{fin} x$, $x_1 \times y_1 \subseteq^{fin} y$. We have $x_0 \uparrow x_1$ and $y_0 \uparrow y_1$. Therefore $x_0 \cap x_1 \in A$, $y_0 \cap y_1 \in B$. Also, $(x_0 \cap x_1) \times (y_0 \cap y_1) \subseteq^{fin} x \cap y$. Hence $x \cap y \in A \otimes B$.

To check compactness we first show that $(a\!\uparrow \otimes b\!\uparrow)$ is compact, where a, b are finite configurations of $\mathcal{F}_0$ and $\mathcal{F}_1$, respectively. Clearly $a \times b \in (a\!\uparrow \otimes b\!\uparrow)$. On the other hand, suppose $u \in (a\!\uparrow \otimes b\!\uparrow)$. Then by definition, there exist a', b', $a \subseteq a' \in \mathcal{F}_0$, $b \subseteq b' \in \mathcal{F}_1$, such that $a' \times b' \subseteq^{fin} u$. Hence $a \times b \subseteq^{fin} u$. We have shown that

$$(a\!\uparrow \otimes b\!\uparrow) = (a \times b)\!\uparrow,$$

therefore $(a\!\uparrow \otimes b\!\uparrow)$ is compact. By Proposition 8.1.3 and the fact that $A \otimes (B \cup C) = (A \otimes B) \cup (A \otimes C)$ when $B \cap C = \emptyset$, we deduce that any $A \otimes B$ is compact. ∎

From the proof we can easily see that

$$\widehat{e_0} \otimes \widehat{e_1} = \{\, x \in \mathcal{F}_0 \otimes \mathcal{F}_1 \mid (e_0, e_1) \in x \,\},$$

where $\widehat{e_0} = \{\, x_0 \in \mathcal{F}_0 \mid e_0 \in x_0 \,\}$ and $\widehat{e_1} = \{\, x_1 \in \mathcal{F}_1 \mid e_1 \in x_1 \,\}$. It is also clear that by using finite union and finite intersection we can get all compact stable neighbourhood of $\mathcal{F}_0 \otimes \mathcal{F}_1$ out of compact neighbourhoods of the form $A \otimes B$.

It is also clear that for $x, y \in \mathcal{F}_0 \otimes \mathcal{F}_1$, $x \subseteq y$ iff $x \in (A \otimes B)$ implies $y \in (A \otimes B)$ for all $A \in \mathbf{SN}(\mathcal{F}_0)$, $B \in \mathbf{SN}(\mathcal{F}_1)$.

Proposition 8.4.1 Suppose $A_1,\ A_2 \in \mathbf{KSN}(\mathcal{F}_0)$, $B_1,\ B_2 \in \mathbf{KSN}(\mathcal{F}_1)$, where $\mathcal{F}_0$, $\mathcal{F}_1$ are coherent spaces. Then

$$(A_1 \cap A_2) \otimes (B_1 \cap B_2)$$

$$= (A_1 \otimes B_1) \cap (A_2 \otimes B_1) \cap (A_1 \otimes B_2) \cap (A_2 \otimes B_2).$$

The proof goes through without using the compactness assumption.

Proof $\subseteq$: Suppose $x \in (A_1 \cap A_2) \otimes (B_1 \cap B_2)$. Then there exist $y_0 \in A_1 \cap A_2$, $y_1 \in B_1 \cap B_2$ such that $y_0 \times y_1 \subseteq^{fin} x$. It is obvious

$$x \in (A_1 \otimes B_1) \cap (A_2 \otimes B_1) \cap (A_1 \otimes B_2) \cap (A_2 \otimes B_2).$$

$\supseteq$: Assume

$$x \in (A_1 \otimes B_1) \cap (A_2 \otimes B_1) \cap (A_1 \otimes B_2) \cap (A_2 \otimes B_2).$$

By definition there are y_{11}, y_{21}, y_{12}, $y_{22} \in \mathcal{F}_0$ and z_{11}, z_{21}, z_{12}, $z_{22} \in \mathcal{F}_0$ such that $y_{ij} \in A_i$, $z_{ij} \in B_j$, and $y_{ij} \times z_{ij} \subseteq^{fin} x$ for $i = 1, 2$, $j = 1, 2$. We have, as A_1, B_1, A_2, B_2 are stable neighbourhoods,

$$y_{11} \cap y_{12} \in A_1,\ \ y_{21} \cap y_{22} \in A_2,$$

$$z_{11} \cap z_{21} \in B_1,\ \ z_{12} \cap z_{22} \in B_2.$$

Clearly

$$(y_{11} \cap y_{12}) \cup (y_{21} \cap y_{22}) \in A_1 \cap A_2,$$

$$(z_{11} \cap z_{21}) \cup (z_{12} \cap z_{22}) \in B_1 \cap B_2,$$

and,

$$[(y_{11} \cap y_{12}) \cup (y_{21} \cap y_{22})] \times [(z_{11} \cap z_{21}) \cup (z_{12} \cap z_{22})]$$

$$\subseteq (y_{11} \times z_{11}) \cup (y_{12} \times z_{12}) \cup (y_{21} \times z_{21}) \cup (y_{22} \times z_{22})$$

$$\subseteq^{fin} x.$$

Therefore $x \in (A_1 \cap A_2) \otimes (B_1 \cap B_2)$. ∎

Both Theorem 8.4.1 and Proposition 8.4.1 can be generalise to the tensor product of stable event structures.

Theorem 8.4.2 Let $\mathcal{F}_0$, $\mathcal{F}_1$ be coherent spaces and $A \in \mathbf{KSN}(\mathcal{F}_0)$, $B \in \mathbf{KSN}(\mathcal{F}_1)$. Then $A \multimap B \in \mathbf{KSN}(\mathcal{F}_0 \otimes \mathcal{F}_1)$, where

$$A \multimap B =_{def} \{\ x \in \mathcal{F}_0 \multimap \mathcal{F}_1 \mid A \sqsubseteq_\mu (Pt\, x)^{-1} B\ \},$$

$Pt\,x$ is as defined in section 7.1 of Chapter 7, and $\sqsubseteq_\mu$ is given in Definition 8.1.2.

Proof Similar to that of Theorem 8.3.1. ∎

We remark that

$$(\widehat{e_0} \multimap \widehat{e_1}) = \{\, x \in \mathcal{F}_0 \multimap \mathcal{F}_1 \mid (e_0, e_1) \in x \,\},$$

where $\widehat{e_0} = \{\, x_0 \in \mathcal{F}_0 \mid e_0 \in x_0 \,\}$ and $\widehat{e_1} = \{\, x_1 \in \mathcal{F}_1 \mid e_1 \in x_1 \,\}$. In fact, let $x \in \mathcal{F}_0 \multimap \mathcal{F}_1$ and $(e_0, e_1) \in x$. Then $\{\, e_1 \,\} \subseteq Pt\,x\ \{\, e_0 \,\}$, and $\{\, e_0 \,\} \in \mu(Pt\,x)^{-1}\widehat{e_1}$. So $x \in (\widehat{e_0} \multimap \widehat{e_1})$. Suppose, on the other hand, that $x \in (\widehat{e_0} \multimap \widehat{e_1})$. We have $\{\, e_0 \,\} \in \mu(Pt\,x)^{-1}\widehat{e_1}$. Therefore, $Pt\,x\ \{\, e_0 \,\} \supseteq \{\, e_1 \,\}$ and $(e_0, e_1) \in x$.

By this observation, each stable neighbourhood of $\mathcal{F}_0 \multimap \mathcal{F}_1$ can be constructed out of $A \multimap B$ by using finite union and finite intersection. Also, for $x, y \in \mathcal{F}_0 \multimap \mathcal{F}_1$, $x \subseteq y$ iff $x \in (A \multimap B)$ implies $y \in (A \multimap B)$ for all $A \in \mathbf{SN}(\mathcal{F}_0)$, $B \in \mathbf{SN}(\mathcal{F}_1)$.

Proposition 8.4.2 Let A, B, C, D be stable neighbourhoods and a a finite configuration. Then with appropriate types we have

$$A \cap B = \emptyset \Longrightarrow (A \cup B) \multimap C = (A \multimap C) \cap (B \multimap C)$$

$$A \cap B = \emptyset \Longrightarrow a{\uparrow} \multimap (A \cup B) = (a{\uparrow} \multimap A) \cup (a{\uparrow} \multimap B)$$

$$(A \multimap C) \cap (B \multimap D) \subseteq (A \cap B) \multimap (C \cap D)$$

Proof Easy. ∎

Note the rule

$$A' \supseteq A \;\&\; B \subseteq B' \Longrightarrow (A \multimap B) \subseteq (A' \multimap B')$$

is not valid.

Proposition 8.4.3 Let $x \in \mathcal{F}_0$ and $y \in \mathcal{F}_1$ be finite configurations and $\mathcal{F}_0$, $\mathcal{F}_1$ coherent spaces.

$$e' \in y \Longrightarrow (x{\uparrow} \multimap y{\uparrow}) \subseteq \bigcup_{e \in x} (\hat{e} \multimap \hat{e'}).$$

Proof Suppose $w \in (x{\uparrow} \multimap y{\uparrow})$ and $e' \in y$. Then $x \in \mu(Pt\,w)^{-1}y{\uparrow}$. We have $(Pt\,w)\ (x) \supseteq y$. This implies $e' \in (Pt\,w)\ (x)$. Hence $\exists e \in x.\ (e, e') \in w$, which implies $w \in \hat{e} \multimap \hat{e'}$, by the remark given after Theorem 8.4.2. ∎

Theorem 8.4.3 Let $\mathcal{F}$ be a coherent space and $A \in \mathbf{KSN}(\mathcal{F})$. Then $!A \in \mathbf{KSN}(!\mathcal{F})$, where $!A =_{def} \{\ x \in !\mathcal{F} \mid x \cap \mu A \neq \emptyset\ \}$, and μA, defined in Definition 8.1.2, is the set of minimal points of A.

Proof Clearly $!A$ is upwards closed. It is open because the elements of μA are finite configurations. Now suppose $x, y \in !A$ and $x \uparrow y$. We have $x \cap y \in !\mathcal{F}$, $x \cap \mu A \neq \emptyset$, and $y \cap \mu A \neq \emptyset$. But $x \cup y \in !\mathcal{F}$, hence $(x \cap y) \cap \mu A \neq \emptyset$, because the elements of μA are pairwise inconsistent. So $x \cap y \in !A$ and $!A$ is a stable neighbourhood.

To see $!A \in \mathbf{KSN}(!\mathcal{F})$ note that

$$!A = \bigcup_{a \in \mu A} !(\{\, a \,\}\uparrow)$$

and each $!(\{\, a \,\}\uparrow)$ is compact. ∎

Notice that for $x, y \in !\mathcal{F}$, $x \subseteq y$ iff $x \in !A$ implies $y \in !A$ for all $A \in \mathbf{SN}(\mathcal{F})$.

Proposition 8.4.4 Let $A, B \in \mathbf{SN}(\mathcal{F})$ and $A \cap B = \emptyset$. Then

$$!(A \cup B) = (!A) \cup (!B).$$

Proof When $A \cap B = \emptyset$ we have $\mu(A \cup B) = \mu A \cup \mu B$. ∎

Note we have $!\{\, \emptyset \,\} = \emptyset$, where the first $\emptyset$ is in $\mathbf{SN}(\mathcal{F})$, the second $\emptyset$ is in $\mathbf{SN}(!\mathcal{F})$.

Of course Definition 8.3.1 specialises down to coherent spaces. For coherent spaces $\mathcal{F}_0$ and $\mathcal{F}_1$ with $A \in \mathbf{KSN}(\mathcal{F}_0)$, $B \in \mathbf{KSN}(\mathcal{F}_1)$, we define

$$A \to B = \{\ x \in \mathcal{F}_0 \to \mathcal{F}_1 \mid A \sqsubseteq_\mu (Pt\, x)^{-1} B\ \}.$$

As a corollary of Theorem 8.3.1, $A \to B \in \mathbf{KSN}(\mathcal{F}_0 \to \mathcal{F}_1)$. We have an isomorphism $\ell : [\mathcal{F}_0 \to_s \mathcal{F}_1] \to [!\mathcal{F}_0 \multimap \mathcal{F}_1]$, as pointed at the end of Section 7.1. Is it also true, under the isomorphism, that $A \to B = (!A) \multimap B$, as one may expect? Of course. Actually this is one of the touchstones to test the correctness of the approach. We have

Proposition 8.4.5 Suppose

$$\ell : [\mathcal{F}_0 \to_s \mathcal{F}_1] \to [!\mathcal{F}_0 \multimap \mathcal{F}_1]$$

is the isomorphism. Then

$$x \in A \to B \text{ iff } \ell x \in (!A) \multimap B,$$

where $A \in \mathbf{KSN}(\mathcal{F}_0)$, $B \in \mathbf{KSN}(\mathcal{F}_1)$.

Proof

$$\begin{aligned} x \in (A \to B) &\iff \mu A \subseteq \mu(Pt\, x)^{-1} B \\ &\iff \{\, \{\, a \,\} \mid a \in \mu A \,\} \subseteq \mu(Pt\, \ell x)^{-1} B \\ &\iff \mu(!A) \subseteq \mu(Pt\, \ell x)^{-1} B \\ &\iff (\ell x) \in (!A) \multimap B \end{aligned}$$

where $\ell x = \{\, (\, \{\, u \,\},\ e\,) \mid (\, u,\ e\,) \in x \,\}$. Note we used the fact that $\mu(!A) = \{\, \{\, a \,\} \mid a \in \mu A \,\}$. ∎

8.5 Partially Synchronous Morphisms and Their Logic (with an eye to CCS)

This section studies the stable-neighbourhood constructions in the categories $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$. Stable event structures can be used to give semantics to languages like CCS and CSP. Building a domain logic on $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$ should help us in understanding logics for CCS and CSP like languages. A first step toward such a logic is to study the stable-neighbourhood constructions corresponding to constructions in $\mathbf{SEV}^*_{syn}$ and $\mathbf{SEV}_{syn}$. Again the sum construction is easy. We focus on the partially synchronous product and synchronous product. First we give some general results on how events and the relationships among them determine stable neighbourhoods. These results lead us to the main result of this section – Theorem 8.5.4. The construction introduced in Theorem 8.5.4, however, does not preserve compactness. This is an unwelcome phenomenon, since it implies that there does not exist a prime normal form for assertions as given in Chapter 5.

Theorem 8.5.1 Let $\underline{E} = (\, E,\ Con,\ \vdash\,)$ be a stable event structure and $X \subseteq Con$ a finite set of consistent events. Define

$$\widehat{X} = \{\, x \in \mathcal{F}(\underline{E}) \mid X \subseteq x \,\}.$$

$\widehat{X}$ is a stable neighbourhood of $(\, \mathcal{F}(\underline{E}),\ \subseteq\,)$.

Proof First we show that $\widehat{X}$ is Scott open. Obviously $\widehat{X}$ is upwards closed. Let

$$x_0 \subseteq x_1 \subseteq \cdots \subseteq x_i \cdots$$

be an increasing chain in $\mathcal{F}(\underline{E})$ such that $\bigcup_{i\in\omega} x_i \in \widehat{X}$. As X is finite, there must be some i such that $X \subseteq x_i$, or $x_i \in \widehat{X}$. Hence $\widehat{X}$ is open.

Let $x \uparrow y$ and $x, y \in \widehat{X}$. We have $x \cap y \in \mathcal{F}(\underline{E})$ as $\underline{E}$ is stable [Wi86]. Also $X \subseteq x \cap y$ as $X \subseteq x$ and $X \subseteq y$. Therefore $\widehat{X}$ is a stable neighbourhood. ∎

When $X = \{ e \}$ we write $\hat{e}$ for $\widehat{\{ e \}}$. In particular $\hat{e}$ is a stable neighbourhood. But it is not necessarily compact. From the work of Winskel [Wi86] we know that the complete primes of $(\mathcal{F}(\underline{E}), \subseteq)$ for a stable event structure $\underline{E}$ are of the form $\lceil e \rceil_x$ where $e \in x$ and

$$\lceil e \rceil_x = \bigcap \{ y \in \mathcal{F}(\underline{E}) \mid e \in y \ \& \ y \subseteq x \}.$$

It is then not difficult to see that the minimal points of $\hat{e}$ are all complete primes.

