Fully Abstract Models of Programming Languages

Allen Stoughton

Doctor of Philosophy
University of Edinburgh
1986
Abstract

We present a theory of fully abstract denotational semantics of programming languages. Following initial algebra semantics, syntax is specified by many-sorted signatures, and models are universal algebras whose carriers are complete partial orders and operations are continuous. The informal requirement that the meanings of iteration and recursion constructs be least fixed points of appropriate unary derived operations is formally expressed by least fixed point constraints. Full abstraction is treated as a relation between models and notions of program equivalence (congruences over the term algebra). A model is fully abstract with reference to an equivalence if it identifies exactly the equivalent terms.

We give a necessary and sufficient condition for the existence of fully abstract models of programming languages. This condition is used to show that fully abstract models do not exist for two nondeterministic imperative languages: one with random assignment and the other with infinite output streams. It is also used to develop a model-theoretic necessary and sufficient condition that is used to show the existence of fully abstract models of two other programming languages: a simple typed applicative language and an imperative language with explicit storage allocation and higher and recursive types. A notion of full abstraction that is based upon contexts instead of simple terms is also studied, and the possibility of collapsing models that are too concrete, via continuous homomorphisms, to fully abstract models is considered.
Acknowledgements

It is a pleasure to acknowledge many stimulating discussions with my supervisor, Gordon Plotkin. Numerous conversations with Ilaria Castellani, Matthew Hennessy, Kim Larsen, David McCarty, Robin Milner, Peter Mosses, K.V.S. Prasad, Dave Schmidt, Oliver Schoet and Colin Stirling, as well as particular talks with Gerard Berry and Evelyn Nelson, were also helpful. Special thanks are due to Peter Dybjer and David McCarty for critically reading a draft of this thesis.

I was financially supported by a University of Edinburgh Studentship and a Research Assistantship in the Department of Computer Sciences, Chalmers University of Technology.
Declaration

This thesis was composed by myself and the work reported is my own.
Contents

Abstract 1

Acknowledgements 2

Declaration 3

1 Introduction 6
  1.1 Program Equivalence and Full Abstraction 6
  1.2 A Theory of Fully Abstract Models 11

2 Universal Algebras and Ordered Algebras 14
  2.1 Mathematical Conventions 15
  2.2 Many-Sorted Algebras 17
  2.3 Ordered Algebras 30
  2.4 Completion and Quotienting Theorems 44

3 Full Abstraction and Least Fixed Point Models 61
  3.1 Full Abstraction 62
  3.2 Least Fixed Point Models 65

4 Example Correct Models 73
  4.1 Defining Notions of Program Ordering 73
  4.2 A Metalanguage for Denotational Semantics 75
  4.3 The Programming Language PCF 77
Chapter 1

Introduction

1.1 Program Equivalence and Full Abstraction

Notions of program equivalence are fundamental to the theory and practice of programming languages. They are the semantic basis for program optimization and can be used to justify the correctness preserving transformations that are employed by program manipulation systems. Notions of program equivalence are generally substitutive in the sense that the results of embedding equivalent terms (program fragments) into a context (a term with "holes" in it) are also equivalent. Thus a programmer can replace fragments of a program by equivalent terms without considering the details of the whole program.

Program equivalences are typically defined according to the following paradigm. Certain terms are designated as programs, or directly executable terms, and their behaviour is defined. Then two terms are defined to be equivalent if and only if (iff) they have the same behaviour in all program contexts, i.e., iff one can be replaced by the other in a program without affecting its behaviour. Thus term equivalence is reduced to program behaviour.

By the behaviour of programs we mean the actions of programs that are visible to external observers. Program behaviours for a deterministic programming language might simply be functions from inputs to outputs, whereas behaviours
for languages with communicating processes might consist of communication histories. Much depends upon the level of detail that external observers are allowed to see. The distinction between terms and programs is often suggested by the syntactic categories of programming languages. For example, in an imperative language with statements and expressions the statements might be taken to be the programs, reflecting the view that expressions can only be executed as parts of statements. For languages with block structure, i.e., in which identifiers can be statically bound, it is common to take the closed terms as the programs.

It is also possible to consider notions of program ordering, i.e., notions of when one term should be considered less defined, or convergent, than another. Program orderings are typically defined by ordering the set of program behaviours and then defining one term to be less than another iff the behaviour of the first is less than that of the second in all program contexts.

Program behaviours and their orderings can be defined as abstractions of both operational and denotational semantics, although the literature is currently biased toward the use of operational semantics. Examples of the use of denotational semantics in this way are given in this thesis. Often there are multiple natural notions of behaviour that can be defined via a given semantics. Examples of behaviourally defined program orderings and equivalences can be found in [Milner1], [Milner3], [Plotkin1] and [HennessyPlotkin1].

Once a notion of program equivalence has been selected for a programming language, its properties must be determined and proof techniques found. Denotational semantics, as developed by Scott, Strachey and their followers (see [Stoy] for an introduction and extensive references), is a suitable framework for these activities. The idea is to reduce the equivalence of terms to the equality of their semantic values in appropriate models, i.e., to semantically capture the notion of program equivalence. Thus it is necessary to work with models that are equationally correct (or simply correct) in the sense that only equivalent terms are identified (mapped to the same semantic value). Models with
the ideal property that exactly the equivalent terms are identified are called *equationally-fully abstract* (or simply *fully abstract*).

Similarly, one can judge denotational semantics with reference to notions of program ordering. A model is said to be *inequationally correct* with reference to a program ordering iff one term is less than another in the program ordering whenever the meaning of the first is less than that of the second in the model, and *inequationally fully abstract* iff one term is less than another in the program ordering exactly when the meaning of the first is less than that of the second in the model.

For models to be useful for reasoning about program equivalences or orderings, it is necessary that their structure be understandable independently from those equivalences or orderings; informally, we call such models *natural*. For example, models synthesized using the standard constructions of denotational semantics are generally natural, in contrast to term models, i.e., models constructed from equivalence classes of terms, etc.

The idea of judging denotational semantics with reference to predefined notions of program ordering and equivalence is due to Milner [Milner1] and Plotkin [Plotkin1] and has been studied, for a variety of programming languages, by Abramsky, Berry, Curien, Hennessy and others. Research on full abstraction can be divided into two categories:

1. The synthesis and analysis of natural models.

2. The theoretical study of the conditions under which fully abstract models exist.

We consider each of these in turn.

For many programming languages, the standard techniques of denotational semantics yield natural models that are too concrete, i.e., correct but not fully abstract. Many common language features, such as functions of higher type, concurrency, storage allocation and data abstraction, are problematic. This phe-
nomenon was first noticed in connection with a simple applicative programming language, based upon the typed lambda calculus, called PCF (Programming Computable Functions). Plotkin [Plotkin1] showed that the natural continuous function model of PCF is correct, but not fully abstract, with reference to its standard notion of program equivalence, which is based upon the total evaluation of closed ground terms. This lack of full abstraction is due to the presence of certain "parallel" elements in the model, which are not realized by terms in the programming language. In fact, Plotkin showed that if a "parallel conditional" is added to the language then the continuous function model of this extended language is fully abstract. The problem of finding a natural fully abstract model of the original language is still open, although much progress has been made by Berry and Curien (see [Berry] for a summary). An important consequence of their work is the sequential algorithms model of PCF [BerryCurien], which is fully abstract with reference to an alternative notion of program equivalence that is sensitive to the order and extent of evaluation of function arguments.

Other examples of the search for natural fully abstract models can be found in [HennessyPlotkin1], which considers a simple parallel programming language; [Abramsky1] and [Abramsky2], which treat a nondeterministic applicative language with infinite streams; and [Brookes] and [Halpern], which deal with Algol-like languages. Many open problems exist.

The difficulty of finding natural fully abstract models for many programming languages has led to the theoretical study of the conditions under which fully abstract models exist. Proofs of the existence or nonexistence of fully abstract models of programming languages are relative, of course, to what count as models of those languages. Positive results spur on the search for natural models, whereas negative ones indicate that the class of models being considered must be widened or otherwise changed.

The study of the existence of fully abstract models can be carried out within the framework of initial algebra semantics [Scott] [ADJ1] [CourcelleNivat]. Pro-
gramming language syntax is specified in this framework by many sorted sig-
natures, whose sorts and operators correspond to the syntactic categories and
constructs, respectively, of programming languages, and models are universal
algebras whose carriers have certain order-theoretic structure and whose oper-
ations respect that structure. Usually the carriers are taken to be complete
partial orders and the operations continuous functions, but it is also possible
to work with weaker notions of continuity [AptPlotkin] [Plotkin2] or to gener-
alize from partial orders to categories [Lehmann] [Abramsky2]. The meanings
assigned by models to iteration and recursion constructs are normally required
to be least fixed points of appropriate unary derived operations. For example,
the meaning of a while-loop \texttt{while \textbf{E} do \textbf{S} od} should be the least fixed point of
\texttt{if \textbf{E} then \textbf{S}; -- else skip fi}. Many additional requirements may be set for models
of particular programming languages, e.g., extensionality for models of applica-
tive languages.

Positive results are typically proved via term model constructions. The first
use of such techniques was by Milner, who constructed a fully abstract model
of PCF [Milner2]. His construction was applied to a nondeterministic variant of
PCF in [AstesianoCosta]. Milner’s construction is somewhat ad hoc, and general
algebraic techniques for constructing term models were subsequently developed
in [Berry], for PCF, and [HennessyPlotkin2] and [Hennessy], for two variants
of CCS. Recently, Mulmuley has given a slightly more semantic construction
of a fully abstract model for PCF [Mulmuley]. His model is a retract of the
natural continuous function model of PCF that is based upon complete lattices.
(His construction fails for the usual model, based upon complete partial orders.)
Unfortunately, the retract is defined via the operational semantics of PCF and
the resulting model yields no new understanding of program equivalence.

The first negative result was proved by Apt and Plotkin [AptPlotkin] for a
nondeterministic imperative programming language with random assignment,
i.e., the facility for choosing an arbitrary natural number and assigning it to a
variable. They prove that there does not exist a fully abstract model that is based upon complete partial orders and continuous functions for this language. However, they are able to give a natural fully abstract model that is based upon a weaker notion of continuity. Abramsky, following this work, has proved a similar negative result for a nondeterministic applicative programming language with infinite streams [Abramsky3].

1.2 A Theory of Fully Abstract Models

All of the research described above has focused on full abstraction for specific programming languages. In this thesis we try to develop a theory of fully abstract models of programming languages that is applicable to programming languages in general. The following paragraphs summarize the contents of the thesis.

We begin by building a mathematical framework for studying full abstraction, based upon initial algebra semantics. As models we take complete ordered algebras, i.e., many-sorted universal algebras whose carriers are sort-indexed families of complete partial orders and operations are continuous functions. Following [CourcelleNivat], every signature is required to contain a distinguished nullary operator $\Omega$ of each sort, which stands for divergence or nontermination, and is interpreted as the least element of its sort in every model. Although programming languages rarely contain such constants explicitly, many languages for which divergence is possible in all syntactic categories do contain terms that the constants $\Omega$ can be modelled after, e.g., $\text{while true do skip od}$, in some imperative languages. Chapter 2 consists of the definitions and theorems concerning universal algebras and ordered algebras that will be needed in the sequel. In particular, we prove several quotienting and completion theorems that will be used in term model constructions.

Chapter 3 is devoted to the definitions and elementary properties of full
abstraction and least fixed point models. We consider three kinds of full abstraction (and also correctness): equational, contextual and inequational. The first two are as described above, and the third is the natural generalization of equational full abstraction from ordinary terms to contexts. Formally, notions of program equivalence are congruences over the term algebra, and notions of program ordering are substitutive pre-orderings over the term algebra in which the maximally divergent terms Ω are least elements. Least fixed point models are intended to assign iteration and recursion constructs meanings that are least fixed points of appropriate unary derived operations. Such requirements are formally expressed in our framework via least fixed point constraints, which specify that the meanings of certain terms should be the least upper bounds of the meanings of their syntactic approximations. We also consider contextually least fixed point models, which are the natural generalization of least fixed point models from terms to contexts.

In chapter 4 we study two programming languages within our framework. The first is PCF and the second is an imperative language with explicit storage allocation and higher and recursive types, which we call TIE. We give denotational semantics for both of these languages, define notions of program ordering and equivalence as abstractions of these models, in a uniform manner, and show that the models are inequationally correct with reference to these notions of ordering. In contrast, the model of PCF is already known not to be fully abstract and we conjecture that neither is our model of the second language.

In chapter 5 we give necessary and sufficient conditions for the existence of correct and fully abstract models, for each of the three kinds of correctness and full abstraction. The condition for the existence of inequationally fully abstract models is the cornerstone of these results: it is developed first, using a general term model construction, and the other conditions are derived from it. We also prove theorems concerning the existence of initial objects and the nonexistence of terminal objects in various categories of models.
Chapter 6 consists of simplified proofs of the negative results of [AptPlotkin] and [Abramsky3], using the condition for the existence of equational fully abstract models given in chapter 5. No model-theoretic reasoning is involved in these proofs. Instead of Abramsky's nondeterministic applicative language with streams, we actually work with a nondeterministic imperative language with infinite output streams. The notions of program equivalence for these languages are defined via operational semantics.

In chapter 7, we investigate two approaches to obtaining fully abstract models from correct ones. In the first, we use the condition for the existence of inequationally fully abstract models given in chapter 5 in order to develop useful necessary and sufficient conditions involving the existence of correct models. In the second, we consider the possibility of collapsing correct models, via continuous homomorphisms, to fully abstract ones. We show that this is impossible, in general, but give a sufficient condition for its possibility. Both of these approaches yield fully abstract models for the languages introduced in chapter 4, and, more generally, for languages whose notions of program ordering and equivalence are defined as abstractions of models using the technique of chapter 4.

Finally, in chapter 8, we consider the limitations of the thesis and the corresponding possibilities for further research.
Chapter 2

Universal Algebras and Ordered Algebras

This chapter introduces the definitions and theorems concerning universal algebras and ordered algebras that are the basis of the thesis. We begin, in section 2.1, by describing the (mostly standard) conventions of notation and terminology that will be followed in the sequel.