Now we consider some general constructions on stable neighbourhoods.

Definition 8.5.1 Suppose $A, B \in \mathbf{SN}(D)$, where D is a dI-domain. Define

$$A \ll B = \{ x \in D \mid x \in B \ \& \ [\forall y \sqsubseteq x. y \in B \Rightarrow y \in A] \}.$$

$A \ll B$ reads 'B needs A' or 'A proceeds B'. A computation x has property $A \ll B$ if it has property B and at any earlier stage of the computation if property B is satisfied then property A is also satisfied.

Example 8.5.1 Let D be $\mathcal{F}(\underline{E})$, the dI-domain associated with a stable event structure $\underline{E}$, and let $A = \hat{e}$, $B = \hat{e'}$, where $e, e' \in E$. In this case $x \in (\hat{e} \ll \hat{e'})$ iff $e' \in x$ and for any $y \subseteq x$, $e' \in y$ implies $e \in x$. Comparing this with Definition 7.1.9 we find that $x \in (\hat{e} \ll \hat{e'})$ iff $e \leq_x e'$! Note according to Definition 7.1.9 $e \leq_x e'$ is only defined for e, e' *in* x.

Theorem 8.5.2 Suppose D is a dI-domain and $A, B \in \mathbf{SN}(D)$. Then $A \ll B \in \mathbf{SN}(D)$ and, moreover,

$$(A \ll B) = (A \cap \mu B)\uparrow .$$

Proof First we show that $(A \ll B)$ is upwards closed. Suppose $x \in (A \ll B)$ and $x \sqsubseteq z$. $x \in (A \ll B)$ implies $x \in B$. Hence $z \in B$. Let $y \sqsubseteq z$ and $y \in B$. We want to

show that $y \in A$. x and y are compatible because $x \sqsubseteq z$ and $y \sqsubseteq z$. Hence $x \sqcap y \in B$. Since $x \in (A \ll B)$, $x \sqcap y \subseteq x$, and $x \sqcap y \in B$, we have $x \sqcap y \in A$. Therefore $y \in A$. Namely, $z \in (A \ll B)$.

Next we show that $(A \ll B) = (A \cap \mu B)\uparrow$, which implies $(A \ll B)$ is a stable neighbourhood since it is then clear that $\mu(A \ll B) \subseteq \mu B$.

Let $x \in (A \ll B)$. Then $x \in B$. For some $c \in \mu B$, $c \sqsubseteq x$. Clearly $c \in A$, therefore $c \in (A \cap \mu B)$ and $x \in (A \cap \mu B)\uparrow$. Thus

$$(A \ll B) \subseteq (A \cap \mu B)\uparrow .$$

On the other hand, if $x \in (A \cap \mu B)\uparrow$ then $x \sqsupseteq c$ for some $c \in (A \cap \mu B)$. We have $c \in (A \cap B)$ and for any $y \sqsubseteq c$, $y \in B$ implies $y = c$ since c is in μB, and c is in A. Hence $c \in (A \ll B)$. But $(A \ll B)$ is upwards closed, so $x \in (A \ll B)$. We have shown that

$$(A \ll B) \supseteq (A \cap \mu B)\uparrow .$$

∎

Corollary 8.5.1 Let $\underline{E} = (E, Con, \vdash)$ be a stable event structure. Define, for two events e, e',

$$(e \leq e') = \{ x \mid x \in \mathcal{F}(\underline{E}) \ \& \ e \leq_x e' \}.$$

$(e \leq e')$ is a stable neighbourhood of $\mathcal{F}(\underline{E})$ which is equal to $\widehat{e} \ll \widehat{e'}$.

Abbreviate $(A \ll B) \cap (B \ll A)$ as $A \frown B$. Then a computation x has property $(A \frown B)$ if x has both properties A and B, and at any earlier stage of the computation, property A and property B either hold or not hold, at the *same* time. In other words, a computation x has property $(A \frown B)$ if property A and B start holding *simultaneously* on or before x. We can read $(A \frown B)$ as 'A *synchronous-and* B'.

Definition 8.5.2 Suppose A, $B \in \mathbf{SN}(D)$, where D is a dI-domain. Define

$$A \not\ll B = \{ x \in D \mid x \in B \ \& \ [\exists y \sqsubseteq x.(y \in B \ \& \ y \notin A)] \}.$$

$(A \not\ll B)$ reads 'B independent of A'. Abbreviate $(A \not\ll B) \cap (B \not\ll A)$ as $A \smile B$. Then a computation x has property $(A \smile B)$ if x has both properties A and B, and at some earlier stage in the computation, property B holds but property A does not. Also

there is some earlier stage at which property A holds but property B does not. In other words, a computation x has property $(A \smile B)$ if at x both property A and B hold, and they started holding *independently*. We can read $(A \smile B)$ as 'A *asynchronous-and* B'.

Example 8.5.2 Let D be $\mathcal{F}(\underline{E})$, the dI-domain associated with a stable event structure $\underline{E}$, and let $A = \widehat{e}$, $B = \widehat{e'}$, where e, $e' \in E$. We have $x \in (\widehat{e} \smile \widehat{e'})$ iff $e' \in x$, $e \in x$ and, for some $y \subseteq x$, $e' \in y$ but $e \notin y$, for some $y' \subseteq x$, $e \in y'$ but $e' \notin y'$. Hence $x \in (\widehat{e} \smile \widehat{e'})$ is a similar notion to $e \, co_x \, e'$ which people use to describe that events e and e' can occur *concurrently* in x.

Theorem 8.5.3 Suppose D is a dI-domain and A, $B \in \mathbf{SN}(D)$. Then $A \not\ll B \in \mathbf{SN}(D)$ and, moreover,

$$(A \not\ll B) = (\mu B \setminus A)\uparrow .$$

Proof It is enough to show that $(A \not\ll B) = (\mu B \setminus A)\uparrow$.

Let $x \in (A \not\ll B)$. Then $x \in B$ and for some $y \sqsubseteq x$, $y \in B$ but $y \notin A$. For some $c \in \mu B$, $c \sqsubseteq y$. Clearly $c \notin A$. Therefore $c \in (\mu B \setminus A)$ and $x \in (\mu B \setminus A)\uparrow$. Thus

$$(A \not\ll B) \subseteq (\mu B \setminus A)\uparrow .$$

On the other hand, if $x \in (\mu B \setminus A)\uparrow$ then $x \sqsupseteq c$ for some $c \in (\mu B \setminus A)$. We have $x \in B$ and $c \sqsubseteq x$, $c \in B$ but $c \notin A$. So $x \in (A \not\ll B)$. We have shown that

$$(A \not\ll B) \supseteq (\mu B \setminus A)\uparrow .$$

∎

Let $A \in \mathbf{SN}(\mathcal{F}(\underline{E_0}))$ and $B \in \mathbf{SN}(\mathcal{F}(\underline{E_1}))$, where $\underline{E_0}$ and $\underline{E_1}$ are stable event structures. How can we construct, out of A and B, a reasonable stable neighbourhood for the partially synchronous product $\underline{E_0} \times \underline{E_1}$ given in Definition 7.1.11?

Definition 8.5.3 Suppose $A \in \mathbf{SN}(\mathcal{F}(\underline{E_0}))$ and $B \in \mathbf{SN}(\mathcal{F}(\underline{E_1}))$, where $\underline{E_0}$ and $\underline{E_1}$ are stable event structures. Define

$$(A \leftrightarrow_{ps} B) = \{ \quad x \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \mid \pi_0 x \in A \;\&\; \pi_1 x \in B \;\&$$
$$\forall y \, (y \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \;\&\; y \subseteq x). \; \pi_0 y \in A \Leftrightarrow \pi_1 y \in B \}$$

and

$$(A \leftrightarrow_{s} B) = \{ \quad x \in \mathcal{F}(\underline{E_0} \oplus \underline{E_1}) \mid \pi_0 x \in A \;\&\; \pi_1 x \in B \;\&$$
$$\forall y \, (y \in \mathcal{F}(\underline{E_0} \oplus \underline{E_1}) \;\&\; y \subseteq x). \; \pi_0 y \in A \Leftrightarrow \pi_1 y \in B \}$$

Theorem 8.5.4 For stable event structures $\underline{E_0}$ and $\underline{E_1}$, if $A \in \mathbf{SN}(\mathcal{F}(\underline{E_0}))$ and $B \in \mathbf{SN}(\mathcal{F}(\underline{E_1}))$, then $(A \leftrightarrow_{ps} B) \in \mathbf{SN}(\mathcal{F}(\underline{E_0} \times \underline{E_1}))$. Moreover, if $e_0 \in E_0$ and $e_1 \in E_1$, then $\widehat{e_0} \leftrightarrow_{ps} \widehat{e_1} = \widehat{(e_0, e_1)}$. The same result holds for the synchronous product, that is, if $A \in \mathbf{SN}(\mathcal{F}(\underline{E_0}))$ and $B \in \mathbf{SN}(\mathcal{F}(\underline{E_1}))$, then $(A \leftrightarrow_{s} B) \in \mathbf{SN}(\mathcal{F}(\underline{E_0} \oplus \underline{E_1}))$. Moreover, if $e_0 \in E_0$ and $e_1 \in E_1$, then $\widehat{e_0} \leftrightarrow_{s} \widehat{e_1} = \widehat{(e_0, e_1)}$.

Proof We check the case for the partially synchronous product. The proof for the synchronous product is similar.

Assume $x \in (A \leftrightarrow_{ps} B)$ and $x \subseteq z$, where $z \in \mathcal{F}(\underline{E_0} \times \underline{E_1})$. We have $\pi_0 z \in A$ and $\pi_1 z \in B$, clearly. For any $y \subseteq z$,

$$\begin{aligned} \pi_0 y \in A &\iff^{*} \pi_0(x \cap y) \in A \\ &\iff \pi_1(x \cap y) \in B \qquad (x \cap y \subseteq x \in (A \leftrightarrow_{ps} B)) \\ &\iff \pi_1 y \in B \end{aligned}$$

($*$: we have, in this special case, $\pi_0(x \cap y) = (\pi_0 x) \cap (\pi_0 y)$ when $x \uparrow y$. This is because $\pi_0 e = \pi_0 e' \in (\pi_0 x) \cap (\pi_0 y)$ implies $e = e'$ with e, e' consistent.) Hence $z \in (A \leftrightarrow_{ps} B)$, which means $(A \leftrightarrow_{ps} B)$ is upwards closed.

Let

$$x_0 \subseteq x_1 \subseteq \cdots \subseteq x_n \subseteq \cdots$$

be a chain in $\mathcal{F}(\underline{E_0} \times \underline{E_1})$ such that $\bigcup_{i \in \omega} x_i \in (A \leftrightarrow_{ps} B)$. Obviously there is some x_i for which $\pi_0 x_i \in A$ and $\pi_1 x_i \in B$. It is then easy to see that $x_i \in (A \leftrightarrow_{ps} B)$. So $(A \leftrightarrow_{ps} B)$ is open. From x, $y \in (A \leftrightarrow_{ps} B)$ and $x \uparrow y$ we can easily deduce that $x \cap y \in (A \leftrightarrow_{ps} B)$.

To prove $(\widehat{e_0} \leftrightarrow_{ps} \widehat{e_1}) = \widehat{(e_0, e_1)}$ it is clearly enough to show that $(\widehat{e_0} \leftrightarrow_{ps} \widehat{e_1}) \subseteq \widehat{(e_0, e_1)}$ because the inclusion in the other direction is obvious. Let $x \in (\widehat{e_0} \leftrightarrow_{ps} \widehat{e_1})$. We have $e_0 \in \pi_0 x$ and $e_1 \in \pi_1 x$. Therefore there exist e, $e' \in x$, such that $\pi_0 e = e_0$ and $\pi_1 e' = e_1$. From the coincidence-freeness we deduce that $e = e' = (e_0, e_1)$. ∎

At this point it is appropriate to ask whether all the criteria set at the beginning of section 8.3 are satisfied for the constructions introduced in Definition 8.5.3. From Theorem 8.5.4 we know that for the synchronous product all the three conditions are satisfied. It is easy to see that the first and the third conditions hold. The second condition also holds since Theorem 8.5.4 concludes that it is possible to get the stable neighbourhoods $\widehat{(e_0, e_1)}$. Clearly any prime stable neighbourhood in $\mathbf{SN}(\mathcal{F}(\underline{E_0} \oplus \underline{E_1}))$ is a finite intersection of stable neighbourhoods of the form $\widehat{(e_0, e_1)}$.

For the partially synchronous product the first and the third conditions hold. The second condition does not hold. This is because we cannot get stable neighbourhoods of the form $\widehat{(e_0, *)}$ and $\widehat{(*, e_1)}$ in general. Based on Definition 8.5.2 we can introduce a similar notion as that is given in Definition 8.5.3 to capture the independence of events. Suppose $A \in \mathbf{SN}(\mathcal{F}(\underline{E_0}))$ and $B \in \mathbf{SN}(\mathcal{F}(\underline{E_1}))$, where $\underline{E_0}$ and $\underline{E_1}$ are stable event structures. Define

$$(A \not\ll_{ps} B) = \{ \quad x \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \mid \pi_1 x \in B \;\&$$
$$\exists y\, (y \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \;\&\; y \subseteq x).\; \pi_1 y \in B \;\&\; \pi_0 y \notin A \;\}$$

and

$$(A \not\gg_{ps} B) = \{ \quad x \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \mid \pi_0 x \in A \;\&$$
$$\exists y\, (y \in \mathcal{F}(\underline{E_0} \times \underline{E_1}) \;\&\; y \subseteq x).\; \pi_0 y \in A \;\&\; \pi_1 y \notin B \;\}$$

A stable neighbourhood $\widehat{(e_0, *)}$ consists of all the configurations x for which $(e_0, *) \in x$. This means e_0 has occurred in x but it is independent of any event in E_1. By using the construction just introduced we only have $\widehat{(e_0, *)}$ as

$$\widehat{(e_0, *)} = \bigcap_{e_1 \in E_1} (\widehat{e_0} \not\gg_{ps} \widehat{e_1})$$

(a fact that has a similar proof to that of Theorem 8.5.4) which is not necessarily a finite intersection. This suggests that for the logic we should use not only the finite disjunctions and finite conjunctions, but also limit forms of quantification like

$$\forall e \in E_1.\, (e_0 \text{ independent-of } e)$$

to express the fact that an event $(e_0, *)$ occured in a configuration of the partially synchronous product.

Note that, unfortunately, in the partially synchronous product or synchronous product, a stable neighbourhood $\widehat{(e_0, e_1)}$ is not necessarily compact as the following example shows.

Example 8.5.3 Let $\underline{E_0} = (\{ e, e' \}, Con_0, \vdash_0)$, where $\{ e, e' \} \in Con_0$ and $\{ e \} \vdash_0 e'$, $\emptyset \vdash_0 e$; $\underline{E_1} = (\omega, Con_1, \vdash_1)$, where $X \in Con_1$ for $X \subseteq^{fin} \omega$ and $\emptyset \vdash_1 i$ for every $i \in \omega$. In the partially synchronuos product or synchronous product, $\widehat{(e', 0)}$ is not compact because $\{ (e, i) \}$, $i \geq 1$, are minimal points in $\widehat{(e', 0)}$.

As remarked earlier, this means that there does not exist a prime normal form for assertions of the kind of logic we are looking for. This does not imply, however, that there does not exist a neat logic on $\mathbf{SEV}^*_{syn}$ or $\mathbf{SEV}_{syn}$.

Chapter 9

Logics of DI-Domains

In this chapter we study logics of dI-domains.