Sections 2.2 and 2.3 deal with the basics of many sorted algebras and ordered algebras, respectively. Most of the definitions and theorems in these sections are both standard and straightforward and detailed references will not be given. Those readers who are interested in the history of these ideas are referred to [Grätzer], for the universal algebra, and [Scott], [ADJ1], [CourcelleNivat] and [Nelson], for the work on ordered algebras. The exception to this is the definition and treatment of "unary-substitutive pre-orderings", which I believe to be new (see definition 2.2.24).

Section 2.4 consists of a completion theorem and two quotienting theorems for ordered algebras. The completion theorem is a variation of that of [CourcelleRaoult] and concerns the embedding of ordered algebras into complete ordered algebras in such a way that certain existing least upper bounds are preserved. For our results in chapters 5 and 7 we must preserve sets of exist-
ing least upper bounds that cannot be described by the usual families of subsets [CourcelleRaoult] (subset systems in the terminology of [ADJ2] and [Nelson]), which are defined uniformly for all ordered algebras. As a result, we work with families of subsets that are associated with individual ordered algebras. The quotienting theorems are taken from [CourcelleNivat] and [CourcelleRaoult].

2.1 Mathematical Conventions

The reader is assumed to be familiar with elementary set theory and category theory (see, e.g., [Kunen] and [Mac Lane]).

We identify the set of natural numbers \( \mathbb{N} \) with the ordinal \( \omega \), so that \( 0 = 0 \) and \( n = \{0,1,\ldots,n-1\} \), and write \( Tr \) for the set \( \{tt,ff\} \) of booleans.

Function space formation, \( X \to Y \), associates to the right and function application, \( f a \), to the left. We sometimes write \( Y^X \) for \( X \to Y \). For \( f: X \to Y \) and \( X' \subseteq X \), \( f X' \) is \( \{ f x \mid x \in X' \} \subseteq Y \), the image of \( X' \) under \( f \), and \( f \mid X' \) is \( \{ (x,y) \in f \mid x \in X' \} : X' \to Y \), the restriction of \( f \) to \( X' \). For a set \( X \), \( id_X: X \to X \) is the identity function, and for \( f: X \to Y \) and \( g: Y \to Z \), \( g \circ f: X \to Z \) is the composition of \( f \) and \( g \). The \( n \)th iterate, \( f^n \), of a function \( f: X \to X \) is defined by \( f^0 = id_X \) and \( f^{n+1} = f \circ f^n \).

For a set \( X \), the set \( X^* \) of finite sequences of elements of \( X \) is \( \bigcup_{n \in \omega} X^n \), and the set \( X^\infty \) of finite and infinite sequences of elements of \( X \) is \( X^* \cup X^\omega \). For \( a \in X^* \) (respectively, \( a \in X^\omega \)), \( \|a\| \), the cardinality of \( a \), doubles as the length of \( a \). Furthermore, \( \subseteq \) doubles as the is-a-prefix-of relation on sequences. We write \( \langle x_1,\ldots,x_n \rangle \) for elements of \( X^n \subseteq X^* \); in particular, \( \langle \rangle = \emptyset \in X^0 \) is the empty sequence. For \( a \in X^* \) and \( b \in X^* \) (respectively, \( b \in X^\omega \)), the concatenation of \( a \) and \( b \), \( a b \in X^* \) (respectively, \( a b \in X^\omega \)), is

\[
a \cup \{ \langle n + \|a\|, x \rangle \mid \langle n, x \rangle \in b \}.\]

The product \( D_1 \times \cdots \times D_n \) of sets \( D_1,\ldots,D_n \), \( n \leq 0 \), is \( \{ \langle d_1,\ldots,d_n \rangle \mid \)
Thus, if \( n = 0 \) then \( D_1 \times \cdots \times D_n = \{()\} \). The projection functions 
\[ \pi_i : D_1 \times \cdots \times D_n \rightarrow D_i, 1 \leq i \leq n, \]
are defined by \( \pi_i (d_1, \ldots, d_n) = d_i \).

We write \( \mathcal{P} X \) for the powerset of a set \( X \), i.e., the set of all subsets of \( X \).

A binary relation over a set is a pre-ordering iff it is reflexive and transitive, a partial ordering iff it is an antisymmetric pre-ordering, and an equivalence relation iff it is a symmetric pre-ordering. If \( R \) is a relation over \( X \) then we write \( R^* \) for the reflexive-transitive closure of \( R \). If \( \leq \) is a pre-ordering then we write \( \geq \) for its inverse (\( x \geq y \) iff \( y \leq x \)). Other examples of the notation for inverses are \( \geq^f \) for \( \leq^f \), and \( \supseteq_A \) for \( \subseteq_A \). Note that the inverse is not always the exact mirror image of the original ordering. If \( \equiv \) is an equivalence relation over \( X \) then \( X/\equiv \), the quotient of \( X \) by \( \equiv \), is \( \{[x] | x \in X\} \), where \( [x] \equiv \), the \( \equiv \)-equivalence class of \( x \), is \( \{x' \in X | x' \equiv x\} \). Sometimes we drop the relation \( \equiv \) from \([x] \equiv \).

As we will make extensive use of many-sorted algebras, we will frequently need to manipulate families of (structured) sets. Many operations and concepts extend naturally from sets to families of sets, in a pointwise manner. For example, if \( A \) and \( B \) are \( X \)-indexed families of sets, i.e., functions with domain \( X \), then a function \( f : A \rightarrow B \) is an \( X \)-indexed family of functions \( f_x : A_x \rightarrow B_x, x \in X; A \subseteq B \) iff \( A_x \subseteq B_x \), for all \( x \in X; \) and \( (A \cap B)_x = A_x \cap B_x \), for all \( x \in X \).

We will make use of these and other such extensions without explicit comment.

We often give inductive definitions of sets, i.e., we define a set \( X \) to be the least set (under the subset relation) satisfying certain closure conditions. A proof by induction over \( X \) of a proposition \( \forall x \in X \phi(x) \) consists of showing that the set \( Y = \{x \in X | \phi(x)\} \) satisfies the closure conditions, since, by the leastness of \( X \), we can then conclude that \( Y = X \). Induction over the natural numbers and structural induction over term algebras (see definition 2.2.5) are special cases of this general principle.
2.2 Many-Sorted Algebras

This section contains the definitions and results concerning many-sorted algebras that will be used in the sequel. We begin with the definitions of signatures, algebras, homomorphisms and subalgebras. The initial or term algebra is then defined, followed by the definition of reachability. Substitutive and Ω-least pre-orderings over algebras are then considered. Next, derived operations are introduced, leading to the important notion of unary-substitutive pre-orderings. Several results relating unary-substitutivity and substitutivity then follow, and the section concludes with two lemmas concerning the relations over the term algebra that are induced by relations over algebras.

Definition 2.2.1 A signature Σ consists of a set of sorts S, a set of operators Σ, and a function from Σ to (S* × S), which assigns types to operators. We write s1 × ... × sn → s' for n-ary types ⟨⟨s1, ..., sn⟩, s'⟩; unary types ⟨⟨s1⟩, s'⟩ are written s1 → s', and nullary types ⟨⟨⟩⟩, s'⟩ as s'. In addition, each signature contains a distinguished nullary operator Ωs of type s, for each s ∈ S.

We use Σ to stand both for signatures and their sets of operators. The operators Ωs may be thought of as representing divergence or nontermination. We often drop the sort s from Ωs.

Definition 2.2.2 A Σ-algebra A is an S-indexed family of sets A (the carrier of A) together with an operation σA: As1 × ... × Asn → As', for each σ ∈ Σ of type s1 × ... × sn → s'. A homomorphism h: A → B over algebras is a function h: A → B such that for all σ ∈ Σ of type s1 × ... × sn → s',

\[ h_{s'}σ_A(a_1, \ldots, a_n) = σ_B(h_{s_1}a_1, \ldots, h_{s_n}a_n), \]

for all a_i ∈ As_i, 1 ≤ i ≤ n.

We use uppercase script letters (A, B, etc.) to denote algebras and the corresponding italic letters (A, B, etc.) to stand for their carriers. We often
drop the algebra $\mathcal{A}$ from $\sigma_\mathcal{A}$, and write $\sigma$, $\sigma a$ and $a_1 \sigma a_2$, instead of $\sigma(\cdot)$, $\sigma(a)$ and $\sigma(a_1, a_2)$, for nullary, unary and binary operations, respectively. As usual, if $\Phi(\cdot)$ is an operation on algebras then we write $\Phi(\mathcal{A})$ for the carrier of $\Phi(\mathcal{A})$.

**Definition 2.2.3** For algebras $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}$ is a subalgebra of $\mathcal{B}$ iff $\mathcal{A} \subseteq \mathcal{B}$ and for all $\sigma \in \Sigma$ of type $s_1 \times \cdots \times s_n \rightarrow s'$ and $a_i \in A_{s_i}$, $1 \leq i \leq n$,

$$\sigma_\mathcal{A}(a_1, \ldots, a_n) = \sigma_\mathcal{B}(a_1, \ldots, a_n).$$

We write $\mathcal{A} \subseteq \mathcal{B}$ for $\mathcal{A}$ is a subalgebra of $\mathcal{B}$.

A consequence of this definition is that $\mathcal{A}$ is a subalgebra of $\mathcal{B}$ iff $\mathcal{A} \subseteq \mathcal{B}$ and the inclusion map from $\mathcal{A}$ to $\mathcal{B}$ is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. If $\mathcal{A}$ is an algebra and $\mathcal{B} \subseteq \mathcal{A}$ then by $\mathcal{B}$ is a subalgebra of $\mathcal{A}$ we mean that $\mathcal{B}$ is closed under the operations of $\mathcal{A}$. Note that the $\subseteq$ relation over the class of algebras is a partial ordering.

**Definition 2.2.4** If $f: \mathcal{A} \rightarrow \mathcal{B}$ is a homomorphism then $f \mathcal{A}$, the subalgebra of $\mathcal{B}$ induced by $f$, consists of $f \mathcal{A}$, together with the restrictions of the operations of $\mathcal{B}$ to $f \mathcal{A}$.

The set $f \mathcal{A}$ is closed under the operations of $\mathcal{B}$, since if $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $a_i \in A_{s_i}$, $1 \leq i \leq n$, then

$$\sigma_\mathcal{B}(f_{s_1} a_1, \ldots, f_{s_n} a_n) = f_{s'} \sigma_\mathcal{A}(a_1, \ldots, a_n).$$

Note that $f$ is also a homomorphism from $\mathcal{A}$ to $f \mathcal{A}$.

**Definition 2.2.5** We define the term algebra $T_\Sigma$ (or simply $T$) as follows. Its carrier $T$ is least such that if $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $t_i \in T_{s_i}$, $1 \leq i \leq n$, then $\langle \sigma, \langle t_1, \ldots, t_n \rangle \rangle \in T$. If $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ then the operation $\sigma_T$ is defined by $\sigma_T \langle t_1, \ldots, t_n \rangle = \langle \sigma, \langle t_1, \ldots, t_n \rangle \rangle$.

A standard result then easily follows.
Lemma 2.2.6 The term algebra $T$ is initial in the category of algebras and homomorphisms.

Definition 2.2.7 For an algebra $A$, we write $M_A$ (or simply $M$) for the unique homomorphism from $T$ to $A$.

Here $M$ stands for "meaning" and can be thought of as the meaning or semantic function from syntax to semantics. An easy application of lemma 2.2.6 is that $M_A t = M_B s$, for all $t \in T$, $s \in S$, if $A$ is a subalgebra of $B$.

Definition 2.2.8 An algebra $A$ is reachable iff $M_A T = A$, i.e., every element of $A$ is definable by a term.

An equivalent definition is that an algebra is reachable iff it has no proper subalgebras. An obvious consequence of this definition is that $T$ itself is reachable.

We now consider several kinds of relations over algebras.

Definition 2.2.9 If $A$ is an algebra and $R$ is a relation over $A$ then $R$ is substitutive iff the operations of $A$ respect $R$: for all $\sigma \in \Sigma$ of type $s_1 \times \cdots \times s_n \to s'$ and $a_i, a'_i \in A_{s_i}, 1 \leq i \leq n$,

$$a_i R_{s_i} a'_i, 1 \leq i \leq n, \text{ then } \sigma(a_1, \ldots, a_n) R_{s'} \sigma(a'_1, \ldots, a'_n).$$

As usual, substitutive equivalence relations are called congruences.

It is easy to see that if $\leq$ is a substitutive pre-ordering over $A$ then $\leq \cap \geq$ is a congruence. Note that if $R$ is a substitutive pre-ordering (respectively, partial ordering, equivalence relation) over $A$, and $B$ is a subalgebra of $A$, then the restriction of $R$ to $B$ is a substitutive pre-ordering (respectively, partial ordering, equivalence relation) over $B$.

Definition 2.2.10 If $f: D \to E$ is a function over sets then the equivalence relation over $D$ induced by $f$, $\equiv_f$, is defined by: $d_1 \equiv_f d_2$ iff $f d_1 = f d_2$. 

19
If \( f: A \to B \) is a homomorphism then \( \equiv_f \) is clearly a congruence over \( A \). We make use of this definition in giving the next one.

**Definition 2.2.11** For an algebra \( A \), the congruence over \( T \) induced by \( A \), \( \equiv_A \), is \( \equiv_M^A \).

Note that if \( A \subseteq B \) then \( \equiv_A = \equiv_B \).

**Definition 2.2.12** If \( A \) is an algebra and \( R \) is a pre-ordering over \( A \) then \( R \) is \( \Omega \)-least iff for all \( s \in S \) and \( a \in A_s \), \( \Omega_s R_s a \).

We will extensively use both \( \Omega \)-least substitutive pre-orderings and congruences. Note that if \( A \) is an algebra and \( R \) is a relation over \( A \) then there is a least \( \Omega \)-least substitutive pre-ordering containing \( R \), as well as a least congruence containing \( R \).

As there are no constraints concerning the \( \Omega \) operations on congruences, it is not surprising that not every congruence is induced by an \( \Omega \)-least substitutive pre-ordering, as the next lemma shows.

**Lemma 2.2.13** There is a signature \( \Sigma \) and a congruence \( \approx \) over \( T \) such that there is no \( \Omega \)-least substitutive pre-ordering \( \leq \) over \( T \) with the property that \( \approx = \leq \cap \geq \).

**Proof.** Let \( \Sigma \) over \( S = \{0,1\} \) have the following operators:

- \( \Omega_0 \) and \( a \) of type 0;
- \( \Omega_1 \) of type 1;
- \( f \) of type \( 0 \to 1 \).

Let \( \approx \) be the least congruence over \( T \) with the property that \( \Omega_1 \approx f a \). Then, no other unequal terms are congruent.
Suppose, towards a contradiction, that $a \leq b$ as in the statement of the lemma exists. Then

$$\Omega_1 \leq_1 f \Omega_0 \leq_1 f a \leq_1 \Omega_1,$$

showing that $\Omega_1 \approx_1 f \Omega_0$—a contradiction. \qed

We now consider derived operators, which are defined via the free algebra over a set of generators.

**Definition 2.2.14** For an $S$-indexed family $X$ of disjoint sets of context variables not occurring in $\Sigma$, $\Sigma(X)$ is the signature formed by adding nullary operators $x$ of type $s$, for each $x \in X_s$, $s \in S$, to $\Sigma$. The $\Sigma$-algebra $\mathcal{T}_{\Sigma}(X)$ (or simply $\mathcal{T}(X)$) is the restriction of $\mathcal{T}_{\Sigma}(X)$ to a $\Sigma$-algebra.

We often use the letter $c$, for "context", to stand for elements of $T(X)$. The standard result that $T(X)$ is the free algebra generated by $X$ now easily follows.

**Lemma 2.2.15** Define $f: X \rightarrow T(X)$ by $f_s x = (x, \langle \rangle)$. If $A$ is an algebra and $g: X \rightarrow A$ then there exists a unique homomorphism $h: T(X) \rightarrow A$ such that $g = h \circ f$:

\[ X \xrightarrow{f} T(X) \xrightarrow{g} A \]

\[ h \]

**Definition 2.2.16** For a signature $\Sigma$, $V_\Sigma$ (or simply $V$) is an $S$-indexed family of disjoint, countably-infinite sets of context variables not occurring in $\Sigma$. We often view a set $Y$ of variables ($Y \subseteq \bigcup_{s \in S} V_s$) as the $S$-indexed family of variables $Y'$ defined by $Y'_s = V_s \cap Y$. 

21
Definition 2.2.17 A derived operator \( c \) of type \( s_1 \times \cdots \times s_n \rightarrow s' \) is a pair
\[
\langle c, (v_1, \ldots, v_n) \rangle,
\]
where the \( v_i \in V_{S_i} \) are distinct variables and \( c \in T(\{v_1, \ldots, v_n\})_{s'} \). We write \( c[v_1, \ldots, v_n] \) for derived operators \( \langle c, (v_1, \ldots, v_n) \rangle \). For an algebra \( A \), the derived operation
\[
c_A[v_1, \ldots, v_n] : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_{s'}
\]
is defined by
\[
c_A[v_1, \ldots, v_n](a_1, \ldots, a_n) = h_{s'} c,
\]
where \( h : T(\{v_1, \ldots, v_n\}) \rightarrow A \) is defined via Lemma 2.2.15, by taking \( \{v_1, \ldots, v_n\} \) for \( X \) and defining \( g \) by \( g_{S_i} v_i = a_i \), \( 1 \leq i \leq n \).

We write \( c \) for \( c[v_1, \ldots, v_n] \) when the order of the variables is clear from the context, and we often drop the algebra \( A \) from \( c_A \).

Definition 2.2.18 A derived operator \( c[v_1, \ldots, v_n] \) of type \( s_1 \times \cdots \times s_n \rightarrow s' \) is a projection iff \( c = v_i \) and \( s' = s_i \), for some \( 1 \leq i \leq n \), and a constant iff \( c \in T_{s'} \).

Five standard lemmas concerning derived operations now follow.

Lemma 2.2.19 Homomorphisms preserve derived operations and derived operations respect substitutive pre-orderings.

Proof. Both parts of the lemma are easy structural inductions over \( T(X) \), for appropriate sets of variables \( X \). \( \square \)

The next three lemmas show how derived operators can be constructed from constant and projection derived operators and ordinary operators.

Lemma 2.2.20 Suppose \( A \) is an algebra and \( a_i \in A_{S_i}, 1 \leq i \leq n \).

1. For each projection \( v_i[v_1, \ldots, v_n] \) of type \( s_1 \times \cdots \times s_n \rightarrow s_i \),
\[
v_{iA}(a_1, \ldots, a_n) = a_i.
\]
2. For each constant \( t[v_1, \ldots, v_n] \) of type \( s_1 \times \cdots \times s_n \rightarrow s' \),

\[
t_A(a_1, \ldots, a_n) = M_{s'} t.
\]

**Proof.** Part 1 is immediate from definition 2.2.17, and part 2 is a simple structural induction over \( T \). \( \square \)

**Lemma 2.2.21** If \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \rightarrow s \), \( c_i[v_1, \ldots, v_m], \ 1 \leq i \leq n \), are derived operators of type \( s'_1 \times \cdots \times s'_m \rightarrow s_i \), \( A \) is an algebra and \( a_j \in A_{s'_j}, \ 1 \leq j \leq m \), then

\[
(\sigma_T(\{v_1, \ldots, v_m\}) \langle c_1, \ldots, c_n \rangle)[v_1, \ldots, v_m]
\]

is a derived operator of type \( s'_1 \times \cdots \times s'_m \rightarrow s \) and

\[
(\sigma(c_1, \ldots, c_n))_A(a_1, \ldots, a_m) = \sigma_A(c_{1A} \langle a_1, \ldots, a_m \rangle, \ldots, c_{nA} \langle a_1, \ldots, a_m \rangle).
\]

**Proof.** Immediate from definition 2.2.17. \( \square \)

**Lemma 2.2.22** If \( c[v_1, \ldots, v_m] \) is a derived operator of type \( s_1 \times \cdots \times s_n \rightarrow s \), \( c'_i[v'_1, \ldots, v'_m], \ 1 \leq i \leq n \), are derived operators of type \( s'_1 \times \cdots \times s'_m \rightarrow s_i \), \( A \) is an algebra and \( a_j \in A_{s'_j}, \ 1 \leq j \leq m \), then

\[
(c_T(\{v'_1, \ldots, v'_m\}) \langle c_1, \ldots, c_n \rangle)[v'_1, \ldots, v'_m]
\]

is a derived operator of type \( s'_1 \times \cdots \times s'_m \rightarrow s \) and

\[
(c(c_1, \ldots, c_n))_A(a_1, \ldots, a_m) = c_A(c_{1A} \langle a_1, \ldots, a_m \rangle, \ldots, c_{nA} \langle a_1, \ldots, a_m \rangle).
\]

**Proof.** An easy structural induction over \( T(\{v_1, \ldots, v_n\}) \). \( \square \)

**Lemma 2.2.23** If \( A \) is a subalgebra of \( B \) then for all derived operators \( c[v_1, \ldots, v_m] \) of type \( s_1 \times \cdots \times s_n \rightarrow s' \) and \( a_i \in A_{s_i}, \ 1 \leq i \leq n \),

\[
c_A(a_1, \ldots, a_n) = c_B(a_1, \ldots, a_n).
\]
Proof. Immediate from lemma 2.2.19 and the fact that the inclusion map from $A$ to $B$ is a homomorphism from $A$ to $B$. \qed

It is now possible to define a weaker notion of substitutivity that, as we shall see, arises naturally.

**Definition 2.2.24** If $A$ is an algebra and $R$ is a pre-ordering over $A$ then $R$ is unary-substitutive iff all unary derived operations respect $R$: for all derived operators $c[v]$ of type $s \to s'$ and $a, a' \in A$,\n
$$\text{if } a R a' \text{ then } c(a) R c(a').$$

We could, of course, define the notion of $n$-substitutive pre-orderings, which would be respected by $n$-ary derived operations, but we have no use for this generality in the sequel.

A consequence of lemma 2.2.23 is that if $R$ is a unary-substitutive pre-ordering (respectively, partial ordering, equivalence relation) over $A$, and $B$ is a subalgebra of $A$, then the restriction of $R$ to $B$ is a unary-substitutive pre-ordering (respectively, partial ordering, equivalence relation) over $B$. If $\leq$ is a unary-substitutive pre-ordering over an algebra $A$ then $(\leq \cap \geq)$ is a unary-substitutive equivalence relation over $A$.

We now define an operation that will be employed in the definitions of notions of program ordering and equivalence of chapters 4 and 6.

**Definition 2.2.25** If $P \subseteq S$, $A$ is an algebra and $R$ is a pre-ordering over $A|P$ then $R^c$ is the relation over $A$ defined by: $a R^c a'$ iff $c(a) R_p c(a')$, for all derived operators $c[v]$ of type $s \to p$, $p \in P$.

If $R$ is a pre-ordering over $A$ then $P$ will implicitly be $S$ in the definition of $R^c$.

Subsets $P \subseteq S$ can be thought of as consisting of program sorts, and derived operators $c[v]$ of type $s \to p$ as program contexts. Thus if $R$ is a relation over
T \mid P$ (programs) then two terms are related by $R^c$ iff they are related by $R$ in all program contexts.

The next lemma shows that, as might be guessed, $R^c$ is a unary-substitutive pre-ordering over $A$.

**Lemma 2.2.26** If $P \subseteq S$, $A$ is an algebra and $R$ is a pre-ordering (respectively, equivalence relation) over $A \mid P$ then $R^c$ is the greatest unary-substitutive pre-ordering (respectively, equivalence relation) over $A$ whose restriction to $P$ is included in $R$.

**Proof.** It is easy to see that $R^c$ is a pre-ordering over $A$ and that it is symmetric if $R$ is symmetric. The inclusion of the restriction of $R^c$ to $P$ in $R$ follows from the existence of projection derived operators $v[v]$ of type $p \rightarrow p$, for all $p \in P$. Next, we show that $R^c$ is unary-substitutive. Suppose $a_1 R^c_S a_2$ and $c[v]$ is a derived operator of type $s \rightarrow s'$. We must show that $c(a_1) R^c_S c(a_2)$. Let $p \in P$ and $c'[v']$ be a derived operator of type $s' \rightarrow p$. Then, $(c'(c))[v]$ is a derived operator of type $s \rightarrow p$ and

$$c'(c(a_1)) = (c'(c))(a_1) R_p (c'(c))(a_2) = c'(c(a_2)),$$

by lemma 2.2.22, and by the assumption that $a_1 R^c_S a_2$. Finally, suppose $R'$ is a unary-substitutive pre-ordering (respectively, equivalence relation) over $A$ whose restriction to $P$ is included in $R$; we must show that $R' \subseteq R^c$. Let $a_1 R'_S a_2$. If $p \in P$ and $c[v]$ is a derived operator of type $s \rightarrow p$ then $c(a_1) R' p c(a_2)$, and thus $c(a_1) R_p c(a_2)$. Thus $a_1 R^c_S a_2$, as required. $\Box$

It is easy to see that if $P \subseteq S$, $A$ is an algebra and $\leq$ is a pre-ordering over $A \mid P$ then $(\leq \cap \geq)^c = (\leq^c \cap \geq^c)$.

**Lemma 2.2.27** If $\equiv$ is a unary-substitutive equivalence relation over an algebra $A$ and $\leq$ is a pre-ordering over $A$ that induces $\equiv$ then $\leq^c$ also induces $\equiv$.

**Proof.** Since $\leq^c \subseteq \leq$, $\leq^c \cap \geq^c \subseteq \equiv$. For the opposite inclusion, suppose $a_1 \equiv_S a_2$, $s \in S$. To show that $a_1 \leq^c_S a_2$, let $c[v]$ be a derived operator of
type $s \rightarrow s'$. Then $c(a_1) \equiv_{g'} c(a_2)$, since $\equiv$ is unary-substitutive, and thus $c(a_1) \leq_{g'} c(a_2)$. Similarly, $a_2 \leq_{g} a_1$. □

The next lemma shows that, as mentioned above, unary-substitutivity is weaker than substitutivity. In fact there is even a unary-substitutive equivalence relation over an algebra such that every congruence over that algebra induces a different pre-ordering over $T$.

**Lemma 2.2.28** There is a signature $\Sigma$, an algebra $A$ and an $\Omega$-least unary-substitutive pre-ordering $\leq$ over $A$ such that:

1. $\leq$ is not substitutive.
2. The unary-substitutive equivalence relation $\equiv = \leq \cap \geq$ is not substitutive.
3. There does not exist a congruence $\equiv'$ over $A$ such that

$$M_3 t_1 \equiv_{g} M_3 t_2 \text{ iff } M_3 t_1 \equiv_{g} M_3 t_2,$$

for all $t_1, t_2 \in T_3$, $s \in S$.

**Proof.** Let $\Sigma$ over $S = \{0, 1, 2\}$ have the following operators:

- $\Omega_0$ of type 0;
- $\Omega_1$, $x$ and $y$ of type 1;
- $\Omega_2$ and $z$ of type 2;
- $+$ of type $0 \times 1 \rightarrow 2$.