A logic of coherent spaces is introduced. The proof system is shown to be sound and complete. The logic of coherent spaces employs an assertion language with a disjoint 'or', to cope with the disjunctive nature of stable neighbourhoods which are used to interpret the assertions. Because of the disjunctive nature the assertions cannot be formulated by a simple grammar directly as has been done tradditionally; instead we have to use a mutual recursion between syntactic rules and proof rules. Since the type constructions allowed are sum, product, tensor product, linear function space, Girard's 'of course' operator, which we call shriek, and recursively defined types, we can combine type constructions to get a logic of $\mathbf{COH}_l$, coherent spaces with linear, stable functions, and a logic of $\mathbf{COH}_s$, coherent spaces with stable functions via the shriek construction (the left adjoint). The logic of $\mathbf{COH}_s$ is then generalised to a logic of **DI**, dI-domains with stable functions. Proof systems are introduced and soundness, completeness, and expressiveness results given.

To give meaning to recursively defined domains in **DI** we use the result of [Zh89]. To solve equations of coherent spaces we introduce, in Section 9.1, *coherent event structures* and a substructure order on them. This gives a large cpo on which various constructions of coherent event structures are shown to induce continuous functions. The fixed-point construction of continuous functions provides solutions to equations of coherent spaces.

9.1 Solving Equations of Coherent Spaces

Coherent spaces can be seen as information systems [Zh89] as well as event structures (chapter 7). It is easy to see that the configurations of event structures of the form $(E, Con, \{\, \emptyset \vdash e \mid e \in E \,\})$ determine coherent spaces, where Con is specified by a conflict relation. Therefore a pair $(E, \#)$ determines a coherent space (here we choose to insist that $\#$ be irreflective). Call a structure $(E, \#)$ with $\#$ symmetric, irreflective a *coherent event structure*. This name, we admit, is rather superficial: coherent event

structures are exactly coherent spaces. It is used here since by viewing coherent spaces as a special kind of event structures we can directly apply results on recursively defined event structures.

There is some technical advantage to work with structures $(E, \mathbb{W})$, derived from $(E, \#)$ by taking $\mathbb{W}$ to be $\# \cup \mathbf{1}$, with $\mathbf{1}$ the identity relation. However one can easily recover $(E, \#)$ from $(E, \mathbb{W})$ by taking $\#$ to be $\mathbb{W} \setminus \mathbf{1}$.

We introduce a partial order of coherent event structures. This order captures the notation of rigid embedding. It enables us to give meanings to recursively defined coherent event structures through the construction of least fixed points for continuous functions. This section uses the idea given in section 1.6 of [Wi86] where relevant proofs can be found.

Definition 9.1.1 Let $\underline{E_0} = (E_0, \mathbb{W}_0)$ and $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures. Define $\underline{E_0} \trianglelefteq \underline{E_1}$ if

$$E_0 \subseteq E_1 \quad \text{and}$$
$$e \mathbb{W}_0 e' \iff e, e' \in E_0 \ \& \ e \mathbb{W}_1 e'$$

When $\underline{E_0} \trianglelefteq \underline{E_1}$ we call $\underline{E_0}$ a *substructure* of $\underline{E_1}$. It is easy to see that $\underline{E_0} \trianglelefteq \underline{E_1}$ iff

$$E_0 \subseteq E_1 \quad \text{and}$$
$$e \#_0 e' \iff e, e' \in E_0 \ \& \ e \#_1 e'.$$

This is because we have $E_0 \subseteq E_1$, so $e = e'$ in E_0 is certainly equivalent to $e = e'$ in E_1 and $e, e' \in E_0$. There is a least coherent event structure, the unique one with the emptyset of events. Each $\omega-$chain of coherent event structures, increasing with respect to $\trianglelefteq$ has a least upper bound, with events, and conflicting relations the union of those in the chain.

Definition 9.1.2 Let D, E be coherent families (See Chapter 7). A stable function $f : D \to E$ is a *rigid embedding* if there is a stable function $g : E \to D$ called a *projection* such that

- $\forall d \in D.\ gf(d) = d$
- $\forall e \in E.\ fg(e) \subseteq e$
- $\forall d \in D,\ e \in E\ .\ e \subseteq f(d) \Longrightarrow fg(e) = e$

Proposition 9.1.1 Let $\underline{E} = (E, \mathbb{W})$ be a coherent event structure. Then $Pt\underline{E}$ is a

coherent family, where

$$Pt\underline{E} = \{ x \mid x \subseteq E \;\&\; \forall e, e' \in x.\; e \mathbb{W} e' \Rightarrow e = e' \}.$$

Moreover, if $\underline{E} \trianglelefteq \underline{E}'$ then the inclusion map $i : Pt\underline{E} \to Pt\underline{E}'$ is a rigid embedding with projection $j : Pt\underline{E}' \to Pt\underline{E}$ given by $j(y) = y \cap E$ for $y \in Pt\underline{E}'$.

Proposition 9.1.2 The relation $\trianglelefteq$ is a partial order on coherent event structures. It has a least coherent event structure $\underline{\perp} =_{def} (\emptyset, \emptyset)$. An ω–chain of coherent event structures $\underline{E}_0 \trianglelefteq \underline{E}_1 \cdots \trianglelefteq \underline{E}_n \trianglelefteq \cdots$ where $\underline{E}_n = (E_n, \mathbb{W}_n)$ has a least upper bound

$$\bigcup_{n\in\omega} \underline{E}_n = (\bigcup_{n\in\omega} E_n, \bigcup_{n\in\omega} \mathbb{W}_n).$$

Coherent event structures form a class and not a set. For this reason we cannot say that coherent event structures form a cpo. However it has all the other properties of a cpo which are enough to serve our purpose. We can call coherent event structures with $\trianglelefteq$ a *large cpo* and write **COEV** for it. It is easy to extend the substructure relation to n–tuples of coherent event structures. They form a large cpo, $\mathbf{COEV}^n$, too.

Definition 9.1.3 Write π_j for the *projection* map $\pi_j(\underline{E}_0, \ldots, \underline{E}_{n-1}) = \underline{E}_j$ on n–tuples of coherent event structures. For n–tuples,

$$(\underline{E}_0, \ldots, \underline{E}_{n-1}) \trianglelefteq (\underline{E}'_0, \ldots, \underline{E}'_{n-1}) \text{ if } \underline{E}_0 \trianglelefteq \underline{E}'_0 \;\&\; \cdots \;\&\; \underline{E}_{n-1} \trianglelefteq \underline{E}'_{n-1}.$$

For convenience write $\vec{\underline{E}}$ for $(\underline{E}_0, \underline{E}_1, \cdots \underline{E}_n)$.

The least element of $\mathbf{COEV}^n$ is the n–tuple of empty coherent event structures $(\underline{\perp}, \underline{\perp}, \cdots \underline{\perp})$. The least upper bound of an ω–chain of n–tuples of coherent event structures is then just the n–tuple of coherent event structures consisting of the least upper bounds on each component, i.e. if

$$\vec{\underline{E}}_0 \trianglelefteq \vec{\underline{E}}_1 \cdots \trianglelefteq \vec{\underline{E}}_i \trianglelefteq \cdots$$

is a chain then for the j-th component

$$\pi_j(\bigcup_i \vec{\underline{E}}_i) = \bigcup_i \pi_j(\vec{\underline{E}}_i).$$

An operation F from n–tuples of coherent event structures to m–tuples of coherent event structures is said to be continuous if it is monotonic, i.e. $\vec{\underline{E}} \trianglelefteq \vec{\underline{E}}'$ implies $F(\vec{\underline{E}}) \trianglelefteq$

$F(\vec{\underline{E}'})$ and preserves ω–increasing chains of coherent event structures, i.e.

$$\vec{\underline{E}}_0 \trianglelefteq \vec{\underline{E}}_1 \cdots \trianglelefteq \vec{\underline{E}}_i \trianglelefteq \cdots$$

implies

$$\bigcup_i F(\vec{\underline{E}}_i) = F(\bigcup_i \vec{\underline{E}}_i).$$

It is well known that for functions on (finite) tuples of cpos they are continuous iff by changing (any) one argument while fixing others the induced function is continuous. Thus in verifying that an operation is monotonic or continuous we ultimately have to show certain unary operations are continuous with respect to the substructure relation $\trianglelefteq$. The next proposition will be a great help in proving operations continuous.

Proposition 9.1.3 An unary operation F is continuous iff it is monotonic with respect to $\trianglelefteq$ and *continuous on events*, i.e. for any ω–chain

$$\underline{E}_0 \trianglelefteq \underline{E}_1 \cdots \trianglelefteq \underline{E}_i \trianglelefteq \cdots,$$

each event of $F(\bigcup_i \underline{E}_i)$ is an event of $\bigcup_i F(\underline{E}_i)$.

Proof

($\Rightarrow$): obvious.

($\Leftarrow$): Let

$$\underline{E}_0 \trianglelefteq \underline{E}_1 \cdots \trianglelefteq \underline{E}_i \trianglelefteq \cdots$$

be an ω–chain of coherent event structures. Since F is monotonic, we clearly have

$$\bigcup_i F(\underline{E}_i) \trianglelefteq F(\bigcup_i \underline{E}_i).$$

Thus the events of $F(\bigcup_i \underline{E}_i)$ are the same as the events of $\bigcup_i F(\underline{E}_i)$. Therefore they are the same coherent event structure since for coherent event structures $\underline{E} = (E, \Vdash)$ and $\underline{E}' = (E', \Vdash')$, $E = E'$ and $\underline{E} \trianglelefteq \underline{E}'$ implies $\underline{E} = \underline{E}'$, a fact that can be easily verified. ∎

It is well-known that continuous functions on cpos have least fixed points and the argument is virtually the same for continuous operations on big cpos. Now given any continuous function F on **COEV**, we can get the least fixed point of F, which is the limit of the increasing ω–chain

$$\underline{\perp} \trianglelefteq F(\underline{\perp}) \trianglelefteq F^2(\underline{\perp}) \trianglelefteq \cdots \trianglelefteq F^n(\underline{\perp}) \trianglelefteq \cdots,$$

i.e. $\bigcup_i F^i(\underline{\perp})$. Note since we are working with the partial order $\trianglelefteq$, we get an equality

$$F(\bigcup_i F^i(\underline{\perp})) = \bigcup_i F^i(\underline{\perp}).$$

Thus we have

Proposition 9.1.4 Let F be a continuous function on **COEV**. The coherent event structure $fix\ F$ of **COEV** is the least fixed point of F, where

$$fix\ F = \bigcup_i F^i(\underline{\perp}).$$

There are the following constructions on coherent event structures.

Sum. Let $\underline{E_0} = (E_0, \mathbb{W}_0)$, $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures. Their sum, $\underline{E_0} + \underline{E_1}$, is a structure $\underline{E} = (E, \mathbb{W})$ where

$$E = \{0\} \times E_0 \cup \{1\} \times E_1$$

$$(i, e_0) \# (j, e_1) \iff i \neq j \text{ or } (i = j \ \&\ e_0 \#_i e_1).$$

Product. Let $\underline{E_0} = (E_0, \mathbb{W}_0)$, $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures. Their product, $\underline{E_0} \times \underline{E_1}$, is a structure $\underline{E} = (E, \mathbb{W})$ where

$$E = \{0\} \times E_0 \cup \{1\} \times E_1$$

$$(i, e_0) \mathbb{W} (j, e_1) \iff i = j \ \&\ e_0 \mathbb{W}_i e_1.$$

Tensor product. Let $\underline{E_0} = (E_0, \mathbb{W}_0)$, $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures. Their tensor product, $\underline{E_0} \otimes \underline{E_1}$, is a structure $\underline{E} = (E, \mathbb{W})$ where

$$E = E_0 \times E_1$$

$$(e_0, e_1) \# (e_0', e_1') \iff e_0 \#_0 e_0' \text{ or } e_1 \#_1 e_1'.$$

Linear function space. Let $\underline{E_0} = (E_0, \mathbb{W}_0)$, $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures. The linear function space, $\underline{E_0} \multimap \underline{E_1}$, is a structure $\underline{E} = (E, \mathbb{W})$ where

$$E = E_0 \times E_1$$

$$(e_0, e_1) \mathbb{W} (e_0', e_1') \iff \neg (e_0 \#_0 e_0') \ \&\ e_1 \mathbb{W}_1 e_1'.$$

Shriek. Let $\underline{E_0} = (E_0, \mathbb{W}_0)$ be a coherent event structure. Its shriek, $!\underline{E_0}$, is a structure $\underline{E} = (E, \mathbb{W})$ where

$$E = \mathrm{Fin}(E_0)$$

$$a \# a' \iff \exists e \in a\, \exists e' \in a'.\ e \#_0 e'.$$

Recall that $\mathrm{Fin}(A)$ stands for the set of finite subsets of A, a notation introduced at the beginning of Chapter 4.

Theorem 9.1.1 below asserts that all the constructions introduced above induce continuous functions. This implies that it is possible to give meanings to recursively defined coherent event structures involving these constructions.

Theorem 9.1.1 Shriek is a continuous function $!(\ ) : \mathbf{COEV} \to \mathbf{COEV}$. Sum, product, tensor product, and linear function space

$$+,\ \times,\ \otimes,\ \multimap\ :\ \mathbf{COEV}^2 \to \mathbf{COEV}$$

are also continuous functions.

Proof That all the constructions give well defined functions is clear. We prove that shriek and linear function space are continuous. The proof for the rest of the constructions are similar. By Proposition 9.1.2, to show a construction is continuous it is enough to show that it is monotonic and continuous on events.

To show that shriek is monotonic let $\underline{E_0} = (E_0, \mathbb{W}_0)$, $\underline{E_1} = (E_1, \mathbb{W}_1)$ be coherent event structures such that $\underline{E_0} \trianglelefteq \underline{E_1}$. Let $!\underline{E_0} = (E, \mathbb{W})$ and $!\underline{E_1} = (E', \mathbb{W}')$. Clearly, then, $E \subseteq E'$. By definition, $a \# b$ iff there are $e \in a$, $f \in b$ such that $e \#_0 f$; $a' \#' b'$ iff there are $e' \in a'$, $f' \in b'$ such that $e' \#_1 f'$. Thus $a \# b$ iff $a, b \in E$ and $a \#' b$ since $\underline{E_0} \trianglelefteq \underline{E_1}$. Therefore $!\underline{E_0} \trianglelefteq !\underline{E_1}$.

Now let

$$\underline{E_0} \trianglelefteq \underline{E_1} \cdots \trianglelefteq \underline{E_i} \trianglelefteq \cdots$$

be a chain of coherent event structures. Suppose a is an event of $!(\bigcup_i \underline{E_i})$. We have $a \subseteq^{fin} \bigcup_i E_i$. Thus $a \subseteq^{fin} E_n$ for some n since a is a finite set. Therefore a is an event of $\bigcup_i\, !(\underline{E_i})$. This means $!$ is continuous on events.

We have proved that $!$ is continuous.

The linear function space, $\multimap$, is monotonic: let $\underline{E_0} = (E_0,\ \mathbb{W}_0)$, $\underline{E_1} = (E_1,\ \mathbb{W}_1)$, and $\underline{E_2} = (E_2,\ \mathbb{W}_2)$ be coherent event structures such that $\underline{E_0} \trianglelefteq \underline{E_1}$. Write $(E,\ \mathbb{W})$ for $\underline{E_0} \multimap \underline{E_2}$ and $(E',\ \mathbb{W}')$ for $\underline{E_1} \multimap \underline{E_2}$. Clearly $E \subseteq E'$. By definition, $(e_0,\ e_2)\,\mathbb{W}\,(e_0',\ e_2')$ iff $\neg(e_0 \#_0 e_0')$ and $e_2\,\mathbb{W}_2\,e_2'$; $(e_1,\ e_2)\,\mathbb{W}'\,(e_1',\ e_2')$ iff $\neg(e_1 \#_1 e_1')$ and $e_2\,\mathbb{W}_2\,e_2'$. Thus $(e_0,\ e_2)\,\mathbb{W}\,(e_0',\ e_2')$ iff $(e_0,\ e_2),\ (e_0',\ e_2') \in E$ and $(e_0,\ e_2)\,\mathbb{W}'\,(e_0',\ e_2')$ since $\underline{E_0} \trianglelefteq \underline{E_1}$. Therefore

$$(\underline{E_0} \multimap \underline{E_2}) \trianglelefteq (\underline{E_1} \multimap \underline{E_2}).$$

Similarly $\multimap$ is monotonic in its second argument.