Define the algebra $A$ as follows. Its carrier $A$ is defined by $A_0 = \{\Omega_0, w\}$, $A_1 = \{\Omega_1, x, y\}$ and $A_2 = \{\Omega_2, z\}$. All of the nullary operations have themselves as their values. The operation $+$ is bistrict with reference to the $\Omega$'s, i.e.,

$a + a' = \Omega_2$ if $a = \Omega_0$ or $a' = \Omega_1$; on non-$\Omega$ elements, it is defined by $w + x = z$ and $w + y = \Omega_2$. Note that the element $w$ of $A_0$ is not definable by a term. Let $\leq$ be the least $\Omega$-least pre-ordering over $A$ such that $x \leq_1 y \leq_1 x$: 26
Clearly the constant and projection unary derived operations respect \( \leq \). This leaves \((v + \Omega_1)[v], (v + x)[v] \) and \((v + y)[v] \) of type \(0 \rightarrow 2\) and \((\Omega_0 + v')[v']\) of type \(1 \rightarrow 2\). Since \(+\) is bistrict, \(v + \Omega_1\) and \(\Omega_0 + v'\) respect \(\leq\). The unary-substitutivity of \(\leq\) then follows, since

\[
\Omega_0 + x = \Omega_2 \leq_2 z = w + x
\]

and

\[
\Omega_0 + y = \Omega_2 \leq_2 \Omega_2 = w + y.
\]

Part 1 will follow immediately from part 2, and part 2 immediately from part 3. For part 3, suppose that such an \(\equiv\) exists. Then,

\[
x \equiv_1 y
\]

\[
\Rightarrow x \equiv_1 y
\]

\[
\Rightarrow z = w + x \equiv_2 w + y = \Omega_2
\]

\[
\Rightarrow z \equiv_2 \Omega_2,
\]

which is a contradiction. \(\square\)

As might be guessed from the proof of the previous lemma, a sufficient (but not necessary) condition for a unary-substitutive pre-ordering over an algebra to be substitutive is that the algebra be reachable. As an aid toward proving this, we first give a characterization of substitutivity, which will also be used in section 2.3.

**Lemma 2.2.29** Let \(A\) be an algebra and \(R\) a pre-ordering over \(A\). Then, \(R\) is substitutive iff for all derived operators \(c[v, v_1, \ldots, v_n]\) of type \(s \times s_1 \times \cdots \times s_n \rightarrow s'\), \(n \geq 0\), and \(a, a' \in A, B\), if \(a \equiv_2 a'\) then

\[
c(a, a_1, \ldots, a_n) \equiv_1 c(a', a_1, \ldots, a_n),\ \text{for all} \ a_i \in A, 1 \leq i \leq n.
\]
Proof. The “only if” direction follows from lemma 2.2.19 and the reflexivity of $R$. For the “if” direction, suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$, and $a_i, a'_i \in A_{s_i}$ have the property that $a_i \mathbin R_{s_i} a'_i$, $1 \leq i \leq n$. We must show that

$$\sigma(a_1, \ldots, a_n) \mathbin R_{s'} \sigma(a'_1, \ldots, a'_n).$$

If $n = 0$ then $\sigma R_{s'} \sigma$, since $R$ is reflexive; so, assume that $n \geq 1$. Since $R$ is transitive, it is sufficient to show that

$$\sigma(a_1, \ldots, a_n) \mathbin R_{s'} \sigma(a'_1, a_2, \ldots, a_n) \mathbin R_{s'} \sigma(a'_1, a'_2, a_3, \ldots, a_n) \mathbin R_{s'} \cdots \mathbin R_{s'} \sigma(a'_1, \ldots, a'_n).$$

We show a representative step in this chain:

$$\sigma(a'_1, \ldots, a'_{i-1}, a_i, a_{i+1}, \ldots, a_n) \mathbin R_{s'} \sigma(a'_1, \ldots, a'_{i-1}, a'_i, a_{i+1}, \ldots, a_n).$$

Let $v_i \in V_{s_i}$, $1 \leq i \leq n$, be distinct variables. Then,

$$(\sigma(v_1, \ldots, v_n))[v_i, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n]$$

is a derived operator of type

$$s_i \times s_1 \times \cdots \times s_{i-1} \times s_{i+1} \times \cdots \times s_n \rightarrow s',$$

and thus

$$\sigma(a'_1, \ldots, a'_{i-1}, a_i, a_{i+1}, \ldots, a_n) = (\sigma(v_1, \ldots, v_n))(a_i, a'_1, \ldots, a'_{i-1}, a_{i+1}, \ldots, a_n) \mathbin R_{s'} (\sigma(v_1, \ldots, v_n))(a'_1, a'_1, \ldots, a'_{i-1}, a_{i+1}, \ldots, a_n) = \sigma(a'_1, \ldots, a'_{i-1}, a'_i, a_{i+1}, \ldots, a_n),$$

since $a_i \mathbin R_{s_i} a'_i$. □
Lemma 2.2.30 *Unary-substitutive pre-orderings over reachable algebras are substitutive.*

**Proof.** Let $R$ be a unary-substitutive pre-ordering over a reachable algebra $A$. We make use of the characterization of substitutivity given by lemma 2.2.29. Suppose $c[v, v_1, \ldots, v_n]$ is a derived operator of type $s \times s_1 \times \cdots \times s_n \rightarrow s'$, $n \geq 0$, $a, a' \in A_s$, $a_i \in A_{s_i}$, $1 \leq i \leq n$, and $a R_s a'$. Since $A$ is reachable, there are $t_i \in T_{s_i}$ such that $a_i = M_{s_i} t_i$, $1 \leq i \leq n$. Then, $(c(v, t_1, \ldots, t_n))[v]$ is a derived operator of type $s \rightarrow s'$, and

\[
c(a, a_1, \ldots, a_n) \\
= \ (c(v, t_1, \ldots, t_n))(a) \\
R_{s'} \ (c(v, t_1, \ldots, t_n))(a') \\
= \ c(a', a_1, \ldots, a_n),
\]

since $R$ is unary-substitutive. □

Combining lemmas 2.2.26 and 2.2.30 we have that if $P \subseteq S$, $A$ is a reachable algebra and $R$ is a pre-ordering (respectively, equivalence relation) over $A | P$ then $R^c$ is the greatest substitutive pre-ordering (respectively, congruence) over $A$ whose restriction to $P$ is included in $R$.

This section concludes with two lemmas concerning the relations over $T$ that are induced by relations over the carriers of algebras.

Lemma 2.2.31 *Suppose $P \subseteq S$, $A$ is an algebra, $R$ is a pre-ordering over $A | P$, and $Q$ is the pre-ordering over $T | P$ defined by

\[
t_1 Q_p t_2 \text{ iff } M_p t_1 R_p M_p t_2.
\]

Then $R^c$ is a unary-substitutive pre-ordering over $A$, $Q^c$ is a substitutive pre-ordering over $T$, and

\[
t_1 Q^c_{s} t_2 \text{ iff } M_{s} t_1 R^c_{s} M_{s} t_2,
\]

for all $t_1, t_2 \in T_s$, $s \in S$.\]
Proof. The substitutivity of $Q^c$ follows from lemma 2.2.30, and

\[
    \begin{align*}
    t_1 Q^c_s t_2 & \quad \text{iff} \quad c(t_1) Q_p c(t_2), \text{ for all } c[v] \text{ of type } s \rightarrow p, p \in P \\
    & \quad \text{iff} \quad M_p c(t_1) R_p M_p c(t_2), \text{ for all } c[v] \text{ of type } s \rightarrow p, p \in P \\
    & \quad \text{iff} \quad c(M_s t_1) R_p c(M_s t_2), \text{ for all } c[v] \text{ of type } s \rightarrow p, p \in P \\
    & \quad \text{iff} \quad M_s t_1 R^c_s M_s t_2,
    \end{align*}
\]

for all $t_1, t_2 \in T_s$, $s \in S$. □

Lemma 2.2.32 Suppose $A$ is an algebra, $R$ is a pre-ordering over $A$, and $Q$ is the pre-ordering over $T$ defined by

\[
    t_1 Q_s t_2 \quad \text{iff} \quad M_s t_1 R_s M_s t_2.
\]

1. If $R$ is unary-substitutive then $Q$ is substitutive.

2. If $Q$ is substitutive then

\[
    t_1 Q_s t_2 \quad \text{iff} \quad M_s t_1 R^c_s M_s t_2,
\]

for all $t_1, t_2 \in T_s$, $s \in S$.

Proof. Immediate from lemma 2.2.31, with $P = S$. □

2.3 Ordered Algebras

This section consists of the basic definitions and results concerning ordered algebras that will be needed in the sequel. We begin by considering posets, cpo's, continuous functions and inductive pre-orderings. Ordered algebras, complete ordered algebras and inductive subalgebras are then defined, followed by two results concerning the derived operations of ordered algebras, and the definitions of the ordered term algebra and free ordered algebras. Generated inductive subalgebras and inductive reachability are then considered, followed by two
lemmas relating substitutivity and unary-substitutivity for complete ordered algebras. The section concludes with two lemmas concerning the pre-orderings over the terms algebra that are induced by inductive pre-orderings over complete ordered algebras.

**Definition 2.3.1** A pre-ordered set (preset) $D$ is a set $D$, together with a pre-ordering $\subseteq_D$ over $D$. If $d \in D$ and $D' \subseteq D$ then $d$ is an upper bound (ub) of $D'$ iff $d'' \subseteq_D d$, for all $d' \in D'$, and $d$ is a least upper bound (lub) of $D'$ iff $d$ is an ub of $D'$ and $d \subseteq_D d''$, for all ub's $d''$ of $D'$. We write $D' \subseteq_D d$, for $d$ is an ub of $D'$. A subset $D' \subseteq D$ is directed iff it is nonempty and every pair of elements of $D'$ has an ub in $D'$.

Note that lub's in presets may not be unique.

**Definition 2.3.2** A partially ordered set (poset) $D$ is a preset such that $\subseteq_D$ is a partial ordering. We denote the least element of a poset $D$, when it exists, by $\bot_D$. A complete partial ordering (cpo) $D$ is a poset with a least element, and such that every directed set $D'$ of $D$ has a lub $\bigcup_D D'$ in $D$.

We often drop the $D$ from $\subseteq_D$, $\bot_D$ and $\bigcup_D$ when it is clear from the context.

**Definition 2.3.3** A function $f: D \rightarrow E$ over posets is monotonic iff $f d \subseteq_E f d'$ if $d \subseteq_D d'$, an order-embedding iff $f d \subseteq_E f d'$ iff $d \subseteq_D d'$, and an order-isomorphism iff $f$ is a surjective order-embedding. Two posets are order-isomorphic iff there is an order-isomorphism from one to the other. A function $f: D \rightarrow E$ over posets that have least elements is strict iff $f \bot_D = \bot_E$. A function $f: D \rightarrow E$ over cpo's is continuous iff it is monotonic and $f \bigcup_D D' = \bigcup_E f D'$, for all directed sets $D' \subseteq D$.

Note that order-isomorphism coincides with isomorphism in the category of posets and monotonic functions, and that order-isomorphisms over cpo's are continuous.
We could just as well have worked with the larger category of \( \omega \)-complete partial orderings and \( \omega \)-continuous functions in this thesis. On the other hand, some of our constructions, e.g., the quotienting constructions of section 2.4, do not preserve \( \omega \)-algebraicity and consistent completeness, and so we cannot work in the smaller category of cpo’s with these additional properties.

**Definition 2.3.4** A pre-ordering over a poset \( (D, \sqsubseteq_D) \) is simply a pre-ordering over the set \( D \). A pre-ordering \( \leq \) over a cpo \( (D, \sqsubseteq_D) \) is inductive iff \( \sqsubseteq_D \subseteq \leq \) and whenever \( D' \) is a directed set in \( (D, \sqsubseteq_D) \) and \( D' \leq d, \sqcup D' \leq d \).

Note the requirement that \( \leq \) respect the ordering \( \sqsubseteq_D \) of \( D \).

**Definition 2.3.5** The product \( D_1 \times \cdots \times D_n \) of posets \( D_i, 1 \leq i \leq n, n \geq 0 \), is the product of their underlying sets \( D_i \), ordered componentwise:

\[
D_1 \times \cdots \times D_n \subseteq D'_1 \times \cdots \times D'_n \text{ iff } d_i \sqsubseteq d'_i, 1 \leq i \leq n.
\]

The projection functions \( \pi_i : D_1 \times \cdots \times D_n \to D_i \) are monotonic. A directed set \( D' \subseteq D_1 \times \cdots \times D_n \) has a lub iff the directed sets \( \pi_i D', 1 \leq i \leq n, \) have lub’s, and \( \sqcup \pi_1 D', \ldots, \sqcup \pi_n D' \) is the lub of \( D' \), when it exists. Thus, if all of the \( D_i \)'s are cpo’s then so is \( D_1 \times \cdots \times D_n \), and the projection functions are continuous. If \( D'_i \subseteq D_i, 1 \leq i \leq n, \) are directed sets then so is \( D'_1 \times \cdots \times D'_n \). Finally, if \( f : D_1 \times \cdots \times D_n \to E \) is a monotonic function, for cpo’s \( D_i, 1 \leq i \leq n, \) and \( E \), then \( f \) is continuous iff for all directed sets \( D'_i \subseteq D_i, 1 \leq i \leq n, \)

\[
f(\sqcup D'_1, \ldots, \sqcup D'_n) = \sqcup f(D'_1 \times \cdots \times D'_n).
\]

**Definition 2.3.6** If \( D \) and \( E \) are posets then \( D \to^m E \) is the poset of monotonic functions from \( D \) to \( E \), with the pointwise ordering:

\[
f \sqsubseteq g \text{ iff for all } d \in D, f d \sqsubseteq g d.
\]

If \( D \) and \( E \) are cpo’s then \( D \to^c E \) is the cpo of continuous functions from \( D \) to \( E \), ordered pointwise.
The constantly \( \bot \) function is the least element of \( D \to E \) and if \( F \subseteq D \to E \) is a directed set then \( \bigcup F \) is the least element of \( \{ f \mid f \in F \} \), for all \( d \in D \).

**Definition 2.3.7** An ordered \( \Sigma \)-algebra \( A \) is an \( S \)-indexed family of posets \( A \) (the carrier of \( A \)), together with a monotonic operation \( \sigma_A : A_{s_1} \times \cdots \times A_{s_n} \to A_{s'} \), for each \( \sigma \in \Sigma \) of type \( s_1 \times \cdots \times s_n \to s' \), and such that \( \Omega_{s} (\bot) \) is the least element of \( A_s \), for all \( s \in S \). Such an \( A \) is complete iff each \( A_s \) is a cpo and each \( \sigma_A \) is continuous.