Let

$$\underline{E_0} \trianglelefteq \underline{E_1} \cdots \trianglelefteq \underline{E_i} \trianglelefteq \cdots$$

be a chain of coherent event structures. Suppose $(e_0,\ e)$ is an event of $(\bigcup_i \underline{E_i}) \multimap \underline{E}$, with $\underline{E} = (E,\ \mathbb{W})$. We have $(e_0,\ e) \in (\bigcup_i E_i) \times E$. Thus $(e_0,\ e) \in E_n \times E$ for some n. Therefore $(e_0,\ e)$ is an event of $\bigcup_i(\underline{E_i} \multimap \underline{E})$. This means $\multimap$ is continuous on events in its first argument. Similarly it is continuous on events in the second argument.

Therefore $\multimap$ is continuous on both its first and second argument, and hence it is continuous. ∎

We remark that similar results hold for coherent families, in other words, coherent families form a big cpo under a substructure order and all the constructions given in Chapter 7 for coherent families induce continuous functions. Therefore we can also solve equations of coherent families involving constructions like sum, product, tensor product, linear function space, and shriek.

Note that since we use concrete structures of sets to solve equations of coherent event structures the usual domain isomorphism becomes an equality.

9.2 Disjunctive Assertions and Proof System

There is a language of type expressions for coherent spaces introduced as follows:

$$\sigma ::= O \mid \sigma \times \tau \mid \sigma \otimes \tau \mid \sigma + \tau \mid \sigma \multimap \tau \mid !\sigma \mid t \mid rec\, t.\sigma$$

where t is a type variable and σ, τ ranges over type expressions.

Each closed type expression can be interpretd as a coherent space described as follows. The atomic type expression O is interpretd as the two point coherent space $\mathcal{O}$. To fully determine the interpretation it is enough to specify the interpretation for the type constructors. $\times$ is interpretd as the cartesian product, $\otimes$ the tensor product, $+$ the coproduct or sum, $\multimap$ the linear function space, $!$ the shriek construction on coherent spaces, and $rec\, t.\sigma$ the initial solution of the corresponding equation in the category of coherent spaces. Write $\mathcal{D}(\sigma)$ for the coherent space associated with σ specified in this way.

For each type expression we associate it with an assertion language. To characterise coherent spaces the assertions are interpreted as compact stable neighbourhoods of the coherent space determined by the type expression. Thus it is intended that the assertions be disjunctive, in other words, when $\varphi \vee \psi$ appear as an assertion it should be provable from the proof system that $\varphi \wedge \psi \leq \mathbf{f}$. There is actually a weaker notion than this, which requires that if $\varphi \vee \psi$ is an assertion then $[\![\varphi \wedge \psi]\!] \subseteq \emptyset$. However since our proof system is complete (Theorem 9.2.2), these two definitions coincide.

Surprisingly, such a simple requirement of disjunctiveness makes it impossible to specify the assertion language solely by a simple grammar. In what follows we introduce the assertion language by first introducing atomic assertions (tokens) and a conflict relation on the atomic assertions, then giving syntactic rules and proof rules at the same time, with a mutual recursion between them. We do not get all assertions immediately. At the beginning there are only the atomic assertions. But when it is provable from the system that $\varphi \wedge \psi \leq \mathbf{f}$, $\varphi \vee \psi$ becomes a proper assertion and we can prove more facts about these assertions, and can get even more assertions.

Notation. In this chapter all the index set like I are finite.

Atomic Assertions

$$\mathbf{A}_O(\top)$$

$$\frac{\mathbf{A}_\sigma(\varphi)}{\mathbf{A}_{\sigma+\tau}(\dot{inl}\varphi)} \qquad \frac{\mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma+\tau}(\dot{inr}\psi)}$$

$$\frac{\mathbf{A}_\sigma(\varphi)}{\mathbf{A}_{\sigma\times\tau}(\varphi \times \mathbf{t}_\tau)} \qquad \frac{\mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma\times\tau}(\mathbf{t}_\sigma \times \psi)}$$

$$\frac{\mathbf{A}_\sigma(\varphi) \qquad \mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma\otimes\tau}(\varphi \otimes \psi)}$$

$$\frac{\forall i,\, j \in I.\ \mathbf{A}_\sigma(\varphi_i)\ \&\ \neg(\varphi_i \,\#\, \varphi_j)}{\mathbf{A}_{!\sigma}(!\bigwedge_{i\in I}\varphi_i)}$$

$$\frac{\mathbf{A}_\sigma(\varphi) \qquad \mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma\multimap\tau}(\varphi \multimap \psi)}$$

These rules are self explanatory. $\top$ is an atomic assertion of O which, by the semantic interpretation given later, 'picks up' the top element of the domain $\mathcal{O}$. As a special case for the atomic rule, we have

$$\mathbf{A}_{!\sigma}(\ !(\mathbf{t}_\sigma)\ ),$$

which means $!\mathbf{t}_\sigma$ is an atomic assertion of type $!\sigma$. When the type is clear from the context, we just write $\mathbf{A}(\varphi)$. The following table specifies the inconsistency relation between atomic assertions. All assertions here are assumed to be atomic.

Atomic Inconsistency

$$\dot{\mathit{inl}}\varphi \# \dot{\mathit{inr}}\psi$$

$$\frac{\varphi \# \varphi'}{\dot{\mathit{inl}}\varphi \# \dot{\mathit{inl}}\varphi'} \qquad \frac{\psi \# \psi'}{\dot{\mathit{inr}}\psi \# \dot{\mathit{inr}}\psi'}$$

$$\frac{\varphi \# \varphi'}{\varphi \times \mathbf{t} \# \varphi' \times \mathbf{t}} \qquad \frac{\psi \# \psi'}{\mathbf{t} \times \psi \# \mathbf{t} \times \psi'}$$

$$\frac{\varphi \# \varphi'}{\varphi \otimes \psi \# \varphi' \otimes \psi'} \qquad \frac{\psi \# \psi'}{\varphi \otimes \psi \# \varphi' \otimes \psi'}$$

$$\frac{\exists i \in I\, \exists j \in J.\ \varphi_i \# \psi_j}{! \bigwedge_{i \in I} \varphi_i \# ! \bigwedge_{j \in J} \psi_j}$$

$$\frac{\neg(\varphi \# \varphi') \qquad \psi \# \psi'}{\varphi \multimap \psi \# \varphi' \multimap \psi'} \qquad \frac{\neg(\varphi \# \varphi') \qquad \varphi \not\sim \varphi' \qquad \psi \sim \psi'}{\varphi \multimap \psi \# \varphi' \multimap \psi'}$$

On atomic assertions the *similarity relation* $\sim$ almost means syntactic equality ($\equiv$). The subtlety comes from the atomic assertions of type $!\sigma$. By the rules for atomic assertions, an atomic assertion of $!\sigma$ is of the form $! \bigwedge_{i \in I} \varphi_i$, where φ_i's are atomic assertions of type σ. We want to have $!(\varphi_0 \wedge \varphi_1) \sim !(\varphi_1 \wedge \varphi_0)$, though. Therefore, formally $\sim$ is the syntactic equality with every assertion $! \bigwedge_{i \in I} \varphi_i$ of type $!\sigma$ understood as $!(\{\, \varphi_i \mid i \in I \,\})$. It is easy to see from the proof system that the similarity relation is a relation stronger than logical equivalence ($=$). Clearly, if two atomic assertions are not similar, $\varphi \not\sim \psi$, then they do not have the same interpretation as stable neighbourhoods.

Assertions

$$\mathbf{t}, \mathbf{f} : \sigma \qquad \frac{\mathbf{A}_\sigma(\varphi)}{\varphi : \sigma}$$

$$\frac{\varphi,\ \psi : \sigma}{\varphi \wedge \psi : \sigma} \qquad \frac{\varphi \wedge \psi \leq \mathbf{f} \qquad \varphi,\ \psi : \sigma}{\varphi \vee \psi : \sigma}$$

$$\frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \times \psi : \sigma \times \tau} \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \otimes \psi : \sigma \otimes \tau}$$

$$\frac{\varphi : \sigma}{\dot{\imath}nl\varphi : \sigma + \tau} \qquad \frac{\psi : \tau}{\dot{\imath}nr\psi : \sigma + \tau}$$

$$\frac{\varphi : \sigma}{!\varphi : !\sigma} \qquad \frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \multimap \psi : \sigma \multimap \tau}$$

$$\frac{\varphi : \sigma[rec\, t.\sigma/t]}{\varphi : rec\, t.\sigma}$$

The rule that ensures disjunctiveness is

$$\frac{\varphi \wedge \psi \leq \mathbf{f} \qquad \varphi,\ \psi : \sigma}{\varphi \vee \psi : \sigma},$$

which says $\varphi \vee \psi$ is an assertion if we can prove $\varphi \wedge \psi \leq \mathbf{f}$ using the rules given in the forthcoming tables. The assertion language is then the minimal set of assertions inductively closed under these rules. Therefore, if $\varphi \vee \psi$ is an assertion we have $\vdash \varphi \wedge \psi \leq \mathbf{f}$. Write $\mathcal{B}_\sigma$ for the set of disjunctive assertions of type σ built up this way.

When $\varphi : \sigma$ we say φ is well-formed. For example $\mathbf{t} \vee \mathbf{f}$ is a well-formed assertion but $\mathbf{t} \vee \mathbf{t}$ is not. We assume that all assertions from now on are well-formed whenever they appear in the proof system.

Propositional Rules

$$(\mathbf{t})\quad \varphi \leq \mathbf{t} \qquad (\mathbf{f})\quad \mathbf{f} \leq \varphi$$

$$(\text{Ref})\quad \varphi \leq \varphi \qquad (\#)\quad \frac{\varphi \# \psi}{\varphi \wedge \psi \leq \mathbf{f}}$$

$$(\text{Trans})\quad \frac{\varphi \leq \varphi' \quad \varphi' \leq \varphi''}{\varphi \leq \varphi''}$$

$$(\leq - =)\quad \frac{\varphi \leq \psi \quad \psi \leq \varphi}{\varphi = \psi}$$

$$(= - \leq)\quad \frac{\varphi = \varphi'}{\varphi \leq \varphi'} \qquad \frac{\varphi = \varphi'}{\varphi' \leq \varphi}$$

$$(\wedge - \leq)\quad \varphi \wedge \varphi' \leq \varphi \qquad \varphi \wedge \varphi' \leq \varphi'$$

$$(\leq - \wedge)\quad \frac{\varphi \leq \varphi' \quad \varphi \leq \varphi''}{\varphi \leq \varphi' \wedge \varphi''}$$

$$(\vee - \leq)\quad \frac{\varphi \leq \varphi' \quad \psi \leq \varphi'}{\varphi \vee \psi \leq \varphi'}$$

$$(\leq - \vee)\quad \varphi \leq \varphi' \vee \varphi \qquad \varphi' \leq \varphi' \vee \varphi$$

$$(\wedge - \vee)\quad \varphi \wedge (\varphi_1 \vee \varphi_2) \leq (\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2)$$

Note it is necessary to require that all the assertions appear in a rule be well-formed when we applying the rule. For example, with respect to the following axiom

$$\varphi \wedge (\varphi_1 \vee \varphi_2) \leq (\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2)$$

the fact that $(\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2)$ is well-formed does not imply $\varphi_1 \vee \varphi_2$ is also well formed: Simply take the assertion to be $(\mathbf{f} \wedge \mathbf{t}) \vee (\mathbf{f} \wedge \mathbf{t})$. This is a well-formed assertion, but not $\mathbf{t} \vee \mathbf{t}$.

The proof systems associated with sum and product are the same as those given in Chapter 5 for **SFP** objects. So no explanation is needed for them.

Sum

$$(\dot{inl}-\leq)\quad \frac{\varphi \leq_\sigma \psi}{\dot{inl}\,\varphi \leq_{\sigma+\tau} \dot{inl}\,\psi}$$

$$(\dot{inr}-\leq)\quad \frac{\varphi \leq_\tau \psi}{\dot{inr}\,\varphi \leq_{\sigma+\tau} \dot{inr}\,\psi}$$

$$(\dot{inl}-\wedge)\quad \dot{inl}\,(\bigwedge_{i\in I}\varphi_i) =_{\sigma+\tau} \bigwedge_{i\in I}\dot{inl}\,\varphi_i$$

$$(\dot{inr}-\wedge)\quad \dot{inr}\,(\bigwedge_{i\in I}\psi_i) =_{\sigma+\tau} \bigwedge_{i\in I}\dot{inr}\,\psi_i$$

$$(\dot{inl}-\vee)\quad \dot{inl}\,(\bigvee_{i\in I}\varphi_i) =_{\sigma+\tau} \bigvee_{i\in I}\dot{inl}\,\varphi_i$$

$$(\dot{inr}-\vee)\quad \dot{inr}\,(\bigvee_{i\in I}\psi_i) =_{\sigma+\tau} \bigvee_{i\in I}\dot{inr}\,\psi_i$$

Note that from the rules for Atomic Inconsistency we have $\dot{inl}\varphi \,\#\, \dot{inr}\psi$, and by (#) of the Propositional Rules, we can derive (by the prime normal form theorem later) the rule

$$\frac{(\varphi)\!\downarrow \qquad (\psi)\!\downarrow}{\dot{inl}\varphi \wedge \dot{inr}\psi}$$

where $(\ )\!\downarrow$ is a notation introduced just after the proof systems.

Product

$$(\times-\leq)\quad \frac{\psi \leq_\sigma \psi' \quad \varphi \leq_\tau \varphi'}{\psi \times \varphi \leq_{\sigma\times\tau} \psi' \times \varphi'}$$

$$(\times-\vee)\quad \varphi \times (\bigvee_{i\in I}\psi_i) =_{\sigma\times\tau} \bigvee_{i\in I}\varphi \times \psi_i$$

$$(\bigvee_{i\in I}\varphi_i) \times \psi =_{\sigma\times\tau} \bigvee_{i\in I}\varphi_i \times \psi$$

$$(\times-\wedge)\quad \bigwedge_{i\in I}\varphi_i \times \psi_i =_{\sigma\times\tau} (\bigwedge_{i\in I}\varphi_i) \times (\bigwedge_{i\in I}\psi_i)$$

The proof systems for tensor product, shriek, and linear function space are introduced below.

Tensor Product

$$(\otimes - \mathbf{t}) \qquad \frac{\mathbf{P}(\varphi)}{\varphi \otimes \mathbf{t}_\tau =_{\sigma\otimes\tau} \mathbf{t}_{\sigma\otimes\tau}} \qquad \frac{\mathbf{P}(\psi)}{\mathbf{t}_\sigma \otimes \psi =_{\sigma\otimes\tau} \mathbf{t}_{\sigma\otimes\tau}}$$

$$(\otimes - \leq) \qquad \frac{\psi \leq_\sigma \psi' \quad \varphi \leq_\tau \varphi'}{\psi \otimes \varphi \leq_{\sigma\otimes\tau} \psi' \otimes \varphi'}$$

$$(\otimes - \wedge) \qquad \varphi \otimes (\psi_1 \wedge \psi_2) =_{\sigma\otimes\tau} \varphi \otimes \psi_1 \wedge \varphi \otimes \psi_2$$

$$(\varphi_1 \wedge \varphi_2) \otimes \psi =_{\sigma\otimes\tau} \varphi_1 \otimes \psi \wedge \varphi_2 \otimes \psi$$

$$(\otimes - \vee) \qquad \varphi \otimes \bigvee_{i\in I} \psi_i =_{\sigma\otimes\tau} \bigvee_{i\in I} \varphi \otimes \psi_i$$

$$(\bigvee_{i\in I} \varphi_i) \otimes \psi =_{\sigma\otimes\tau} \bigvee_{i\in I} \varphi_i \otimes \psi$$

P is a predicate which holds on prime assertions – the definition is given shortly. Note the special property of tensor product: for certain assertions if one of the component is **t** then the whole assertion is **t**. As a special case of axiom $(\otimes - \vee)$ we have $\varphi \otimes \mathbf{f} = \mathbf{f}$ and $\mathbf{f} \otimes \psi = \mathbf{f}$.