Ordered algebras can be viewed as algebras by forgetting the partial orderings, and we will often do so without explicit comment. Thus, for an ordered algebra \( A \), \( A \) will stand for both the carrier of \( A \) (a family of posets) and for the carrier of the underlying algebra (a family of sets). We write \( \subseteq_A \) for the family of posets \( \subseteq_{(A_s)} \), \( s \in S \). Thus \( \subseteq_A \) is a partial ordering over \( A \). As usual, if \( \Phi(\cdot) \) is an operation on ordered algebras then we write \( \Phi(A) \) for the carrier of \( \Phi(\cdot) \).

For example, we call an ordered algebra reachable iff its underlying algebra is reachable (cf., inductively reachable complete ordered algebras, definition 2.3.30). Note that a homomorphism \( h : A \to B \) over ordered algebras (i.e., a homomorphism over the underlying algebras) is strict, since for all \( s \in S \),

\[
h_s \bot_{A_s} = h_s \Omega_A = \Omega_B = \bot_{B_s}.
\]

**Definition 2.3.8** For complete ordered algebras \( A \) and \( B \), \( A \) is an inductive subalgebra of \( B \) iff \( A \) is a subalgebra of \( B \), and for all \( s \in S \), \( \subseteq_A \) is the restriction of \( \subseteq_B \) to \( A_s \) and \( \bigcup_{A_s} A' = \bigcup_{B_s} A' \), whenever \( A' \subseteq A_s \) is a directed set. We write \( A \leq B \) for \( A \) is an inductive subalgebra of \( B \).

If \( A \) is a complete ordered algebra and \( B \subseteq A \) then by \( B \) is an inductive subalgebra of \( A \) we mean that \( B \) is a subalgebra of \( A \) and \( \bigcup_{A_s} B' \subseteq B_s \), whenever \( B' \subseteq B_s \) is a directed set in \( A_s \). This definition is sensible since if \( B \) is an inductive subalgebra of \( A \) then the complete ordered algebra \( B \) consisting of \( B \),
together with the restrictions of the operations and partial orderings of \( A \) to \( B \), is indeed an inductive subalgebra of \( A \). Note that the relation \( \leq \) over the class of complete ordered algebras is a partial ordering.

**Definition 2.3.9** An order-embedding \( h: A \to B \) over ordered algebras is a homomorphism such that \( h: A \to B \) is an order-embedding. An order-isomorphism over ordered algebras is a surjective order-embedding. Two ordered algebras are order-isomorphic iff there is an order-isomorphism from one to the other.

Note that order-isomorphism coincides with isomorphism in the category of ordered algebras and monotonic homomorphisms. Furthermore, if \( h: A \to B \) is an order-isomorphism over complete ordered algebras then \( h \) is continuous. A consequence of the above definitions is that for complete ordered algebras \( A \) and \( B \), \( A \) is an inductive subalgebra of \( B \) iff \( A \subseteq B \) and the inclusion from \( A \) to \( B \) is a continuous order-embedding from \( A \) to \( B \).

**Definition 2.3.10** For an ordered algebra \( A \), the \( \Omega \)-least substitutive pre-ordering over \( T \) induced by \( A \), \( A_\preceq \), is defined by:

\[
t_1 \preceq t_2 \text{ iff } M_s t_1 \preceq_A M_s t_2.
\]

Note that for any ordered algebra \( A \), \( \equiv_A = (\preceq_A \cap \succeq_A) \), and that if \( A \) is a subalgebra of an ordered algebra \( B \) then \( \preceq_A = \preceq_B \).

**Definition 2.3.11** If \( f: D \to E \) is a monotonic function over posets then the pre-ordering over \( D \) induced by \( f \), \( \preceq_f \), is defined by:

\[
d_1 \preceq_f d_2 \text{ iff } f d_1 \preceq_E f d_2.
\]

Clearly \( \preceq_f \) respects the ordering of \( D \), and if \( f \) is a continuous function over cpo's then \( \preceq_f \) is inductive. Furthermore, if \( h: A \to B \) is a monotonic homomorphism over ordered algebras then \( \preceq_h \) is a substitutive pre-ordering over \( A \).
Lemma 2.3.12 If $A$ is an inductive subalgebra of $B$ and $\leq$ is a substitutive (respectively, unary-substitutive) inductive pre-ordering over $B$ then the restriction of $\leq$ to $A$ is a substitutive (respectively, unary-substitutive) inductive pre-ordering over $A$.

Proof. Immediate from the definitions and lemma 2.2.23. □

Two results concerning derived operations of ordered algebras now follow.

Lemma 2.3.13 Derived operations of ordered algebras are monotonic and derived operations of complete ordered algebras are continuous.

Proof. Both parts are easy and standard structural inductions over $T(X)$, for appropriate X's. □

Lemma 2.3.14 If $A$ is a complete ordered algebra and $\leq$ is an inductive pre-ordering over $A \mid P$, for $P \subseteq S$, then $\leq^c$ is a unary-substitutive inductive pre-ordering over $A$.

Proof. By lemma 2.2.26, it is sufficient to show that $\leq^c$ is inductive. We begin by showing that $\subseteq A \subseteq \leq^c$. Let $a, a' \in A_S$, $s \in S$, and $a \subseteq A_S a'$. If $c[v]$ is a derived operator of type $s \rightarrow p$, $p \in P$, then $c(a) \subseteq A_p c(a')$, by lemma 2.3.13, and thus $c(a) \leq p c(a')$, since $\leq$ is inductive. Thus $a \leq^c a'$, as required. Now, suppose $A' \subseteq A_S$ is a directed set, $a \in A_S$ and $A' \leq^c a$. If $c[v]$ is a derived operator of type $s \rightarrow p$, $p \in P$, then

$$c(\bigcup A') = \bigcup \{ c(a') \mid a' \in A' \} \leq_p c(a),$$

by lemma 2.3.13, and since $A' \leq^c S a$ and $\leq$ is inductive. Thus $\bigcup A' \leq^c S a$, as required. □

We now give a definition and two lemmas in preparation for the definition of the ordered term algebra.

Definition 2.3.15 Let $\leq^\Omega$ be the least $\Omega$-least substitutive pre-ordering over $T$. 

35
The next lemma shows that one term is less than another in \( \leq^\Omega \) iff the second can be formed by replacing occurrences of \( \Omega \) in the first by terms.

**Lemma 2.3.16** For all \( s \in S \) and \( t, t' \in T_s \), \( t \leq^\Omega_s t' \) iff \((\dagger)\) \( t = \Omega_s \) or there is a \( \sigma \in \Sigma \) of type \( s_1 \times \cdots \times s_n \to s \) and \( t_i, t'_i \in T_{s_i} \), \( 1 \leq i \leq n \), such that \( t = \sigma(t_1, \ldots, t_n) \), \( t' = \sigma(t'_1, \ldots, t'_n) \) and \( t_i \leq^\Omega_{s_i} t'_i \), \( 1 \leq i \leq n \).

**Proof.** Define a relation \( R \) over \( T \) by: \( t R t' \) iff \((\dagger)\) holds. It is sufficient to show that \( \leq^\Omega = R \). Clearly \( R \subseteq \leq^\Omega \). Furthermore, it is easy to see that \( R \) is an \( \Omega \)-least substitutive pre-ordering over \( T \). Thus, by the leastness of \( \leq^\Omega \), \( \leq^\Omega \subseteq R \). \( \square \)

**Lemma 2.3.17** The relation \( \leq^\Omega \) is a partial ordering.

**Proof.** An easy structural induction over \( t \), using lemma 2.3.16, shows that for all \( t, t' \in T_s \), \( s \in S \), if \( t \leq^\Omega_s t' \) and \( t' \leq^\Omega_s t \) then \( t = t' \). \( \square \)

**Definition 2.3.18** The ordered algebra \( OT_{\Sigma} \) (or simply \( OT \)) consists of \( T \) ordered by \( \leq^\Omega \).

**Lemma 2.3.19** The ordered algebra \( OT \) is initial in the category of ordered algebras and monotonic homomorphisms.

**Proof.** It is sufficient to show that \( M_A : OT \to A \) is monotonic. Suppose \( t \leq^\Omega t' \). Then \( t \subseteq_{A_S} t' \), by the leastness of \( \leq^\Omega \), and so \( M_S t \subseteq_{A_S} M_S t' \), by the definition of \( \subseteq_A \). \( \square \)

We can now generalize from the initial ordered algebra to free ordered algebras.

**Definition 2.3.20** If \( X \) is an \( S \)-indexed family of disjoint countably-infinite sets of context variables not occurring in \( \Sigma \) then \( OT_{\Sigma}(X) \) (or simply \( OT(X) \)) is the restriction of \( OT_{\Sigma}(X) \) to an ordered \( \Sigma \)-algebra.
The standard result that $OT(X)$ is the free ordered algebra generated by $X$ now easily follows.

**Lemma 2.3.21** Define $f : X \to OT(X)$ by $f_s x = (x, \langle \rangle)$. If $A$ is an ordered algebra and $g : X \to A$ then there exists a unique monotonic homomorphism $h : OT(X) \to A$ such that $g = h \circ f$:

![Diagram](image)

The next lemma shows that we could have defined derived operations over ordered algebras via free ordered algebras, instead of free algebras.

**Lemma 2.3.22** If $A$ is an ordered algebra and $c[v_1, \ldots, v_n]$ is a derived operator of type $s_1 \times \cdots \times s_n \to s'$ then for all $a_i \in A_{s_i}, 1 \leq i \leq n$,

$$c_A(a_1, \ldots, a_n) = h_s c,$$

where $h : OT(\{v_1, \ldots, v_n\}) \to A$ is defined via lemma 2.3.21, by taking $\{v_1, \ldots, v_n\}$ for $X$ and defining $g$ by $g_{s_i} v_i = a_i, 1 \leq i \leq n$.

**Proof.** Simply note that $h$ is a homomorphism from $T(\{v_1, \ldots, v_n\})$ to the algebra $A$ such that $g = h \circ f$. $\square$

**Lemma 2.3.23** If $A$ is an ordered algebra and $c_1[v_1, \ldots, v_n]$ and $c_2[v_1, \ldots, v_n]$ are derived operators of type $s_1 \times \cdots \times s_n \to s'$ such that

$$c_1 \subseteq OT(\{v_1, \ldots, v_n\})_{s'} c_2$$

then

$$c_1A \subseteq (A_{s_1} \times \cdots \times A_{s_n} \to mA_{s'}) c_2A.$$
Proof. Immediate from lemma 2.3.22. □

We now consider the inductive subalgebras of complete ordered algebras that are generated by ordinary subalgebras. This notion is then specialized to reachable inductive subalgebras.

Definition 2.3.24 If $\mathcal{A}$ is a complete ordered algebra and $\mathcal{B}$ is a subalgebra of $\mathcal{A}$ (where $\mathcal{B}$ is an ordinary algebra) then $[\mathcal{B}]$, the subset of $\mathcal{A}$ generated by $\mathcal{B}$, is the least subset of $\mathcal{A}$ such that for all $s \in S$, $B_s \subseteq [\mathcal{B}]_s$ and $\bigcup A' \in [\mathcal{B}]_S$, whenever $A' \subseteq [\mathcal{B}]_S$ is a directed set in $A_S$.

The next lemma shows that $[\mathcal{B}]$ is a subalgebra of $\mathcal{A}$ and thus, since $[\mathcal{B}]$ is closed under $\mathcal{A}$-lub's, that $[\mathcal{B}]$ is an inductive subalgebra of $\mathcal{A}$.

Lemma 2.3.25 If $\mathcal{A}$ is a complete ordered algebra and $\mathcal{B}$ is a subalgebra of $\mathcal{A}$ then $[\mathcal{B}]$ is a subalgebra of $\mathcal{A}$.

Proof. Let $\sigma \in \Sigma$ have type $s_1 \times \cdots \times s_n \rightarrow s'$. We must show that

$$\sigma^{\mathcal{A}}([\mathcal{B}]_{s_1} \times \cdots \times [\mathcal{B}]_{s_n}) \subseteq [\mathcal{B}]_{s'}.$$  

By the definition of subalgebra,

$$\sigma^{\mathcal{A}}(B_{s_1} \times \cdots \times B_{s_n}) \subseteq B_{s'} \subseteq [\mathcal{B}]_{s'}.$$  

If $n = 0$ then

$$\sigma^{\mathcal{A}}([\mathcal{B}]_{s_1} \times \cdots \times [\mathcal{B}]_{s_n}) = \sigma^{\mathcal{A}}(\emptyset) = \sigma^{\mathcal{A}}(B_{s_1} \times \cdots \times B_{s_n});$$

so, assume $n \geq 1$. It is sufficient to show that the following chain of implications holds:

$$\sigma^{\mathcal{A}}(B_{s_1} \times \cdots \times B_{s_n}) \subseteq [\mathcal{B}]_{s'}$$
$$\Rightarrow \sigma^{\mathcal{A}}([\mathcal{B}]_{s_1} \times B_{s_2} \times \cdots \times B_{s_n}) \subseteq [\mathcal{B}]_{s'}$$
$$\Rightarrow \sigma^{\mathcal{A}}([\mathcal{B}]_{s_1} \times [\mathcal{B}]_{s_2} \times B_{s_3} \times \cdots \times B_{s_n}) \subseteq [\mathcal{B}]_{s'}$$
$$\vdots$$
$$\Rightarrow \sigma^{\mathcal{A}}([\mathcal{B}]_{s_1} \times \cdots \times [\mathcal{B}]_{s_n}) \subseteq [\mathcal{B}]_{s'}.$$
We show a representative step

\[\sigma([B]_{s_1} \times \cdots \times [B]_{s_{i-1}} \times B_{s_i} \times B_{s_{i+1}} \times \cdots \times B_{s_n}) \subseteq [B]_{s'}\]

\[\Rightarrow \sigma([B]_{s_1} \times \cdots \times [B]_{s_{i-1}} \times [B]_{s_i} \times B_{s_{i+1}} \times \cdots \times B_{s_n}) \subseteq [B]_{s'},\]

by induction on \([B]_{s_i}\). Let \(C\) be the set of all \(b_i \in [B]_{s_i}\) such that \(\sigma(b_1, \ldots, b_n) \in [B]_{s'}\), for all \(b_1 \in [B]_{s_1}, \ldots, b_{i-1} \in [B]_{s_{i-1}}, b_{i+1} \in B_{s_{i+1}}, \ldots, b_n \in B_{s_n}\). By assumption, \(B_{s_i} \subseteq C\). Let \(C' \subseteq C\) be a directed set in \(A_{s_i}\); we must show that \(\bigcup C' \subseteq C\). Let \(b_1 \in [B]_{s_1}, \ldots, b_{i-1} \in [B]_{s_{i-1}}, b_{i+1} \in B_{s_{i+1}}, \ldots, b_n \in B_{s_n}\). Then,

\[
\sigma(b_1, \ldots, b_{i-1}, \bigcup C', b_{i+1}, \ldots, b_n) = \bigcup \sigma(\{b_1\} \times \cdots \times \{b_{i-1}\} \times C' \times \{b_{i+1}\} \times \cdots \times \{b_n\}) \subseteq [B]_{s'},
\]

since

\[
\sigma(\{b_1\} \times \cdots \times \{b_{i-1}\} \times C' \times \{b_{i+1}\} \times \cdots \times \{b_n\}) \subseteq [B]_{s'}
\]

is a directed set in \(A_{s'}\). \(\Box\)

**Definition 2.3.26** For a complete ordered algebra \(A\) and a subalgebra \(B\) of \(A\), \([B]\), the inductive subalgebra of \(A\) generated by \(B\), is \([B]\), together with the restrictions of the operations and partial orderings of \(A\) to \([B]\).

**Lemma 2.3.27** If \(A\) is a complete ordered algebra and \(B\) is a subalgebra of \(A\) then \([B]\) is the \(\leq\)-least inductive subalgebra of \(A\) that contains \(B\).

**Proof.** If \(C\) is an inductive subalgebra of \(A\) that contains \(B\) then \(C\) is closed under the defining conditions of \([B]\), and so \([B] \subseteq C\). Then, since \([B]\) and \(C\) are both inductive subalgebras of \(A\), it follows that \([B] \leq C\). \(\Box\)

**Definition 2.3.28** For a complete ordered algebra \(A\), define \(R(A)\), the reachable inductive subalgebra of \(A\), to be \([M_A \top]\).

The following lemma is an immediate consequence of lemma 2.3.27.
Lemma 2.3.29 If \( A \) is a complete ordered algebra then \( R(A) \) is the \( \leq \)-least inductive subalgebra of \( A \).

Definition 2.3.30 A complete ordered algebra \( A \) is inductively reachable iff \( A = R(A) \).

It is easy to see that \( R(A) \) itself is inductively reachable (clearly \( R(R(A)) \leq R(A) \), and \( R(A) \leq R(R(A)) \) since \( R(R(A)) \) is an inductive subalgebra of \( A \) and \( R(A) \) is the \( \leq \)-least such inductive subalgebra), and that a complete ordered algebra is inductively reachable iff it has no proper inductive subalgebras. We can carry out proofs by induction over inductively reachable complete ordered algebras \( A \): if \( B \subseteq A_S \) contains \( M_S T_S \), and \( \bigcup B' \in B \), whenever \( B' \subseteq B \) is a directed set, then \( B = A_S \).

Three useful lemmas concerning inductive reachability now follow.

Lemma 2.3.31 There is at most one continuous homomorphism from an inductively reachable complete ordered algebra to a complete ordered algebra.

Proof. Suppose \( f \) and \( g \) are continuous homomorphisms from an inductively reachable complete ordered algebra \( A \) to a complete ordered algebra \( B \), and let \( s \in S \). We prove that \( f_s a = g_s a \), for all \( a \in A_S \), by induction over \( A_S \). Let \( A' = \{ a \in A_S \mid f_s a = g_s a \} \). Firstly, \( M_S T_S \subseteq A' \), since, by the initiality of \( T \),

\[
 f_s(M_{A_S} t) = M_{B_S} t = g_s(M_{A_S} t),
\]

for all \( t \in T_S \). Secondly, if \( A'' \subseteq A' \) is a directed set then

\[
 f_s \bigcup A'' = \bigcup f_s A'' = \bigcup g_s A'' = g_s \bigcup A'',
\]

and thus \( \bigcup A'' \in A' \). \( \square \)

Lemma 2.3.32 If \( A \) and \( B \) are complete ordered algebras, \( A \) is inductively reachable and \( f: A \rightarrow B \) is a continuous homomorphism then \( f \) is also a continuous homomorphism from \( A \) to \( R(B) \).
Proof. It is sufficient to show that \( f_s a \in R(B)_s \), for all \( a \in A_s, s \in S \), and this follows by induction over \( A_s \). □

The next lemma shows that order-isomorphism is respected by inductive reachability.

Lemma 2.3.33 If \( A \) and \( B \) are order-isomorphic complete ordered algebras and, in addition, \( A \) is inductively reachable then \( B \) is also inductively reachable.

Proof. Since \( A \) and \( B \) are order-isomorphic, there is a continuous, surjective order-embedding \( f: A \rightarrow B \). By lemma 2.3.32, it follows that \( fA \subseteq R(B) \). Then, since \( f \) is surjective, it follows that \( B = R(B) \), and thus that \( B = R(B) \). □

We now consider the relationship between substitutive and unary-substitutive inductive pre-orderings over complete ordered algebras. The following two lemmas show that the situation is similar to that for unary-substitutive and substitutive pre-orderings over ordinary algebras: there exist unary-substitutive inductive pre-orderings that are not substitutive, and unary-substitutive pre-orderings over inductively reachable complete ordered algebras are substitutive.

Lemma 2.3.34 There is a signature \( \Sigma \), a complete ordered algebra \( A \) and a unary-substitutive inductive pre-ordering \( \leq \) over \( A \) such that

1. \( \leq \) is not substitutive.

2. The unary-substitutive equivalence relation \( \equiv = \leq \cap \geq \) is not substitutive.

3. There does not exist a congruence \( \equiv' \) over \( A \) such that

\[ M_s t_1 \equiv_s M_s t_2 \text{ if and only if } M_s t_1 \equiv'_s M_s t_2, \]

for all \( t_1, t_2 \in T_s, s \in S \).

Proof. Consider the \( \Sigma, A \) and \( \leq \) from the proof of lemma 2.2.28. Order each \( A_s \) by \( \sqsubseteq_{A_s} \), where \( a_1 \sqsubseteq_{A_s} a_2 \) if \( a_1 = \Omega_s \) or \( a_1 = a_2 \). Then \( A \) is an ordered
algebra and $\subseteq_A \subseteq \leq$. Since $A$ is finite, it then follows that $A$ is complete and $\leq$ is inductive. The rest of the lemma follows by lemma 2.2.28. □

**Lemma 2.3.35** *Unary-substitutive inductive pre-orderings over inductively reachable complete ordered algebras are substitutive.*

**Proof.** Let $\leq$ be a unary-substitutive inductive pre-ordering over an inductively reachable complete ordered algebra $A$. We make use of the characterization of substitutivity given by lemma 2.2.29. It is sufficient to show that for all derived operators $c[v, v_1, \ldots, v_n]$ of type $s \times s_1 \times \cdots \times s_n \rightarrow s'$ and $a, a' \in A_s$, if $a \leq s a'$ then

$$c(a, a_1, \ldots, a_n) \leq_{s'} c(a', a_1, \ldots, a_n), \text{ for all } a_i \in A_{s_i}, 1 \leq i \leq n;$$

we prove this by induction on $n$. The case $n = 0$ follows from the unary-substitutivity of $\leq$. For the induction step, suppose that $c[v, v_1, \ldots, v_{n+1}]$ is a derived operator of type $s \times s_1 \times \cdots \times s_{n+1} \rightarrow s'$ and that $a \leq s a'$. We show by induction over $A_{s_{n+1}}$ that for all $a_{n+1} \in A_{s_{n+1}},$

$$c(a, a_1, \ldots, a_{n+1}) \leq_{s'} c(a', a_1, \ldots, a_{n+1}), \text{ for all } a_i \in A_{s_i}, 1 \leq i \leq n. \quad (2.1)$$

Let $A'$ be the set of all $a_{n+1} \in A_{s_{n+1}}$ such that (2.1). Suppose $t \in T_{s_{n+1}}$; we must show that $M_{s_{n+1}} t \in A'$. Then,

$$(c[v, v_1, \ldots, v_n, t])[v, v_1, \ldots, v_n]$$

is a derived operator of type $s \times s_1 \times \cdots \times s_n \rightarrow s'$, and, by the inductive hypothesis on $n$,

$$c(a, a_1, \ldots, a_n, M_{s_{n+1}} t)$$

$$= (c[v, v_1, \ldots, v_n, t])[a, a_1, \ldots, a_n]$$

$$\leq_{s'} (c[v, v_1, \ldots, v_n, t])[a', a_1, \ldots, a_n]$$

$$= c(a', a_1, \ldots, a_n, M_{s_{n+1}} t),$$
for all \(a_i \in A_{S_i}, 1 \leq i \leq n\). Now, suppose \(A'' \subseteq A'\) is a directed set; we must show that \(\bigsqcup A'' \in A'\). Suppose \(a_i \in A_{S_i}, 1 \leq i \leq n\). Then,

\[
\begin{align*}
    c(a, a_1, \ldots, a_n, \bigsqcup A'') \\
    &= \bigsqcup c(\{a\} \times \{a_1\} \times \cdots \times \{a_n\} \times A'') \\
    \leq_{\delta'} \bigsqcup c(\{a'\} \times \{a_1\} \times \cdots \times \{a_n\} \times A'') \\
    &= c(a', a_1, \ldots, a_n, \bigsqcup A''),
\end{align*}
\]

since \(A\) is complete and \(\leq\) is inductive. \(\square\)

A consequence of lemmas 2.2.26, 2.3.14 and 2.3.35 is that if \(P \subseteq S\), \(A\) is an inductively reachable complete ordered algebra, and \(\leq\) is an inductive pre-ordering over \(A|P\) then \(\leq^C\) is the greatest substitutive inductive pre-ordering over \(A\) whose restriction to \(P\) is included in \(\leq\).

This section concludes with two lemmas concerning the pre-orderings over \(T\) that are induced by inductive pre-orderings over the carriers of complete ordered algebras.

**Lemma 2.3.36** Suppose \(P \subseteq S\), \(A\) is a complete ordered algebra, \(\leq\) is an inductive pre-ordering over \(A|P\), and \(\leq\) is the pre-ordering over \(T|P\) defined by

\[t_1 \leq_P t_2 \iff M_p t_1 \leq_P M_p t_2.\]

Then \(\leq^C\) is a unary-substitutive inductive pre-ordering over \(A\), \(\leq^C\) is an \(\Omega\)-least substitutive pre-ordering over \(T\), and

\[t_1 \leq^C_S t_2 \iff M_s t_1 \leq^C_S M_s t_2,\]

for all \(t_1, t_2 \in T_s, s \in S\).

**Proof.** All that remains after applying lemma 2.2.31 is to show that \(\leq^C\) is inductive and \(\leq^C\) is \(\Omega\)-least. The former fact follows from lemma 2.3.14. For the second, if \(t \in T_s, s \in S\), then

\[M_s \Omega_s = \bot \leq^C_S M_s t,\]
since $\subseteq_A \subseteq \leq^c$, and thus $\Omega_s \leq^c t$. □

**Lemma 2.3.37** Suppose $A$ is a complete ordered algebra, $\leq$ is an inductive pre-ordering over $A$, and $\leq$ is the pre-ordering over $T$ defined by

$$t_1 \leq_s t_2 \text{ iff } M_s t_1 \leq_s M_s t_2.$$

1. If $\leq$ is unary-substitutive then $\leq$ is $\Omega$-least and substitutive.

2. If $\leq$ is substitutive then

$$t_1 \leq_s t_2 \text{ iff } M_s t_1 \leq^c M_s t_2,$$

for all $t_1, t_2 \in T_s$, $s \in S$.

**Proof.** Immediate from lemma 2.3.36, with $P = S$. □

### 2.4 Completion and Quotienting Theorems

In this section, we present a completion theorem and two quotienting theorems for ordered algebras, which will be employed in chapters 5 and 7. The main result is theorem 2.4.2, a completion construction in which ordered algebras are embedded into complete ordered algebras in such a way that certain existing lub's are preserved. Because the operations of complete ordered algebras are required to be continuous, it is impossible, in general, to preserve arbitrary sets of existing lub's. Thus, to begin with, we need a way to specify suitably consistent sets of lub's of ordered algebras. This we do via families of subsets.

**Definition 2.4.1** A family of subsets $\Gamma$ for an ordered algebra $A$ is an $S$-indexed family of sets such that $\Gamma_s$, $s \in S$, is a set of directed subsets of $A_s$; $\{a\} \in \Gamma_s$, for all $a \in A_s$, $s \in S$; and if $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $A'_i \in \Gamma_{s_i}$, $1 \leq i \leq n$, then $\sigma(A'_1 \times \cdots \times A'_n) \in \Gamma_{s'}$. Such an $A$ is $\Gamma$-complete iff $\bigcup A'$ exists,
whenever $A'_i \in \Gamma_S$, $s \in S$, and if $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $A'_i \in \Gamma_{S_i}$, $1 \leq i \leq n$, then

$$\sigma(\bigsqcup A'_1, \ldots, \bigsqcup A'_n) = \bigsqcup \sigma(A'_1 \times \cdots \times A'_n).$$

A homomorphism $f$ from a $\Gamma$-complete ordered algebra $A$ to an ordered algebra $B$ is $\Gamma$-continuous iff $f$ is monotonic and for all $s \in S$ and $A' \in \Gamma_S$, $\bigsqcup f_s A'$ exists and $f_s \bigsqcup A' = \bigsqcup f_s A'$.