There is only one axiom for shriek. In particular we have $!\mathbf{f} = \mathbf{f}$. We do not have, however,

$$!(\varphi_0 \wedge \varphi_1) = !(\varphi_0) \wedge !(\varphi_1),$$

neither $!\mathbf{t} = \mathbf{t}$.

Shriek

$$(! - \vee) \qquad !(\bigvee_{i\in I} \varphi_i) =_{!\sigma} \bigvee_{i\in I} !(\varphi_i)$$

$$(! - \leq) \qquad \frac{\varphi =_\sigma \psi}{!\varphi =_{!\sigma} !\psi}$$

Note that we do not have

$$\frac{\varphi \leq \psi}{!\varphi \leq !\psi}$$

in general. This is because of the particular way in which the shriek construction works.

Linear Function Space

$$(\multimap -\mathbf{f}) \qquad \frac{\varphi\downarrow}{(\varphi \multimap \mathbf{t}_\tau) \leq_{\sigma\multimap\tau} \mathbf{f}_{\sigma\multimap\tau}}$$

$$(\multimap) \qquad \frac{\lceil\varphi\rceil \neq \lceil\varphi'\rceil \qquad \mathbf{P}(\varphi \wedge \varphi') \qquad \mathbf{P}(\psi)}{(\varphi \multimap \psi) \wedge (\varphi' \multimap \psi) \leq_{\sigma\multimap\tau} \mathbf{f}_{\sigma\multimap\tau}}$$

$$(\multimap - \leq) \qquad \frac{a \in \lceil\psi\rceil \qquad \mathbf{P}(\varphi)}{\varphi \multimap \psi \leq_{\sigma\multimap\tau} \bigvee_{b\in\lceil\varphi\rceil}(b \multimap a)}$$

$$(\multimap -\wedge) \qquad \bigwedge_{i\in I}(\varphi_i \multimap \psi_i) \leq_{\sigma\multimap\tau} (\bigwedge_{i\in I}\varphi_i) \multimap (\bigwedge_{i\in I}\psi_i)$$

$$(\multimap -\vee) \qquad \frac{\mathbf{P}(\varphi)}{\varphi \multimap (\bigvee_{i\in I}\psi_i) =_{\sigma\multimap\tau} \bigvee_{i\in I}(\varphi \multimap \psi_i)}$$

$$(\bigvee_{i\in I}\varphi_i) \multimap \psi =_{\sigma\multimap\tau} \bigwedge_{i\in I}(\varphi_i \multimap \psi)$$

As special cases of the axioms and rules for linear function space, we have

$$\mathbf{f} \multimap \psi = \mathbf{t},$$

$$\mathbf{t} \leq \mathbf{t} \multimap \mathbf{t},$$

and $\varphi \multimap \mathbf{f} = \mathbf{f}$ for a prime assertion φ. Note that by $(\multimap - \leq)$,

$$\mathbf{t} \multimap \psi = \mathbf{f}$$

for ψ convergent (i.e. $(\psi)\downarrow$); by $(\multimap -\mathbf{f})$,

$$\varphi \multimap \mathbf{t} = \mathbf{f}$$

provided $\varphi\downarrow$. These two facts are somehow dual of those for tensor product, where we have

$$\mathbf{t} \otimes \psi = \mathbf{t},$$

$$\varphi \otimes \mathbf{t} = \mathbf{t}$$

under certain conditions.

Note that stable function space $\sigma \to_s \tau$ can be given by the type $!\sigma\multimap\tau$. By Proposition 8.4.5, we can use an assertion $!\varphi \multimap \psi$ of $!\sigma \multimap \tau$ to express an assertion $\varphi \to \psi$ of type $\sigma \to_s \tau$. Therefore our proof system can also treat stable function space.

In giving the axioms and rules we used some side conditions like $\mathbf{P}(\varphi)$, read φ is *prime*, $\varphi\!\downarrow$, read φ is *convergent*, and an operator called the *atomizer*, which returns a set of atomic assertions $\lceil\varphi\rceil$ not similar to one another, for a prime assertion φ. This practice is not new; we have seen similar side conditions used in predicate calculus and lambda calculus, like ' x does not occur free in M '. Those conditions are usually purely syntactic and easily checked before applying a rule. The following are formal definitions of these syntactic predicates.

$$\mathbf{P}(\varphi) \Longleftrightarrow^{\text{def}} \varphi \equiv \bigwedge_{i\in I} \varphi_i \;\&\; \forall i \in I.\, \mathbf{A}(\varphi_i) \;\&\; \forall i, j \in I.\, \neg(\varphi_i \# \varphi_j)$$

$$\varphi\!\downarrow \Longleftrightarrow^{\text{def}} \mathbf{P}(\varphi) \;\&\; \varphi \equiv \bigwedge_{i\in I} \varphi_i \;\&\; I \neq \emptyset$$

Let φ be a prime assertion, i.e. $\mathbf{P}(\varphi)$. Then there are assertions φ_i, $i \in I$, such that $\varphi \equiv \bigwedge_{i\in I} \varphi_i$ and $\forall i \in I.\, \mathbf{A}(\varphi_i)$. We define $\lceil\varphi\rceil = \{\, \varphi_i \mid i \in I \,\}/\!\sim$.

We now give an interpretation (semantics) for assertions. For each closed type expression σ we define an *interpretation function*

$$[\![\ \ ]\!]_\sigma : \mathcal{B}_\sigma \to \mathbf{KSN}(\,\mathcal{D}(\,\sigma\,)\,)$$

in the following structured way.

For each closed type expression σ, let

$$[\![\,\mathbf{t}\,]\!]_\sigma = \mathcal{D}(\,\sigma\,)$$

$$[\![\,\mathbf{f}\,]\!]_\sigma = \emptyset$$

$$[\![\top]\!]_O = \top$$

$$[\![\,\varphi \vee \psi\,]\!]_\sigma = [\![\,\varphi\,]\!]_\sigma \cup [\![\,\psi\,]\!]_\sigma$$

$$[\![\,\varphi \wedge \psi\,]\!]_\sigma = [\![\,\varphi\,]\!]_\sigma \cap [\![\,\psi\,]\!]_\sigma$$

With respect to type constructions we define

$$[\![\dot{inl}\,\varphi]\!]_{\sigma+\tau} = \{ (0, u) \mid u \in [\![\varphi]\!]_\sigma \setminus \{\emptyset\} \} \cup \{ x \in \mathcal{D}(\sigma+\tau) \mid \emptyset \in [\![\varphi]\!]_\sigma \}$$

$$[\![\dot{inr}\,\varphi]\!]_{\sigma+\tau} = \{ (1, u) \mid u \in [\![\varphi]\!]_\tau \setminus \{\emptyset\} \} \cup \{ x \in \mathcal{D}(\sigma+\tau) \mid \emptyset \in [\![\varphi]\!]_\tau \}$$

$$[\![\varphi \times \psi]\!]_{\sigma\times\tau} = \{ (0, u) \mid u \in [\![\varphi]\!]_\sigma \} \cup \{ (1, v) \mid v \in [\![\psi]\!]_\tau \}$$

$$[\![\varphi \otimes \psi]\!]_{\sigma\otimes\tau} = \{ x \subseteq \mathcal{D}(\sigma) \otimes \mathcal{D}(\tau) \mid \exists x_0 \in [\![\varphi]\!]_\sigma \exists x_1 \in [\![\psi]\!]_\tau . x_0 \times x_1 \subseteq x \}$$

$$[\![!\varphi]\!]_{!\sigma} = !([\![\varphi]\!]_\sigma)$$

$$[\![\varphi \multimap \psi]\!]_{\sigma\multimap\tau} = \{ x \in [\mathcal{D}(\sigma) \multimap \mathcal{D}(\tau)] \mid [\![\varphi]\!]_\sigma \sqsubseteq_\mu (Pt\, x)^{-1}([\![\psi]\!]_\tau) \}$$

$$[\![\varphi]\!]_{rec\,t.\,\sigma} = \{ \epsilon_\sigma(u) \mid u \in [\![\varphi]\!]_{\sigma[(rec\,t.\,\sigma)\backslash t]} \}$$

where $\epsilon_\sigma : [\, \mathcal{D}(\sigma[(rec\,t.\,\sigma)\backslash t]) \to \mathcal{D}(rec\,t.\,\sigma)\,]$ is the isomorphism arising form the initial solution to the domain equation associated with type $rec\,t.\,\sigma$.

See previous chapters for constructions $\otimes$, $\multimap$, and ! on stable neighbourhoods.

9.3 Soundness, Completeness, and Expressiveness

In this section we show that the proof system introduced for coherent spaces in the previous section is sound and complete. As for expressiveness we proof that any compact stable neighbourhood is expressible in the logic of coherent spaces.

Definition 9.3.1 For $\varphi, \psi \in \mathcal{B}_\sigma$, write $\models_\sigma \varphi \leq_\sigma \psi$ if $[\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma$.

Definition 9.3.2 For $\varphi, \psi \in \mathcal{B}_\sigma$, write $\vdash_\sigma \varphi \leq_\sigma \psi$ if $\varphi \leq_\sigma \psi$ can be derived from the proof system given in the previous section.

Definition 9.3.3 The proof system is called *sound* if $\vdash \varphi \leq_\sigma \psi$ implies $\models \varphi \leq_\sigma \psi$. It is *complete* if $\models \varphi \leq_\sigma \psi$ implies $\vdash \varphi \leq_\sigma \psi$. An axiom is *valid* if it is a valid formula. A rule is *sound* if it produces valid formulae from valid formulae.

Proposition 9.3.1 There is an isomorphism between atomic assertions of σ and tokens of $\mathcal{D}(\sigma)$ such that for any φ, ψ, atomic assertions of type σ, $\varphi \,\#\, \psi$ iff

$$[\![\varphi]\!]_\sigma \cap [\![\psi]\!]_\sigma \subseteq \emptyset.$$

Proof By an easy induction on the types. ∎

Proposition 9.3.2 1. Suppose $\varphi \in \mathcal{B}_\sigma$ and $\mathbf{P}(\varphi)$. Then there exists a finite element $x \in \mathcal{D}(\sigma)$ such that $[\![\varphi]\!]_\sigma = x\uparrow$. 2. Suppose φ is an atomic assertion of type σ. Then there exists $x \in \mathcal{D}(\sigma)$, a complete prime, such that $[\![\varphi]\!]_\sigma = x\uparrow$. 3. Moreover, if $\lceil\varphi\rceil = \{\varphi_i \mid i \in I\}$, then for the above x we have $x = \{[\![\varphi_i]\!] \mid i \in I\}$. 4. Suppose $\varphi \in \mathcal{B}_\sigma$ and $\varphi\downarrow$. Then $\mathcal{D}_\sigma \not\subseteq [\![\varphi]\!]_\sigma$.

Proof Routine. ∎

It is clear that a proof system is sound iff all its axioms are valid and rules sound. Theorem 9.3.1 establishes the soundness of the proof system.

Theorem 9.3.1

- The logical axioms are valid and logical rules sound.
- The axioms for sum, product, tensor product, shriek, and linear function space are valid.
- The rules for sum, product, tensor product, shriek, and linear function space are sound.

Proof That the logical axioms are valid and logical rules sound is obvious. The axioms and rules for sum and product are sound, similar to the case for **SFP** objects.

We check the tensor product, shriek, and linear function space in detail.

The soundness of $(\otimes - \mathbf{t})$ follows from the definition given in Theorem 8.4.1: for two stable neighbourhoods A and B, if $A \neq \emptyset$ and $\emptyset \in B$, then clearly $\emptyset \in A \otimes B$ as $x \times \emptyset = \emptyset$. The soundness of $(\otimes - \leq)$ and $(\otimes - \vee)$ is trivial while the soundness of $(\otimes - \wedge)$ follows from Proposition 8.4.1.

The soundness of $(! - \vee)$ for shriek follows from Proposition 8.4.4.

The soundness of $(\multimap - \vee)$, $(\multimap - \wedge)$ follows from Proposition 8.4.2 and Proposition 9.3.2. The soundness fo $(\multimap - \leq)$ follows from Proposition 8.4.3, Proposition 9.3.1 and Proposition 9.3.2, while the soundness of $(\multimap - \mathbf{f})$ follows from the definition given in Theorem 8.4.2 and Proposition 9.3.2. The rule $(\multimap)$ is sound since if f is a stable function and $x, y \in \mu f^{-1}(A)$ with $x \uparrow y$, then we must have $x = y$. ∎

Remark. In proofs of this section we will not check the case for recursively defined types, because all the time we are dealing with finite sets of assertions, and these assertions

can always be considered as of some finite type.

Definition 9.3.4 Write $\mathcal{P}_\sigma$ for the proof system associated with type σ. $\mathcal{P}_\sigma$ is called *prime complete* if it has property p_0, *prime normal* if it has property p_1, and *complete* if it has property p_2, where

$$(p_0) \qquad \forall \varphi, \psi : \sigma.\ (\mathbf{P}(\varphi)\ \&\ \mathbf{P}(\psi)\ \&\ [\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma \Longrightarrow \vdash \varphi \leq \psi)$$

$$(p_1) \qquad \varphi : \sigma \Longrightarrow \exists \{\, \varphi_i \mid i \in I \,\}.\ \forall i \in I.\, \mathbf{P}(\varphi_i)\ \&\ \vdash \varphi = \bigvee_{i \in I} \varphi_i$$

$$(p_2) \qquad \forall \varphi, \psi : \sigma.\ ([\![\varphi]\!] \subseteq [\![\psi]\!] \Longrightarrow \vdash \varphi \leq \psi)$$

In (p_1), $\bigvee_{i \in I} \varphi_i$ is called a *prime normal form* of φ.

Clearly $\mathcal{P}_\mathcal{O}$ has property p_0, p_1, and p_2. The proof for the completeness of the system is achieved by showing that each type construction preserves property (p_0), (p_1), and (p_2), by the following propositions. We omit the cases for sum and product, since they are not new. Note that for each case the proof of (p_2) is routine. It follows from (p_0) and (p_1) directly.

Proposition 9.3.3 Tensor product preserves p_0, p_1, and p_2.

Proof (p_0). Suppose $\varphi \otimes \psi$, $\varphi' \otimes \psi' : \sigma \times \tau$ are prime and $[\![\varphi \otimes \psi]\!] \subseteq [\![\varphi' \otimes \psi']\!]$. That $\varphi \otimes \psi$, $\varphi' \otimes \psi' : \sigma \times \tau$ are prime implies, by definition, $\varphi \otimes \psi \equiv \bigwedge_{i \in I} \varphi_i \otimes \psi_i$ and $\varphi' \otimes \psi' \equiv \bigwedge_{j \in J} \varphi'_j \otimes \psi'_j$, where φ_i, ψ_i, φ'_j, and ψ'_j are atomic. By Proposition 9.3.1, for any $j \in J$, there is some $i \in I$, $[\![\varphi_i \otimes \psi_i]\!] = [\![\varphi'_j \otimes \psi'_j]\!]$. This implies $[\![\varphi_i]\!] = [\![\varphi'_j]\!]$ and $[\![\psi_i]\!] = [\![\psi'_j]\!]$. By assumption, $\vdash \varphi_i = \varphi'_j$ and $\vdash \psi_i = \psi'_j$. Hence $\vdash \varphi_i \otimes \psi_i = \varphi'_j \otimes \psi'_j$, by $(\otimes\!-\leq)$. Using logical rules we get $\vdash \varphi \otimes \psi \leq \varphi' \otimes \psi'$.

(p_1) is routine. ∎

Proposition 9.3.4 Shriek preserves p_0, p_1, and p_2.