In contrast to [CourcelleRaoult] and [ADJ2], we associate families of subsets with individual ordered algebras—i.e., we deal with non-uniform families of subsets. As a consequence, we must explicitly include the singleton directed sets in our families of subsets. See the proof of lemma 2.4.8 to see why this is necessary.

Next, we state our completion theorem, which is a variation of Theorem 1 of [CourcelleRaoult].

**Theorem 2.4.2** If $A$ is a $\Gamma$-complete ordered algebra then there is a complete ordered algebra $C$, together with a $\Gamma$-continuous order-embedding $f: A \rightarrow C$, such that if $D$ is a complete ordered algebra and $g: A \rightarrow D$ is a $\Gamma$-continuous homomorphism then there exists a unique continuous homomorphism $h: C \rightarrow D$ such that $g = h \circ f$:

```
    A       f       C
     \downarrow g          \downarrow h
       |               |
      h               h
       |               |
      D
```

Before giving the proof of theorem 2.4.2, we give a definition and a series of lemmas. Until the end of the proof of the theorem, $A$, $\Gamma$, $D$ and $g$ will be as in
the statement of the theorem. Some of the techniques used in this section are motivated by sections 5 and 6 of [Markowsky].

Definition 2.4.3 A subset $A'$ of $A_S$, $s \in S$, is closed iff the following conditions hold.

1. $\bot_{A_S} \in A'$.
2. If $a \subseteq_{A_S} a'$ and $a' \in A'$ then $a \in A'$.
3. If $B \subseteq A'$ and $B \in \Gamma_s$ then $\bigcup B \in A'$.

For a subset $A'$ of $A_S$, $cl(A')$, the closure of $A'$, is the least closed set containing $A'$.

A set $A'$ is closed iff it is nonempty, downward closed and closed under $\Gamma$-lub’s. Thus, if $A'$ is nonempty then $cl(A')$ is simply the least set containing $A'$ that is downward closed and closed under $\Gamma$-lub’s. Since $cl(A')$ is inductively defined, we can give proofs by induction over it.

Lemma 2.4.4 For all $X, Y \subseteq A_S$, $s \in S$,

1. $X \subseteq cl(X),$
2. $cl(X) = cl(cl(X)),$
3. if $X \subseteq Y$ then $cl(X) \subseteq cl(Y)$.

Proof. Parts 1 and 2 are obvious from the definition. For part 3, suppose $X \subseteq Y$. Then, by part 1, $X \subseteq Y \subseteq cl(Y)$, and so $cl(Y)$ is a closed set containing $X$. Thus, by the leastness of $cl(X)$, $cl(X) \subseteq cl(Y)$. $\square$

Lemma 2.4.5 If $a \in A_S$, $s \in S$, then $cl(\{a\}) = \{a' \in A_S \mid a' \subseteq_{A_S} a\}$.

Proof. Let $A' = \{a' \in A_S \mid a' \subseteq_{A_S} a\}$. Clearly, $A' \subseteq cl(\{a\})$. To show that $cl(\{a\}) \subseteq A'$, it is sufficient to show that $A'$ is closed under $\Gamma$-lub’s, since $A'$ is downward closed and contains $a$. Suppose $B \subseteq A'$ and $B \in \Gamma_s$. Then, $\bigcup B \subseteq_{A_S} a$, since $a$ is an ub of $B$, and thus $\bigcup B \in A'$. $\square$
Lemma 2.4.6 If $A' \in \Gamma_s$, $s \in S$, then $cl(\{\bigcup A'\}) = cl(A')$.

Proof. Firstly, $\{\bigcup A'\} \subseteq cl(A')$, and thus, by lemma 2.4.4, $cl(\{\bigcup A'\}) \subseteq cl(A')$.
Secondly, $A' \subseteq cl(\{\bigcup A'\})$, since $cl(\{\bigcup A'\})$ is downward closed, and thus, by lemma 2.4.4, $cl(A') \subseteq cl(\{\bigcup A'\})$. □

Lemma 2.4.7 If $X$ is a set of subsets of $A_s$, $s \in S$, then $cl(\bigcup X) = cl(\bigcup_{A' \in X} cl(A'))$.

Proof. Firstly, $\bigcup X \subseteq \bigcup_{A' \in X} cl(A')$, and so $cl(\bigcup X) \subseteq cl(\bigcup_{A' \in X} cl(A'))$. Secondly, for all $A' \in X$, $cl(A') \subseteq cl(\bigcup X)$, and thus $cl(\bigcup_{A' \in X} cl(A')) \subseteq cl(\bigcup X)$. □

Lemma 2.4.8 If $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $A_i' \subseteq A_{s_i}$, $1 \leq i \leq n$, are nonempty then

$$cl(\sigma(cl(A'_1) \times \cdots \times cl(A'_n))) = cl(\sigma(A'_1 \times \cdots \times A'_n)).$$

Proof. Showing that the rhs is a subset of the lhs is trivial by lemma 2.4.4. For the other direction, it is sufficient to show that

$$\sigma(cl(A'_1) \times \cdots \times cl(A'_n)) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n)).$$

If $n = 0$ then $\sigma(\langle \rangle) \subseteq cl(\sigma(\langle \rangle))$; so, assume $n \geq 1$. Clearly,

$$\sigma(A'_1 \times \cdots \times A'_n) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n)),$$

and thus it is sufficient to show that the following chain of implications holds:

$$\sigma(A'_1 \times \cdots \times A'_n) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n))$$
$$\Rightarrow \sigma(cl(A'_1) \times A'_2 \times \cdots \times A'_n) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n))$$
$$\Rightarrow \sigma(cl(A'_1) \times cl(A'_2) \times A'_3 \times \cdots \times A'_n) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n))$$
$$\vdots$$
$$\Rightarrow \sigma(cl(A'_1) \times \cdots \times cl(A'_n)) \subseteq cl(\sigma(A'_1 \times \cdots \times A'_n)).$$
We show a representative step

\[ \sigma(\text{cl}(A'_1) \times \cdots \times \text{cl}(A'_{i-1}) \times A'_i \times A'_{i+1} \times \cdots \times A'_n) \subseteq \text{cl}(\sigma(A'_1 \times \cdots \times A'_n)) \]

\[ \downarrow \]

\[ \sigma(\text{cl}(A'_1) \times \cdots \times \text{cl}(A'_{i-1}) \times \text{cl}(A'_i) \times A'_{i+1} \times \cdots \times A'_n) \subseteq \sigma(\text{cl}(A'_1 \times \cdots \times A'_n)), \]

by induction over \( \text{cl}(A'_i) \). Let \( B \) be the set of all \( a_i \in \text{cl}(A'_i) \) such that

\[ \sigma(a_1, \ldots, a_n) \in \text{cl}(\sigma(A'_1 \times \cdots \times A'_n)), \]

for all \( a_1 \in \text{cl}(A'_1), \ldots, a_{i-1} \in \text{cl}(A'_{i-1}), a_{i+1} \in A'_{i+1}, \ldots, a_n \in A'_n \). By assumption, \( A'_i \subseteq B \). Furthermore, \( B \) is downward closed, since \( \sigma \) is monotonic and \( \text{cl}(\sigma(A'_1 \times \cdots \times A'_n)) \) is downward closed. Since \( A'_i \) is nonempty, it only remains to show that \( B \) is closed under \( \Gamma \)-lub's. Suppose \( B' \subseteq B \) and \( B' \in \Gamma_{s_i} \); we must show that \( \bigcup B' \in B \). Let \( a_1 \in \text{cl}(A'_1), \ldots, a_{i-1} \in \text{cl}(A'_{i-1}), a_{i+1} \in A'_{i+1}, \ldots, a_n \in A'_n \). Then,

\[ \sigma(a_1, \ldots, a_{i-1}, \bigcup B', a_{i+1}, \ldots, a_n) \]

\[ = \bigcup \sigma(\{a_1\} \times \cdots \times \{a_{i-1}\} \times B' \times \{a_{i+1}\} \times \cdots \times \{a_n\}) \]

\[ \in \text{cl}(\sigma(A'_1 \times \cdots \times A'_n)), \]

since \( A \) is \( \Gamma \)-complete, and

\[ \sigma(\{a_1\} \times \cdots \times \{a_{i-1}\} \times B' \times \{a_{i+1}\} \times \cdots \times \{a_n\}) \]

is a subset of \( \text{cl}(\sigma(A'_1 \times \cdots \times A'_n)) \) and an element of \( \Gamma_{s_i} \). (Here, it is essential that \( \Gamma \) contain all singleton sets.) \( \square \)

**Lemma 2.4.9** If \( A' \subseteq A_s, s \in S \), then \( g_s A' \) has a lub iff \( g_s \text{cl}(A') \) has a lub, and they are equal if they exist.

**Proof.** It is sufficient to show that for \( d \in D_s \), \( d \) is an ub of \( g_s A' \) iff \( d \) is an ub of \( g_s \text{cl}(A') \). The "if" direction is trivial, since \( A' \subseteq \text{cl}(A') \). For the "only if"
direction, suppose $d$ is an ub of $g_s A'$. Let $A''$ be $\{a'' \in cl(A') \mid g_s a'' \subseteq D_s d\}$. Clearly $A' \subseteq A''$, and $\perp_{A_s} \in A''$, since

$$g_s \perp_{A_s} = g_s \Omega_{sA} = \Omega_{sD} = \perp_{D_s} \subseteq D_s d.$$ 

Suppose $a'' \in A''$ and $a \subseteq_{A_s} a''$, for some $a \in A_s$. Then,

$$g_s a \subseteq_{D_s} g_s a'' \subseteq_{D_s} d,$$

showing that $a \in A''$. Suppose $B \subseteq A''$ and $B \in \Gamma_s$. Then,

$$g_s \bigcup B = \bigcup g_s B \subseteq D_s d,$$

since $g$ is $\Gamma$-continuous, showing that $\bigcup B \in A''$. Thus $A'' = cl(A')$, and so $d$ is an ub of $g_s cl(A')$. \qed

Proof of theorem 2.4.2. We begin by defining a complete ordered algebra $B$, together with a $\Gamma$-continuous order-embedding $f: \mathcal{A} \rightarrow B$. For $s \in S$, $B_s$ is the set of all closed subsets of $A_s$, ordered by the subset relation, and for $\sigma \in \Sigma$ of type $s_1 \times \cdots \times s_n \rightarrow s'$ and $A'_i \in B_{s_i}$, $1 \leq i \leq n$,

$$\sigma_B \langle A'_1, \ldots, A'_n \rangle = cl(\sigma_A(A'_1 \times \cdots \times A'_n)).$$

Then, for $s \in S$,

$$\Omega_{s\mathcal{B}} \langle \rangle = cl(\sigma_A(\langle \rangle)) = cl(\{\perp_{A_s}\}) = \{\perp_{A_s}\}$$

is the least element of $B_s$. The monotonicity of the operations follows from lemma 2.4.4. Thus $B$ is an ordered algebra.

If $B' \subseteq B_s$, $s \in S$, then $cl(\bigcup B')$ is the lub of $B'$, and so $B$ is a cpo (actually, a complete lattice). Suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $B'_i \subseteq B_{s_i}$, $1 \leq i \leq n$, are nonempty. Then,

$$\sigma_B \langle \bigcup B'_1, \ldots, \bigcup B'_n \rangle = \sigma_B \langle cl(\bigcup B'_1), \ldots, cl(\bigcup B'_n) \rangle$$

49
\[ cl(\sigma_A(cl(\bigcup B'_1) \times \cdots \times cl(\bigcup B'_n))) \]
\[ = cl(\sigma_A(cl(\bigcup B'_1 \times \cdots \times \bigcup B'_n))) \quad (\text{lemma 2.4.8}) \]
\[ = cl(\bigcup\{\sigma_A(A'_1 \times \cdots \times A'_n) | A'_i \in B'_i\}) \]
\[ = cl(\bigcup\{ cl(\sigma_A(A'_1 \times \cdots \times A'_n)) | A'_i \in B'_i\}) \quad (\text{lemma 2.4.7}) \]
\[ = \bigcup\{\sigma_B(A'_1, \ldots, A'_n) | A'_i \in B'_i\} \]
\[ = \bigcup\sigma_B(B'_1, \ldots, B'_n), \]

and thus \( \mathcal{B} \) is complete.

Define \( f: A \rightarrow B \) by \( f_s a = cl(\{a\}) \), for \( a \in A_S, s \in S \). Then, \( f \) is a homomorphism from \( A \) to \( B \), since if \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \rightarrow s' \) and \( a_i \in A_{S_i}, 1 \leq i \leq n \), then

\[
\begin{align*}
  f_s \sigma_A(a_1, \ldots, a_n) \\
  = \quad cl(\{\sigma_A(a_1, \ldots, a_n)\}) \\
  = \quad cl(\sigma_A(\{a_1\} \times \cdots \times \{a_n\})) \\
  = \quad cl(\sigma_A(cl(\{a_1\}) \times \cdots \times cl(\{a_n\}))) \quad (\text{lemma 2.4.8}) \\
  = \quad \sigma_B(cl(\{a_1\}), \ldots, cl(\{a_n\})) \\
  = \quad \sigma_B(f_s a_1, \ldots, f_s a_n).
\end{align*}
\]

Furthermore, \( f \) is an order-embedding, since for \( a_1, a_2 \in A_S, s \in S \),

\[
\begin{align*}
a_1 \sqsubseteq_{A_S} a_2 \\
\iff \{ a' \in A_S \mid a' \sqsubseteq_{A_S} a_1 \} \subseteq \{ a' \in A_S \mid a' \sqsubseteq_{A_S} a_2 \} \\
\iff cl(\{a_1\}) \subseteq cl(\{a_2\}) \quad (\text{lemma 2.4.5}) \\
\iff f_s a_1 \sqsubseteq_{B_S} f_s a_2,
\end{align*}
\]

and \( f \) is \( \Gamma \)-continuous, since if \( A' \in \Gamma_S, s \in S \), then

\[ f_s \bigcup A' \]
= \text{cl}((\bigcup A'))
= \text{cl}(A') \quad \text{(lemma 2.4.6)}
= \text{cl}((\bigcup \{a' \mid a' \in A'\}))
= \text{cl}((\bigcup \{\text{cl}(\{a'\}) \mid a' \in A'\})) \quad \text{(lemma 2.4.7)}
= \bigcup \{\text{cl}(\{a'\}) \mid a' \in A'\}
= \bigcup f_s A'.