Proof (p_0). Suppose

$$[\![\bigwedge_{i \in I} !\varphi_i]\!] \subseteq [\![\bigwedge_{j \in J} !\psi_j]\!],$$

where $!\varphi_i$'s and $!\psi_j$'s are atomic. For any $j \in J$, there is some $i \in I$, $[\![\varphi_i]\!] = [\![\psi_j]\!]$. By assumption, $\vdash \varphi_i = \psi_j$. Hence $\vdash !\varphi_i = !\psi_j$, by $(!\!-\leq)$. Using logical rules we get $\vdash \bigwedge_{i \in I} !\varphi_i \leq \bigwedge_{j \in J} !\psi_j$.

(p_1) is easy. ∎

Proposition 9.3.5 Linear function space preserves p_0, p_1, and p_2.

Proof The proof for property p_0 is similar to previous cases.

p_1 is interesting in this case. Clearly using $(-\circ\ -\vee)$, $(-\circ\ -\mathbf{f})$ and the logical rules one can reduce each assertion of linear function space to a disjunctive form where each disjunct is a conjunction of assertions of the form $\varphi -\circ \psi$, with φ and ψ prime. It is enough to show that every such $\varphi -\circ \psi$ is reducible to a prime normal form. By $(-\circ - \leq)$ and logical rules we have

$$\begin{aligned}
\vdash \varphi -\circ \psi &\leq \bigwedge_{a\in\lceil\psi\rceil} \bigvee_{b\in\lceil\varphi\rceil} (b -\circ a) \\
&\leq \bigvee_{\kappa:\lceil\psi\rceil\to\lceil\varphi\rceil} \bigwedge_{a\in\lceil\psi\rceil} (\kappa(a) -\circ a) \\
&\leq^{*} \bigvee_{\kappa \text{ onto}} \bigwedge_{a\in\lceil\psi\rceil} (\kappa(a) -\circ a) \\
&\leq^{**} \varphi -\circ \psi
\end{aligned}$$

$*$: This is because for those κ's which are not onto, we can assume, say,

$$b_0 \in \lceil\varphi\rceil \setminus \kappa(\lceil\psi\rceil).$$

Therefore

$$\vdash (\varphi -\circ \psi) \wedge \bigwedge_{a\in\lceil\psi\rceil} (\kappa(a) -\circ a) \leq (\varphi -\circ \psi) \wedge (\varphi' -\circ \psi) \leq \mathbf{f}$$

by $(-\circ\ -\wedge)$ and $(-\circ)$, where $\varphi' \equiv \bigwedge_{a\in\lceil\psi\rceil} \kappa(a)$. Now use the fact that if $\varphi \wedge \psi_1 \leq \mathbf{f}$ and $\varphi \leq \psi_0 \vee \psi_1$, then

$$\varphi \leq \varphi \wedge (\psi_0 \vee \psi_1) \leq \varphi \wedge \psi_0.$$

Hence $\varphi \leq \psi_0$.

$**$: For those κ's which are onto, we clearly have, by $(-\circ\ -\wedge)$,

$$\vdash \bigwedge_{a\in\lceil\psi\rceil} (\kappa(a) -\circ a) \leq \varphi -\circ \psi.$$

Hence

$$\vdash \varphi -\circ \psi = \bigvee_{\kappa:\lceil\psi\rceil\to\lceil\varphi\rceil \text{ onto}} \bigwedge_{a\in\lceil\psi\rceil} (\kappa(a) -\circ a)$$

∎

In summary we have proved

Theorem 9.3.2 The proof system for the logic of coherent spaces is complete.

The expressive result is what one can expect:

Theorem 9.3.3 Let σ be a closed type expression. Then

$$[\![\ ]\!]_\sigma : (\mathcal{B}_\sigma/=,\ \leq_\sigma) \to (\mathbf{KSN}(\mathcal{D}(\sigma)),\ \subseteq)$$

is an isomorphism, where $\mathbf{KSN}(D)$ is the set of compact stable neighbourhoods of D.

Proof Any compact stable neighbourhood of a coherent space D is a finite, pairwise disjoint, union of *prime open* sets of the form $\{\, y \sqsupseteq x \mid y \in D \,\}$, with $x \in D$ a finite element. By the completeness theorem, if $[\![\varphi]\!]_\sigma \cap [\![\varphi']\!]_\sigma = \emptyset$ then $\vdash \varphi \wedge \varphi' = \mathbf{f}$, and $\varphi \vee \varphi' \in \mathcal{B}_\sigma$. Therefore, it is enough to show that for any prime open set $\{\, y \sqsupseteq x \mid y \in D \,\}$, there is some $\varphi \in \mathcal{B}_\sigma$ such that

$$[\![\varphi]\!]_\sigma = \{\, y \sqsupseteq x \mid y \in D \,\}.$$

However, finite elements of D are built up from the tokens. Hence it remains to show that for each token a of D there is an atomic assertion ψ, such that

$$[\![\psi]\!]_\sigma = \{\, y \sqsupseteq \{\, a \,\} \mid y \in D \,\}.$$

But this is just the conclusion of Proposition 9.3.1. ∎

9.4 Logic of DI

In this section we introduce a logic of **DI**, the category of dI-domains with stable functions. This framework is a generalisation of the logic of $\mathbf{COH}_s$, coherent spaces with stable functions. For the same reason as mentioned at the beginning of the previous section, we use a disjunctive language which is formulated by using proof rules for the syntax of assertions in addition to syntactic rules.

As usual a meta-language of type expressions is given as follows:

$$\sigma ::= \mathbf{1} \mid \sigma + \tau \mid \sigma \times \tau \mid \sigma \to_s \tau \mid \sigma_\perp \mid t \mid rec\, t.\sigma$$

where t is a type variable and σ, τ ranges over type expressions. Note that we could have used the linear function space $\multimap$ and the shriek ! instead of $\to_s$ via the adjunction

$$\frac{!\sigma \multimap \tau}{\sigma \to_s \tau}.$$

But we made a choice not to this time.

Every closed type expression is interpreted as a dI-domain, with **1** as the one-point domain, $\times$ as the cartesian product, $+$ as the coalesced sum, $(\ )_\perp$ as lifting, and $\rightarrow_s$ as the stable function space. $rec\ t.\sigma$ the initial solution of the associated domain equation in the category of dI-domains. Write again $\mathcal{D}(\sigma)$ for the domain corresponding to σ.

For each type σ we introduce an assertion language $\mathcal{C}_\sigma$ according to the following rules.

Assertions

$$\mathbf{t}, \mathbf{f} : \sigma \qquad \frac{\mathbf{A}_\sigma(\varphi)}{\varphi : \sigma}$$

$$\frac{\varphi,\ \psi : \sigma}{\varphi \wedge \psi : \sigma} \qquad \frac{\varphi \wedge \psi \leq \mathbf{f} \qquad \varphi,\ \psi : \sigma}{\varphi \vee \psi : \sigma}$$

$$\frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \times \psi : \sigma \times \tau} \qquad \frac{\varphi : \sigma}{(\varphi)_\perp : \sigma_\perp}$$

$$\frac{\varphi : \sigma}{inl\varphi : \sigma + \tau} \qquad \frac{\psi : \tau}{inr\psi : \sigma + \tau}$$

$$\frac{\varphi : \sigma \qquad \psi : \tau}{\varphi \rightarrow \psi : \sigma \rightarrow_s \tau}$$

$$\frac{\varphi : \sigma[rec\ t.\sigma/t]}{\varphi :\ rec\ t.\sigma}$$

When an assertion is of some type, *i.e.* $\varphi : \sigma$, we say φ is *well-formed.* Note that one of the rules,

$$\frac{\varphi \wedge \psi \leq \mathbf{f} \qquad \varphi,\ \psi : \sigma}{\varphi \vee \psi : \sigma},$$

makes use of the proof system. Here the same idea is used as was for the logic of coherent spaces, where a mutual recursion between syntactic rules and proof rules is allowed. The logic starts with some basic assertions, which are immediately justifiable to be well-formed. From these, one can use the proof rules to derive facts about them. Some of the facts, like $\varphi \wedge \psi \leq \mathbf{f}$, will allow one to form more assertions. In this way one get a disjunctive language.

Atomic assertions are some of the assertions to start with.

Atomic Assertions

$$\mathbf{A}_{\mathbf{1}_\perp}(\mathbf{t}_\perp)$$

$$\frac{\mathbf{A}_\sigma(\varphi)}{\mathbf{A}_{\sigma+\tau}(\dot{\imath}\mathit{nl}\varphi)} \qquad \frac{\mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma+\tau}(\dot{\imath}\mathit{nr}\psi)}$$

$$\frac{\mathbf{A}_\sigma(\varphi)}{\mathbf{A}_{\sigma_\perp}(\,(\varphi)_\perp\,)}$$

$$\frac{\mathbf{A}_\sigma(\varphi)}{\mathbf{A}_{\sigma\times\tau}(\varphi\times\mathbf{t}_\tau)} \qquad \frac{\mathbf{A}_\tau(\psi)}{\mathbf{A}_{\sigma\times\tau}(\mathbf{t}_\sigma\times\psi)}$$

$\mathbf{A}_{\sigma\to_s\tau}(\bigwedge_{i\in I}\varphi_i\to\psi_i)$ if

- $\forall i\in I.\,\mathbf{P}_\sigma(\varphi_i)\ \&\ \mathbf{A}_\tau(\psi_i)$
- $\exists k\in I.\,(\{\,\varphi_i\mid i\in I\,\}/\!\sim)\subseteq\lceil\varphi_k\rceil\ \&\ (\{\,\psi_i\mid i\in I\,\}/\!\sim)\subseteq[\![\psi_k]\!]$
- $\forall i,j\in I.\,i\neq j\Rightarrow\neg(\varphi_i\to\psi_i\,\#\,\varphi_j\to\psi_j)$
- $\forall j\in I\forall\chi\in[\![\psi_j]\!]\exists i\in I.\,\varphi_i\in\lceil\varphi_j\rceil\ \&\ \chi\sim\psi_i$

Call φ *atomic* if $\mathbf{A}_\sigma(\varphi)$. It is intended that atomic assertions capture complete primes in the corresponding dI-domains. For this reason the specification for the atomic assertion of function space is a bit complicated; however this complication does not seem to be caused by the way the rules are given, but rather by the inherent complexity of the complete primes (see Section 8.2 of Chapter 8). When no confusion arises we often omit type subscripts.

In the rules for atomic assertions several notations are used: they are $\mathbf{P}_\sigma$, $\#$, and $[\![\ ]\!]$. There is also $\lceil\ \rceil$, which will be used later in some other rules. The predicate $\mathbf{P}$ captures those *prime assertions* which correspond to prime open sets. The relation $\#$ captures *inconsistency* among the atomic assertions, so that from $\varphi\,\#\,\psi$ we can derive $\varphi\wedge\psi\leq\mathbf{f}$ by the propositional rule (#). The *atomizer*, $[\![\ ]\!]$, gives a set $[\![\varphi]\!]$ of atomic assertions determined by a prime assertion φ. On the other hand, the *primer*, $\lceil\ \rceil$, gives a set $\lceil\varphi\rceil$ of prime assertions determined by a prime assertion φ.

Inconsistency Relation

$$\dot{inl}\varphi \# \dot{inr}\psi$$

$$\frac{\varphi \# \varphi'}{\dot{inl}\varphi \# \dot{inl}\varphi'}$$

$$\frac{\psi \# \psi'}{\dot{inr}\psi \# \dot{inr}\psi'}$$

$$\frac{\varphi \# \varphi'}{(\varphi)_\perp \# (\varphi')_\perp}$$

$$\frac{\varphi \# \varphi'}{\varphi \times \mathbf{t} \# \varphi' \times \mathbf{t}}$$

$$\frac{\psi \# \psi'}{\mathbf{t} \times \psi \# \mathbf{t} \times \psi'}$$

(In the rules above all the assertions are assumed to be atomic)

$$\frac{\mathbf{P}(\varphi \wedge \varphi') \quad \mathbf{A}(\psi) \quad \mathbf{A}(\psi') \quad \psi \# \psi'}{\varphi \to \psi \# \varphi' \to \psi'}$$

$$\frac{\mathbf{P}(\varphi \wedge \varphi') \quad \varphi \not\sim \varphi' \quad \mathbf{A}(\psi) \quad \mathbf{A}(\psi') \quad \psi \sim \psi'}{\varphi \to \psi \# \varphi' \to \psi'}$$

$$\frac{\mathbf{A}(\bigwedge_{i\in I} \varphi_i \to \psi_i) \quad \mathbf{A}(\bigwedge_{j\in J} \varphi'_j \to \psi'_j) \quad \exists i \in I \exists j \in J.\ \varphi_i \to \psi_i \# \varphi'_j \to \psi'_j}{\bigwedge_{i\in I} \varphi_i \to \psi_i \# \bigwedge_{j\in J} \varphi'_j \to \psi'_j}$$

Here the *similarity relation* $\sim$ is a relation on atomic assertions (then easily extended to prime assertions) which is very close to *syntactic equality*. The only difference between $\sim$ and syntactic equality arise from the convention that for atomic assertions of function space, we ignore the order of the conjuncts. More precisely, let

$$(\varphi_\perp)^* = (\varphi^*)_\perp,$$

$$(\varphi \times \psi)^* = (\varphi^*) \times (\psi)^*,$$

$$(\dot{inl}\varphi)^* = \dot{inl}(\varphi)^*,\ (\dot{inr}\psi)^* = \dot{inr}(\psi)^*,$$

$$(\varphi \to \psi)^* = (\varphi)^* \to (\psi)^*,$$

(note here $(\ )_\perp$, $\times$, $\dot{inl}$, $\dot{inr}$, and $\rightarrow$ should be treated as syntax) and

$$(\bigwedge_{i\in I}\varphi_i)^* = \bigcup_{i\in I}\{(\varphi_i)^*\}$$

for an atomic assertion $\bigwedge_{i\in I}\varphi_i$. Then $\sim$ can be formally defined by the following rules. All the assertions in the following table are assumed to be atomic.

Similarity Relation

$$\mathbf{t}_\perp \sim \mathbf{t}_\perp$$

$$\frac{\varphi \sim \varphi'}{\dot{inl}\varphi \sim \dot{inl}\varphi'}$$

$$\frac{\psi \sim \psi'}{\dot{inr}\psi \sim \dot{inr}\psi'}$$

$$\frac{\varphi \sim \varphi'}{(\varphi)_\perp \sim (\varphi')_\perp}$$

$$\frac{\varphi \sim \varphi'}{\varphi \times \mathbf{t} \sim \varphi' \times \mathbf{t}}$$

$$\frac{\psi \sim \psi'}{\mathbf{t} \times \psi \sim \mathbf{t} \times \psi'}$$

$$\frac{\{(\varphi_i \rightarrow \psi_i)^* \mid i \in I\} = \{(\varphi'_j \rightarrow \psi'_j)^* \mid j \in J\}}{\bigwedge_{i\in I}\varphi_i \rightarrow \psi_i \ \sim\ \bigwedge_{j\in J}\varphi'_j \rightarrow \psi'_j}$$

The similarity relation $\sim$ can then be easily extended to prime assertions by defining

$$\bigwedge_{i\in I}\varphi_i \sim \bigwedge_{j\in J}\psi_j \Longleftrightarrow (\bigwedge_{i\in I}\varphi_i)^* = (\bigwedge_{j\in J}\psi_j)^*.$$

The similarity relation is introduced as a simpler method to decide wheather two atomic assertions have the same interpretations as stable neighbourhoods. It will be shown later (Proposition 9.5.2) that two prime (specially atomic) assertions are similar iff they have the same interpretation as stable neighbourhoods. The following are the definition for the other notations.

$$\mathbf{P}_\sigma(\varphi) \Longleftrightarrow \varphi \equiv \bigwedge_{i\in I}\varphi_i \ \&\ \forall i \in I.\ \mathbf{A}_\sigma(\varphi_i) \ \&\ \forall i \neq j.\ \neg(\varphi_i \# \varphi_j)$$

$$(\varphi)\downarrow \Longleftrightarrow \mathbf{P}(\varphi) \ \&\ \varphi \equiv \bigwedge_{i\in I}\varphi_i \ \&\ I \neq \emptyset$$

Let φ be a prime assertion, i.e. $\mathbf{P}(\varphi)$. Then there are assertions φ_i, $i \in I$, such that $\varphi \equiv \bigwedge_{i\in I} \varphi_i$ and $\forall i \in I.\, \mathbf{A}(\varphi_i)$. We define

$$\lceil \varphi \rceil = \{ \bigwedge_{j\in J} \varphi_j \mid \mathbf{P}(\bigwedge_{j\in J} \varphi_j) \;\&\; J \subseteq I \}/\sim$$

and

$$\lfloor\!\lceil \varphi \rceil\!\rfloor = \{ \bigwedge_{j\in J} \varphi_j \mid \mathbf{A}(\bigwedge_{j\in J} \varphi_j) \;\&\; J \subseteq I \}/\sim .$$

Note that when forming the sets $\lceil \varphi \rceil$ and $\lfloor\!\lceil \varphi \rceil\!\rfloor$ we took the quotient over $\sim$. Thus it is reminded that equalty of sets $\lfloor\!\lceil \varphi \rceil\!\rfloor = \lfloor\!\lceil \varphi' \rceil\!\rfloor$ is equality on quotient sets. Note also that if we use ! and $\multimap$ instead of $\rightarrow_s$ for the type expressions, the simple relation

$$\lfloor\!\lceil !\varphi \rceil\!\rfloor_{!\sigma} = \{ !\psi \mid \psi \in \lceil \varphi \rceil_\sigma \}$$

will hold.

It is easy to see, from the definition of $\mathbf{P}$, that

$\mathbf{P}_{\sigma\rightarrow_s\tau}(\bigwedge_{i\in I} \varphi_i \rightarrow \psi_i)$ iff

- $\forall i \in I.\, \mathbf{P}_\sigma(\varphi_i) \;\&\; \mathbf{A}_\tau(\psi_i)$
- $\forall i, j \in I.\, i \neq j \Rightarrow \neg(\varphi_i \rightarrow \psi_i \,\#\, \varphi_j \rightarrow \psi_j)$
- $\forall j \in I \forall \chi \in \lfloor\!\lceil \psi_j \rceil\!\rfloor \exists i \in I.\, \varphi_i \in \lceil \varphi_j \rceil \;\&\; \chi \sim \psi_i$

where compared with the definition for $\mathbf{A}$, we do not require that

$$\exists k \in I.\, (\{ \varphi_i \mid i \in I \}/\sim) \subseteq \lceil \varphi_k \rceil \;\&\; (\{ \psi_i \mid i \in I \}/\sim) \subseteq \lfloor\!\lceil \psi_k \rceil\!\rfloor.$$

Propositional Rules

(t)	$\varphi \leq \mathbf{t}$	(f)	$\mathbf{f} \leq \varphi$
(Ref)	$\varphi \leq \varphi$	(#)	$\dfrac{\varphi \# \psi}{\varphi \wedge \psi \leq \mathbf{f}}$
(Trans)	$\dfrac{\varphi \leq \varphi' \quad \varphi' \leq \varphi''}{\varphi \leq \varphi''}$		
$(\leq - =)$	$\dfrac{\varphi \leq \psi \quad \psi \leq \varphi}{\varphi = \psi}$		
$(= - \leq)$	$\dfrac{\varphi = \varphi'}{\varphi \leq \varphi'} \quad \dfrac{\varphi = \varphi'}{\varphi' \leq \varphi}$		
$(\wedge - \leq)$	$\varphi \wedge \varphi' \leq \varphi \quad \varphi \wedge \varphi' \leq \varphi'$		
$(\leq - \wedge)$	$\dfrac{\varphi \leq \varphi' \quad \varphi \leq \varphi''}{\varphi \leq \varphi' \wedge \varphi''}$		
$(\vee - \leq)$	$\dfrac{\varphi \leq \varphi' \quad \psi \leq \varphi'}{\varphi \vee \psi \leq \varphi'}$		
$(\leq - \vee)$	$\varphi \leq \varphi' \vee \varphi \quad \varphi' \leq \varphi' \vee \varphi$		
$(\wedge - \vee)$	$\varphi \wedge (\varphi_1 \vee \varphi_2) \leq (\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2)$		

Assertions in the above table are assumed to be all well-formed. The proof system consists of several groups of axioms and rules given below. There are type-specific rules which provide relationships between axioms of different types. There are also axioms that tell us how logical constructions interact with type constructions.

Sum

$$(\dot{inl}-\leq)\quad \frac{\varphi \leq_\sigma \psi}{\dot{inl}\,\varphi \leq_{\sigma+\tau} \dot{inl}\,\psi}$$

$$(\dot{inr}-\leq)\quad \frac{\varphi \leq_\tau \psi}{\dot{inr}\,\varphi \leq_{\sigma+\tau} \dot{inr}\,\psi}$$

$$(\dot{inl}-\wedge)\quad \dot{inl}\,(\bigwedge_{i\in I}\varphi_i) =_{\sigma+\tau} \bigwedge_{i\in I}\dot{inl}\,\varphi_i$$

$$(\dot{inr}-\wedge)\quad \dot{inr}\,(\bigwedge_{i\in I}\psi_i) =_{\sigma+\tau} \bigwedge_{i\in I}\dot{inr}\,\psi_i$$

$$(\dot{inl}-\vee)\quad \dot{inl}\,(\bigvee_{i\in I}\varphi_i) =_{\sigma+\tau} \bigvee_{i\in I}\dot{inl}\,\varphi_i$$

$$(\dot{inr}-\vee)\quad \dot{inr}\,(\bigvee_{i\in I}\psi_i) =_{\sigma+\tau} \bigvee_{i\in I}\dot{inr}\,\psi_i$$

Product

$$(\times-\leq)\quad \frac{\psi \leq_\sigma \psi' \quad \varphi \leq_\tau \varphi'}{\psi\times\varphi \leq_{\sigma\times\tau} \psi'\times\varphi'}$$

$$(\times-\vee)\quad \varphi\times(\bigvee_{i\in I}\psi_i) =_{\sigma\times\tau} \bigvee_{i\in I}(\varphi\times\psi_i)$$

$$(\bigvee_{i\in I}\varphi_i)\times\psi =_{\sigma\times\tau} \bigvee_{i\in I}(\varphi_i\times\psi)$$

$$(\times-\wedge)\quad \bigwedge_{i\in I}(\varphi_i\times\psi_i) =_{\sigma\times\tau} (\bigwedge_{i\in I}\varphi_i)\times(\bigwedge_{i\in I}\psi_i)$$

Lifting

$$(\bot-\leq)\quad \frac{\varphi \leq \psi}{\varphi_\bot \leq \psi_\bot}$$

$$(\bot-\wedge)\quad (\varphi_1\wedge\varphi_2)_\bot = (\varphi_1)_\bot \wedge (\varphi_2)_\bot$$

$$(\bot-\vee)\quad (\bigvee_{i\in I}\varphi_i)_\bot = \bigvee_{i\in I}(\varphi_i)_\bot$$

Function Space

$$(\to -\mathbf{f}) \qquad \frac{\varphi\downarrow}{(\varphi \to \mathrm{t}) \leq \mathbf{f}}$$

$$(\to) \qquad \frac{[\![\varphi]\!] \neq [\![\varphi']\!] \qquad \mathbf{P}(\varphi \wedge \varphi') \qquad \mathbf{P}(\psi)}{(\varphi \to \psi) \wedge (\varphi' \to \psi) \leq \mathbf{f}}$$

$$(\to - \leq) \qquad \frac{\chi \in [\![\psi]\!] \qquad \mathbf{P}(\varphi)}{\varphi \to \psi \leq \bigvee_{\xi \in \lceil\varphi\rceil}(\xi \to \chi)}$$

$$(\to - \wedge) \qquad \bigwedge_{i\in I}(\varphi_i \to \psi_i) \leq (\bigwedge_{i\in I}\varphi_i) \to (\bigwedge_{i\in I}\psi_i)$$

$$(\to - \vee) \qquad \frac{\mathbf{P}(\varphi)}{\varphi \to (\bigvee_{i\in I}\psi_i) = \bigvee_{i\in I}(\varphi \to \psi_i)}$$

$$(\bigvee_{i\in I}\varphi_i) \to \psi = \bigwedge_{i\in I}(\varphi_i \to \psi)$$

Now we give an interpretation (semantics) for assertions. For each closed type expression σ we define an *interpretation function*

$$[\![\ \ ]\!]_\sigma : \mathcal{C}_\sigma \to \mathbf{KSN}(\mathcal{D}(\sigma))$$

with $\mathbf{KSN}(D)$ the collection of compact stable neighbourhoods of D. $[\![\ \ ]\!]_\sigma$ is defined in the following structured way.

For each closed type expression σ, we define

$$[\![\mathrm{t}]\!]_\sigma = \mathcal{D}(\sigma)$$

$$[\![\mathbf{f}]\!]_\sigma = \emptyset$$

$$[\![\varphi \vee \psi]\!]_\sigma = [\![\varphi]\!]_\sigma \cup [\![\psi]\!]_\sigma$$

$$[\![\varphi \wedge \psi]\!]_\sigma = [\![\varphi]\!]_\sigma \cap [\![\psi]\!]_\sigma$$

With respect to type constructions we define

$$[\![\varphi \times \psi]\!]_{\sigma\times\tau} = \{\,(u, v) \mid u \in [\![\varphi]\!]_\sigma \;\&\; v \in [\![\psi]\!]_\tau\,\}$$

$$[\![\dot{inl}\,\varphi]\!]_{\sigma+\tau} = \{\,(0, u) \mid u \in [\![\varphi]\!]_\sigma \setminus \{\,\perp_{\mathcal{D}(\sigma)}\,\}\,\} \cup \{\,x \in \mathcal{D}(\sigma+\tau) \mid \perp_{\mathcal{D}(\sigma)} \in [\![\varphi]\!]_\sigma\,\}$$

$$[\![\dot{inr}\,\varphi]\!]_{\sigma+\tau} = \{\,(1, u) \mid u \in [\![\varphi]\!]_\tau \setminus \{\,\perp_{\mathcal{D}(\tau)}\,\}\,\} \cup \{\,x \in \mathcal{D}(\sigma+\tau) \mid \perp_{\mathcal{D}(\tau)} \in [\![\varphi]\!]_\tau\,\}$$

$$[\![\varphi \to \psi]\!]_{\sigma\to_s\tau} = \{\,f \in \mathcal{D}(\sigma) \to_s \mathcal{D}(\tau) \mid [\![\varphi]\!]_\sigma \sqsubseteq_\mu f^{-1}([\![\psi]\!]_\tau)\,\}$$

$$[\![(\varphi)_\perp]\!]_{(\sigma)_\perp} = \{\,(0, u) \mid u \in [\![\varphi]\!]_\sigma\,\}$$

$$[\![\varphi]\!]_{rec.\sigma} = \{\,\epsilon_\sigma(u) \mid u \in [\![\varphi]\!]_{\sigma[(rec.\sigma)\backslash t]}\,\}$$

where $\epsilon_\sigma : \mathcal{D}(\sigma[(rec.\sigma)\backslash t]) \to \mathcal{D}(rec.\sigma)$ is the equality arising form the initial solution to the domain equation associated with type $rec\,t.\,\sigma$ in the category of dI-domains.

9.5 Completeness

In this section we show that the proof system introduced for dI-domains is sound and complete. The definitions, propositions, and proofs have a similar style to those in Section 9.3.

Definition 9.5.1 For $\varphi, \psi \in \mathcal{C}_\sigma$, write $\models_\sigma \varphi \leq_\sigma \psi$ if $[\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma$.

Definition 9.5.2 For $\varphi, \psi \in \mathcal{C}_\sigma$, write $\vdash_\sigma \varphi \leq_\sigma \psi$ if $\varphi \leq_\sigma \psi$ can be derived from the proof system given in the previous section.

Definition 9.5.3 The proof system is called *sound* if $\vdash \varphi \leq_\sigma \psi$ implies $\models \varphi \leq_\sigma \psi$. It is *complete* if $\models \varphi \leq_\sigma \psi$ implies $\vdash \varphi \leq_\sigma \psi$. An axiom is *valid* if it is a valid formula. A rule is *sound* if it produces valid formulae from valid formulae.

Proposition 9.5.1 There is an isomorphism between atomic assertions of σ and tokens of $\mathcal{D}(\sigma)$ such that for any φ, ψ, atomic assertions of type σ, $\varphi \,\#\, \psi$ iff

$$[\![\varphi]\!]_\sigma \cap [\![\psi]\!]_\sigma \subseteq \emptyset.$$

Proof By an easy induction on the types. ∎

Proposition 9.5.2 Let φ, ψ be prime assertions of type σ. Then $\varphi \sim \psi$ iff $[\![\varphi]\!] = [\![\psi]\!]$.

Proof

($\Rightarrow$) By inspecting the definitions..

($\Leftarrow$) Note that for prime assertions φ and ψ, $(\varphi)^* \neq (\psi)^*$ implies $[\![\varphi]\!] \neq [\![\psi]\!]$ provided that we have $\varphi \not\sim \psi$ implies $[\![\varphi]\!] \neq [\![\psi]\!]$ for atomic assertions φ and ψ. Therefor it is enough to check the conclusion for atomic assertions. We show that $\varphi \not\sim \psi$ implies $[\![\varphi]\!] \neq [\![\psi]\!]$. This is done by structured induction on types. For base type $\mathbf{1}$, $\varphi : \mathbf{1}$ is prime iff $\varphi \equiv \bigwedge \emptyset$ by definition, since type $\mathbf{1}$ has no atomic assertion. Therefore the statement 'for prime assertions φ, ψ, $\varphi \not\sim \psi$ implies $[\![\varphi]\!] \neq [\![\psi]\!]$' is vacuously true since there is not prime assertions φ, ψ of $\mathbf{1}$ for which $\varphi \not\sim \psi$. Next we show that the rules for Similarity Relation preserves the property $\varphi \not\sim \psi$ implies $[\![\varphi]\!] \neq [\![\psi]\!]$, but that is trivial. ∎

Proposition 9.5.3 1. Suppose φ is an atomic assertion of type σ. Then there exists $x \in \mathcal{D}(\sigma)$, a complete prime, such that $[\![\varphi]\!]_\sigma = x\uparrow$. 2. Suppose $\varphi \in \mathcal{C}_\sigma$ and $\mathbf{P}(\varphi)$. Then there exists a finite element $x \in \mathcal{D}(\sigma)$ such that $[\![\varphi]\!]_\sigma = x\uparrow$. 3. Let φ be a prime assertion of σ and write $\check{\varphi}$ for the finite element of $\mathcal{D}(\sigma)$ such that $[\![\varphi]\!]_\sigma = \check{\varphi}\uparrow$. If $\lceil\varphi\rceil = \{\varphi_i \mid i \in I\}$, then $\{\check{\varphi}_i \mid i \in I\} = \{y \in (\mathcal{D}(\sigma))^0 \mid y \sqsubseteq \check{\varphi}\}$; If $\lceil\!\lceil\varphi\rceil\!\rceil = \{\varphi_i \mid i \in I\}$, then $\{\check{\varphi}_i \mid i \in I\} = \{y \in (\mathcal{D}(\sigma))^1 \mid y \sqsubseteq \check{\varphi}\}$. 4. Suppose $\varphi \in \mathcal{C}_\sigma$ and $\varphi\downarrow$. Then $\mathcal{D}_\sigma \not\subseteq [\![\varphi]\!]_\sigma$.

Recall that D^0 stands for the set of finite elements of D and D^1 stands for the set of complete primes of D.

Proof The proofs for 3 and 4 are routine. We use an induction on types to show 1 and 2. The nontrivial case is function space. However, that follows from Definition 8.2.2, Proposition 8.2.1, and Proposition 8.2.6. ∎

Proposition 9.5.4 If $\lceil\!\lceil\varphi\rceil\!\rceil \neq \lceil\!\lceil\psi\rceil\!\rceil$ for prime assertions φ, ψ of σ, then $[\![\varphi]\!]_\sigma \neq [\![\psi]\!]_\sigma$.

Proof It is easy to show that $\lceil\!\lceil\varphi\rceil\!\rceil \neq \lceil\!\lceil\psi\rceil\!\rceil$ implies $\varphi \not\sim \psi$. The conclusion then follows from Proposition 9.5.3. ∎

Theorem 9.5.1 establishes the soundness of the proof system.

Theorem 9.5.1

- The logical axioms are valid and logical rules sound.
- The axioms for sum, product, lifting, and function space are valid.
- The rules for sum, product, lifting, and function space are sound.

Proof We check this fact for the function space construction. Other cases are much easier.

The soundness of $(\to -\vee)$, $(\to -\wedge)$ follows from Proposition 8.3.2 and Proposition 9.5.3. The soundness fo $(\to - \leq)$ follows from Proposition 8.3.3, Proposition 9.5.1 and Proposition 9.5.3, while the soundness of $(\to -\mathbf{f})$ follows from Definition 8.3.1 and the fourth conclusion of Proposition 9.5.3. The rule $(\to)$ is sound since if f is a stable function and $x, y \in \mu f^{-1}(A)$ with $x \uparrow y$, then we must have $x = y$. ∎

Definition 9.5.4 Write $\mathcal{Q}_\sigma$ for the proof system associated with type σ. $\mathcal{Q}_\sigma$ is called *prime complete* if it has property p_0, *prime normal* if it has property p_1, and *complete* if it has property p_2, where

$$(p_0) \qquad \forall \varphi, \psi : \sigma.\ (\mathbf{P}(\varphi)\ \&\ \mathbf{P}(\psi)\ \&\ [\![\varphi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma \Longrightarrow \vdash \varphi \leq \psi)$$

$$(p_1) \qquad \varphi : \sigma \Longrightarrow \exists \{\, \varphi_i \mid i \in I \,\}.\ \forall i \in I.\, \mathbf{P}(\varphi_i)\ \&\ \vdash \varphi = \bigvee_{i \in I} \varphi_i$$

$$(p_2) \qquad \forall \varphi, \psi : \sigma.\ ([\![\varphi]\!] \subseteq [\![\psi]\!] \Longrightarrow \vdash \varphi \leq \psi)$$

In (p_1), $\bigvee_{i \in I} \varphi_i$ is called a *prime normal form* of φ.

Clearly $\mathcal{Q}_1$ has property p_0, p_1, and p_2. The proof for the completeness of the system is achieved by showing that each type construction preserves property (p_0), (p_1), and (p_2). We omit the cases for lifting, sum, and product, since they are straightforward. Note that for each case the proof of (p_2) is routine. It follows from (p_0) and (p_1) directly.

Proposition 9.5.5 Function space preserves p_0, p_1, and p_2.

Proof The proof of property p_0 is easy. We prove the interesting property p_1 for function space. Similarly to the proof of Proposition 9.3.5 we have

$$\vdash \varphi \to \psi = \bigvee_{\kappa : [\![\psi]\!] \to \lceil \varphi \rceil \text{ onto}} \ \bigwedge_{a \in [\![\psi]\!]} (\kappa(a) \to a).$$

Note

$$\bigwedge_{a \in [\![\psi]\!]} (\kappa(a) \to a)$$

need not be a prime assertion, since the property

$$\forall j \in I\, \forall \xi \in [\![\psi_j]\!]\, \exists i \in I.\ \varphi_i \in \lceil \varphi_j \rceil\ \&\ \xi \sim \psi_i$$

required for a prime assertion $\bigwedge_{i \in I} \varphi_i \to \psi_i$ may not hold. To get a prime normal form

we have to apply the same procedure again to $\kappa(a) \to a$ for each conjuncts of

$$\bigwedge_{a \in \lceil\psi\rceil} (\kappa(a) \to a).$$

Repeat this procedure for a finite number of times we get a prime normal form for $\varphi \to \psi$. ∎

From the above we get

Theorem 9.5.2 The proof system for the logic of coherent spaces is complete.

The expressive result is what one can expect:

Theorem 9.5.3 Let σ be a closed type expression. Then

$$[\![\ ]\!]_\sigma : (\mathcal{C}_\sigma/=,\ \leq_\sigma) \to (\mathbf{KSN}(\mathcal{D}(\sigma)),\ \subseteq)$$

is an isomorphism, where $\mathbf{KSN}(D)$ is the set of compact stable neighbourhoods of D.

Proof Similar to that of Theorem 9.3.3 ∎

Thus we obtain results for dI-domains analogous to those of Abramsky for Scott domains. Admittedly the term language is not considered here.

Chapter 10

Conclusion

10.1 What Has Been Achieved

We have studied domain logics of two important frameworks for denotational semantics, the **SFP** objects and the dI-domains with stable functions. Domain logics are shown as the appropriate bridges with which to connect denotational semantics and program logics. They are demonstrated to be an important link in the systematic derivation of proof systems for programming languages from their semantics.

10.2 Further Work

While this thesis contributes to various aspects of domain logics, it has left some open questions as well as rooms for further development. Chief among these is the omission of a treatment of morphisms and the consequent lack of general proof rules reasoning that a program satisfies an assertion (In [Ab87] a general notion of weakest precondition is used for this purpose). The following are additional suggestions for future research.

1. In Chapter 3 we presented an improved version of Brookes' proof system without using labels. In Chapter 5 the style of the assertion language of the improved proof system is shown to be derivable from Plotkin's domain of resumptions. There are two projects related to Brookes' proof system. One is to prove the completeness (or incompleteness) of the improved version of Brookes' proof system. The other is to show that not only the assertions, but also the proof rules of Brookes' proof system are semantics derived.

2. In Chapter 4 we introduced generalised information systems. Special kind of such systems called strongly finite information systems are made into a category, which is equivalent to the category **SFP**. It would be interesting to know what kind of domains generalised information systems represent; whether these domains form a reasonable framework for denotational semantics; and how does it relate to the the existing frameworks.

3. Chapter 6 provided a basic framework for the domain fixed-point calculus. There are several interesting aspects that need further study. It would be satisfying to get the

completeness result in general. The proof for the general completeness is expected to follow the same style as that for the integer case, through normal form theorems.

Some syntactic restriction is imposed to get the completeness of the integer mu-calculus. It is a project to study whether the restriction is necessary, and whether in general it is possible to get completeness without such syntactic restrictions. Normal form theorems should automatically provide decidability results. However, it is not clear whether there are normal forms for mu-assertions in general. It is also not clear to the author whether or not the domain mu-calculus is decidable.

The mu-calculus of Chapter 6 should be easily extendible to a mu-calculus of **SFP** objects, with modal operators $\Box$ and $\Diamond$. It would be interesting to see what is its connections with the mu-calculus of Hennessy-Milner logic.

To get more expressive power further extending of the mu-calculus is necessary. It is tempting to add negation, and nu-operator, to express maximum fixed-points. However, that necessarily leads us outside the Scott open sets, and it is not clear what kind of sets should be used (Mike Smyth has proposed the G_δ sets in [Sm83]).

4. In Chapter 7 the relationship between some of the categories of dI-domains is not established. It would be pleasing to complete the diagram of the relationships between various categories of dI-domains given at the end of Chapter 7.

5. An immediate question about the logic of coherent spaces is whether it has anything to do with linear logic since coherent spaces are used as a semantics for linear logic. Notice, however, there is a mismatch here. In linear logic each proposition is interpreted as a coherent space while in the logic of coherent spaces each assertion is interpreted as a stable neighbourhood of certain coherent space. Resolving this mismatch would be a first step toward the understanding of the relationship, if there is any, between linear logic and the logic I have presented for coherent spaces.

As mentioned, it would be interesting to introduce a language of morphism terms for coherent spaces and dI-domains, and to establish the corresponding logical frameworks so that it is possible to express special kind of Hoare triples and dynamic logic. It is sensible to formulate mu-calculi of coherent spaces and dI-domains.

Although some potential difficulties in formulating logic of partially synchronous mor-

phisms was pointed out, one can still try to get a logic for $\mathbf{SEV}_{syn}$ and $\mathbf{SEV}^{*}_{syn}$, possibily with limited forms of quantification over assertions as proposed at the end of Chapter 8. Such a logic might be helpful in understanding logics for event structures and CCS-like languages.

Bibliography

[Ab85] Abramsky, S. Domain theory in logical form, manuscript (1985)

[Ab87] Abramsky, S. Domain theory in logical form, Proc. of Second Annual Symposium on Logic in Computer Science (1987)

[Ab87a] Abramsky, S. A domain equation for bisimulation, Chapter 5 of 'Domain theory and the logic of observable properties', PhD thesis, University of London, (1987)

[Ab88] Abramsky, S. Domain theory in logical form, Imperial College Research Report DOC 88/15 (1988)

[Ap81] Apt. Ten years of Hoare 's logic Part I, ACM TOPLS 3 (1981)

[Ap84] Apt. Ten years of Hoare's logic Part II: Nondeterminism , Throretical Computer Science 28 (1984)

[Ba80] de Bakker, J. W., *Mathematical theory of program correctness,* Prentice Hall International, (1980)

[Ba84] Barendregt, H., *The λ calculus: Its syntax and semantics,* North-Holland, (1984)

[Be78] Berry, G., Stable models of typed $\lambda-$calculi, Lecture Notes in Computer Science 62 (1978)

[Br85] Brookes, S.D., An axiomatic treatment of a parallel programming language, Lecture Notes in Computer Science 193 (1985)

[CoGuWi87] Coquand, T., Gunter, C., Winskel, G., DI-domians as a model of polymorphism, Technical Report 107, Computer Laboratory, Cambridge University (1987)

[Co78] Cook, S., Soundness and completeness of an axiom system for program verification, SIAM J. Computing 7 (1) (1978)

[Di76] Dijkstra, E.W., A discipline of programming, Prentice-Hall, (1976)

[Die76] Diers, Y, *Catégories Localisables,* thèse de doctorat d'éetat, Paris VI (1976)

[Gi87a] Girard, Jean-Yves, The system F of variable types, fifteen years later, Theoretical Computer Science 45 (1986)

[Gi87b] Girard, Jean-Yves, Linear Logic, Theoretical Computer Science 50 (1987)

[Go79] Gordon, M.J.C., *The denotational description of programming languages*, Springer Verlag (1979)

[Gu85] Gunter, C., *Profinite solutions for recursive domain equations*, PhD thesis, Department of Computer Science, Carnegie-Mellon University, (1985)

[HeMi79] Hennessy, M.C.B., Minler, R., On observing nondeterminism and concurrency, Lecture Notes in Computer Science 85 (1979)

[Ho69] Hoare, C.A.R., An aximatic basis for computer programming, CACM 12 10 (1969) 576-580

[Ho78] Hoare, C.A.R., Communication sequential processes, CACM 21 (8) (1978)

[Jo77] Johnstone, P.T., A syntactic approach to Diers' localizable categories, Lecture Notes in Mathematics 753 (1977)

[Jo82] Johnstone, P.T., *Stone spaces*, Cambridge University Press (1982)

[Ko83] Kozen, D., Results on the propositional $\mu-$calculus, Theoretical Computer Science 27 (1983)

[LaWi84] Larsen, K.G., Winskel, G., Using information systems to solve recursive domain equations effectively, Lecture Notes in Computer Science 173 (1984)

[La87] Larsen, K.G., Proof systems for Hennessy-Minler logic with recursion, Technical Report, Aalborg University Center, Institute for Electronic Systems, Department of Mathematics and Computer Science (1987)

[Ma71] MacLane, S., *Categories for the working mathematician*, Springer-Verlag (1971)

[Me88] Meyer, A. R., Semantical paradigms: Notes for an invited lecture, Third annual symposium on Logic in Computer Science, Edinburgh, Scotland, (1988)

[Mi77] Milner, R., Fully abstract models of typed lambda-calculi, Theoretical Computer Science 4, (1977)

[Mi80] Milner, R., *A calculus of communication systems,* Lecture Notes in Computer Science 92 (1980)

[NiPlWi79] Nielsen, M., Plotkin, G., Winskel, G., Petri nets, event structures and domains, part 1, Theoretical Computer Science 13 (1981)

[NiHe83] de Nicola, R. Hennessy, M.C.B., Testing equivalences for processes Lecture Notes in Computer Science 154 (1983)

[OwGr76] Owicki, S.S., Gries, D., An axiomatic proof technique for parallel programs, Acta Informatica 6 (1976) 319-340

[Pl76] Plotkin, G., A powerdomain construction, SIAM J. Computing 5(1976) 452-486

[Pl78] Plotkin, G.D., The category of complete partial orders: a tool for making meanings, (Pisa Notes) Proc. Summer School on Foundations of Artificial Intelligence and Computer Science, Instituto di Scienze dell' Informazione, Universita di Pisa (1978)

[Pl80] Plotkin, G., Dijkstra's predicate transformers and Smyth's powerdomains, Lecture Notes in Computer Science 86 (1980)

[Pl81] Plotkin, G.D., A structural approach to operational semantics DAIMI Report FN-19, Aarhus University (1981)

[Pn77] Pnueli, A., The temporal logic of programs, Proc. 19th Annual Symposium on the Foundations of Computer Science (1977)

[Pr79] Pratt, V.R., Dynamic Logic, Proc. 6th International Congress of Logic, Methodology and Philosophy of Science, Hannover (1979)

[Ro86] Robinson, E., Power-domains, Modalities and Vietoris Monad, Technical Report 98(1986), Computer Laboratory, University of Cambridge

[Ro87] Robinson, E., Logical aspects of denotational semantics, Lecture Notes in Computer Science 283 (1987)

[Sh87] Shawe-Taylor, J. S., *The semantics and open set analysis of a first order programming language*, MSc thesis, Imperial College of Science and Technology (1987)

[Sc82] Scott, D. S., Domains for denotational semantics, Lecture Notes in Computer Science 140 (1982)

[Sc82a] Scott, D.S., Notes on cpos/SFP-objects and the like, manuscript, (1982)

[ScSt71] Scott, D.S., Strachey, C., Towards a mathematical semantics of computer languages, Technical Report PRG-6, Oxford University Computing Laboratory, Programming Research Group (1971)

[St64] Strachey, C., Towards a formal semantics, Proc. IFIP Working Conference on Formal Language Description Languages (1964)

[SmPl82] Smyth, M.B., Plotkin, G.D., The category-theoretic solution of recursive domain equations, SIAM Journal of Computing, vol. 11 (1982)

[Sm83] Smyth, M.B., Power domains and predicate transfmers: a topological view. Lecture Notes in Computer Science 154 (1983)

[Sm78] Smyth, M.B., Powerdomains, JCSS 16 (1) (1978)

[Ta55] Tarski, A., A lattice-theoretical fixed point theorem and its applications, Pacific Journal of Mathematics, vol. 5 (1955)

[Vi88] Vicker, S. *Logic via topology,* Cambridge University Press, (1989)

[Wi80] Winskel, G., *Events in computation,* PhD thesis, University of Edinburgh (1980)

[Wi82] Winskel, G., Event structures semantics of CCS and related languages, Lecture Notes in Computer Science 140 (1982)

[Wi83] Winskel, G., On powerdomains and modality, Lecture Notes in Computer Science 158 (also Theoretical Computer Science 36 (1985) 127-137)

[Wi83a] Winskel, G., A representation of SFP objects, manuscript, (1983)

[Wi84] Winskel, G., A complete proof system for SCCS with modal assertions, Technical Report 78 , Computer Laboratory, University of Cambridge, (1984)

[Wi84a] Winskel, G., Synchronization trees, Theoretical Computer Science 34 (1984)

[Wi86] Winskel, G., Event structures, Lecture Notes in Computer Science 255 (1986)

[Wi88] Winskel, G., An introduction to event structures, Lecture Notes in Computer Science 354 (1988)

[Wi89] Winskel, G., *Introduction to the Formal Semantics of Programming Languages*, MIT press (1989)

[Zh87] Zhang, G. Q., A logic for SFP, Draft paper, University of Cambridge (1987)

[Zh89] Zhang, G. Q., DI-domains as information systems, ICALP-1989, Italy (1989)