Unfortunately, \mathcal{B} has lub's of too many sets, and thus we take the \preceq-least inductive subalgebra of \mathcal{B} containing \((f A)\) as our candidate for \mathcal{C}, i.e., we define \mathcal{C} to be \([f A]\). Then \(f\) is also a \Gamma-continuous order-embedding from \(A\) to \(\mathcal{C}\), since \(\mathcal{C}\) is an inductive subalgebra of \(\mathcal{B}\). We can carry out inductions over \(\mathcal{C}\), since for \(s \in S\), if \(C' \subseteq C_s\) contains \(f_s A_s\), and \(\bigcup C'' \in C'\), whenever \(C'' \subseteq C'\) is a directed set, then \(C' = C_s\).

It remains to show the universal property of \((f, \mathcal{C})\). Toward this goal, we first show by induction over \(\mathcal{C}\) that \(\bigcup_{D_s} g_s A'\) exists, for all \(A' \in C_s, s \in S\). Let \(s \in S\) and

\[ C' = \{ A' \in C_s \mid \bigcup g_s A' \text{ exists} \}. \]

Clearly \(f_s A_s \subseteq C'\), since for \(a \in A_s\), \(g_s a\) is the lub of

\[ g_s(f_s a) = g_s(\text{cl}(\{a\})) = g_s\{ a' \in A_s \mid a' \subseteq_{A_s} a \}. \]

Suppose \(C'' \subseteq C'\) is a directed set; we must show that \(\bigcup C'' \in C'\), i.e., that \(g_s \bigcup C''\) has a lub. Since \(C''\) is a directed set, it follows that \(\{ g_s A' \mid A' \in C'' \}\) is directed and so has a lub. Thus, the following subsets of \(D_s\) all share the same lub:

- \(\{ \bigcup g_s A' \mid A' \in C'' \}\),
- \(g_s(\bigcup C'')\),
- \(g_s \text{cl}(\bigcup C'')\) \text{(lemma 2.4.9)},

51
Given this fact, we can define $h: C \to D$ by $h_s A' = \bigcup g_s A'$, for $A' \in C_s$, $s \in S$. Clearly $h$ is monotonic, and $h$ is continuous, since if $C' \subseteq C_s$, $s \in S$, is a directed set then

\[
\begin{align*}
    h_s \bigcup C' & = \bigcup g_s \text{cl}(\bigcup C') \\
    & = \bigcup g_s \text{cl}(\bigcup C') \quad \text{(lemma 2.4.9)} \\
    & = \{ \bigcup g_s A' \mid A' \in C' \} \\
    & = h_s C'.
\end{align*}
\]

Furthermore, $g = h \circ f$, since if $a \in A_s$, $s \in S$, then

\[
h_s(f_s a) = h_s \text{cl}(\{a\}) = \bigcup g_s \text{cl}(\{a\}) = g_s a.
\]

Next, we show that $h$ is a homomorphism from $C$ to $D$. Let $\sigma \in \Sigma$ have type $s_1 \times \cdots \times s_n \to s'$. For $C'_i \subseteq C_{s_i}$, $1 \leq i \leq n$, let $\Phi(C'_1, \ldots, C'_n)$ abbreviate the assertion that for all $c_i \in C'_i$, $1 \leq i \leq n$,

\[
h'_s \sigma_C(c_1, \ldots, c_n) = \sigma_D(h_{s_1} c_1, \ldots, h_{s_n} c_n).
\]

If $a_i \in A_{s_i}$, $1 \leq i \leq n$, then

\[
\begin{align*}
h'_s \sigma_C(f_{s_1} a_1, \ldots, f_{s_n} a_n) & = h'_s \text{cl}(\sigma_A(\text{cl}(\{a_1\}) \times \cdots \times \text{cl}(\{a_n\}))) \\
& = h'_s \text{cl}(\sigma_A(\{a_1\} \times \cdots \times \{a_n\})) \quad \text{(lemma 2.4.8)} \\
& = h'_s(\sigma_A(a_1, \ldots, a_n)) \\
& = g'_s \sigma_A(a_1, \ldots, a_n) \\
& = \sigma_D(g_{s_1} a_1, \ldots, g_{s_n} a_n) \\
& = \sigma_D(h_{s_1}(f_{s_1} a_1), \ldots, h_{s_n}(f_{s_n} a_n)),
\end{align*}
\]

52
showing that $\Phi(f_{S_1} A_{S_1}, \ldots, f_{S_n} A_{S_n})$ holds. If $n = 0$ then $h'_g \sigma_c() = \sigma_D()$; so, assume $n \geq 1$. It is sufficient to show that the following chain of implications holds:

$$
\Phi(f_{S_1} A_{S_1}, \ldots, f_{S_n} A_{S_n}) \\
\Rightarrow \Phi(C_{S_1}, f_{S_2} A_{S_2}, \ldots, f_{S_n} A_{S_n}) \\
\Rightarrow \Phi(C_{S_1}, C_{S_2}, f_{S_3} A_{S_3}, \ldots, f_{S_n} A_{S_n}) \\
\vdots \\
\Rightarrow \Phi(C_{S_1}, \ldots, C_{S_n}).
$$

We show a representative step

$$
\Phi(C_{S_1}, \ldots, C_{S_i-1}, f_{S_i} A_{S_i}, f_{S_{i+1}} A_{S_{i+1}}, \ldots, f_{S_n} A_{S_n}) \\
\Rightarrow \Phi(C_{S_1}, \ldots, C_{S_i-1}, C_{S_i}, f_{S_{i+1}} A_{S_{i+1}}, \ldots, f_{S_n} A_{S_n})
$$

by induction over $C_{S_i}$. Let $C'$ be the set of all $c_i \in C_{S_i}$ such that

$$h'_g \sigma_c(c_1, \ldots, c_n) = \sigma_D(h'_{S_1} c_1, \ldots, h'_{S_n} c_n),$$

for all $c_1 \in C_{S_1}, \ldots, c_{i-1} \in C_{S_{i-1}}, c_{i+1} \in f_{S_{i+1}} A_{S_{i+1}}, \ldots, c_n \in f_{S_n} A_{S_n}$. Then $f_{S_i} A_{S_i} \subseteq C'$, and $C'$ is closed under lub's of directed sets, since $h$ is continuous and $C$ and $D$ are complete. Thus we have shown that $h : C \to D$ is a homomorphism.

Finally, we must show that $h$ is the unique continuous homomorphism from $C$ to $D$ such that $g = h \circ f$. Suppose $h'$ is another such homomorphism, and let $s \in S$. We show by induction over $C_s$ that $h_s c = h'_s c$, for all $c \in C_s$. Let $C'$ be $\{ c \in C_s \mid h_s c = h'_s c \}$. Clearly $f_{S_s} A_{S_s} \subseteq C'$, and if $C'' \subseteq C'$ is a directed set then $\bigcup C'' \in C'$, since

$$h_s \bigcup C'' = \bigcup h'_{S_s} C'' = h'_s \bigcup C'' = h_s \bigcup C''.$$

Thus $h = h'$, as required. This completes the proof of theorem 2.4.2. $\square$

We now introduce some notation that is based upon theorem 2.4.2.
Definition 2.4.10 Let \( A \) be a \( \Gamma \)-complete ordered algebra. We write \( A^\Gamma \) (the \( \Gamma \)-completion of \( A \)) and \( \text{em} \) for the complete ordered algebra \( C \) and the \( \Gamma \)-continuous order-embedding \( f \), respectively, that are given by the proof of theorem 2.4.2. If \( g: A \to D \) is a \( \Gamma \)-continuous homomorphism, for a complete ordered algebra \( D \), then we write \( g^\Gamma \) for the unique continuous homomorphism from \( A^\Gamma \) to \( D \) such that \( g = g^\Gamma \circ \text{em} \).

In the remainder of this section, we present two quotienting constructions: one for ordered algebras and substitutive pre-orderings, and the other for complete ordered algebras and substitutive inductive pre-orderings.

Theorem 2.4.11 [Courcelle and Nivat] Let \( A \) be an ordered algebra and \( \leq \) a substitutive pre-ordering over \( A \) that respects the ordering of \( A \), i.e., \( \leq_A \subseteq \leq \). There is an ordered algebra \( B \), together with a surjective monotonic homomorphism \( f: A \to B \) with the property that \( \leq = \leq_f \), such that if \( C \) is an ordered algebra and \( g: A \to C \) is a monotonic homomorphism with the property that \( \leq \subseteq \leq_g \), then there is a unique monotonic homomorphism \( h: B \to C \) such that \( g = h \circ f \):

\[
\begin{align*}
A & \xrightarrow{f} B \\
\downarrow{g} & \downarrow{h} \\
C &
\end{align*}
\]

Proof. Let \( \equiv \) be the congruence over \( A \) induced by \( \leq \), i.e., \( \equiv = \leq \cap \geq \). We define an ordered algebra \( B \) as follows. For \( s \in S \), \( B_s = A_s/\equiv_s \), and \( \subseteq_{B_s} \) is defined by

\([a_1]\equiv_s \subseteq_{B_s} [a_n]\equiv_s \text{ iff } a_1 \leq_s a_2 \).

If \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \to s' \) then the operation \( \sigma_B \) is defined by

\[ \sigma_B([a_1]\equiv_{s_1}, \ldots, [a_n]\equiv_{s_n}) = [\sigma_A(a_1, \ldots, a_n)]\equiv_{s'} \.]
It is easy to see that \( \subseteq_B \) is well-defined on the equivalence classes and is a partial ordering, that the operations are well-defined and monotonic, and that

\[
\Omega_{SB} = [\Omega_{SA}]_S = [\bot_{AS}]_S = \bot_{BS},
\]

for all \( s \in S \).

Next, we define a surjective homomorphism \( f: A \rightarrow B \) by \( f_S a = [a]_S \). Then \( f \) is monotonic, since \( \subseteq_A \subseteq \leq, \) and \( \leq = \leq_f, \) since

\[
a_1 \leq_S a_2 \text{ iff } [a_1]_S \subseteq_B [a_2]_S \text{ iff } f_S a_1 \subseteq_B f_S a_2,
\]

for all \( a_1, a_2 \in A_S, s \in S \).

It remains to show the universal property of \( (f, B) \). Let \( C \) and \( g \) be as in the statement of the theorem. Define a monotonic homomorphism \( h: B \rightarrow C \) by \( h_S [a]_S = g_S a \). Clearly, \( h \) is well-defined on the equivalence classes and monotonic, since \( \subseteq \subseteq \subseteq_g \). Suppose \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \rightarrow s' \) and \( a_i \in A_{S_i}, 1 \leq i \leq n \). Then,

\[
\begin{align*}
  h_S' \sigma_B ([a_1]_{S_1}, \ldots, [a_n]_{S_n}) & = h_S' [\sigma_A (a_1, \ldots, a_n)]_{S'} \\
  & = g_S' \sigma_A (a_1, \ldots, a_n) \\
  & = \sigma_C (g_{S_1} a_1, \ldots, g_{S_n} a_n) \\
  & = \sigma_C (h_{S_1} [a_1]_{S_1}, \ldots, h_{S_n} [a_n]_{S_n}).
\end{align*}
\]

Thus \( h \) is, indeed, a homomorphism. From the definitions of \( h \) and \( f \), it follows immediately that \( g = h \circ f \). For the uniqueness of \( h \), let \( h': B \rightarrow C \) be a monotonic homomorphism such that \( g = h' \circ f \). Then,

\[
h_S [a]_S = h_S (f_S a) = g_S a = h_S' (f_S a) = h_S' [a]_S;
\]

for all \( a \in A_S, s \in S \), showing that \( h = h' \). \( \Box \)

We now give some notation that is based upon theorem 2.4.11.
Definition 2.4.12 Let $A$ be an ordered algebra and $\leq$ a substitutive pre-ordering over $A$ such that $\sqsubseteq_A \subseteq \leq$. We write $A/\leq$ (the quotient of $A$ by $\leq$) and $qt \leq$ for the ordered algebra $B$ and the surjective monotonic homomorphism $f$, respectively, that are given by the proof of theorem 2.4.11. If $g : A \to C$ is a monotonic homomorphism with the property that $\leq \subseteq \leq_g$ then we write $g/\leq$ for the unique monotonic homomorphism from $A/\leq$ to $C$ such that $g = (g/\leq) \circ qt \leq$.

We often drop the subscript $\leq$ from $qt \leq$ when it is clear from the context. Note that if $\leq$ is an $\Omega$-least substitutive pre-ordering over $T$ then $\sqsubseteq_{\Omega T} = \leq^\Omega \subseteq \leq$, and so $\Omega T/\leq$ is well defined. Clearly, such an $\Omega T/\leq$ is reachable.

We now present two simple corollaries of theorem 2.4.11, followed by the second of our quotienting theorems, theorem 2.4.15.

Corollary 2.4.13 Let $A$ be an ordered algebra and $\leq$ a substitutive pre-ordering over $A$ such that $\sqsubseteq_A \subseteq \leq$. Let $A' \subseteq A_S$ and $a \in A_S$, $s \in S$. Then, $a$ is a lub of $A'$ in $(A_S, \leq_S)$ iff $qt_S a$ is the lub of $qt_S A'$ in $(A/\leq_S)$.

Proof. Follows easily from the surjectivity of $qt$ and the fact that $a_1 \leq_S a_2$ iff $qt_S a_1 \sqsubseteq_S qt_S a_2$, for $a_1, a_2 \in A_S$, $s \in S$. □

Corollary 2.4.14 If $A$ is a reachable ordered algebra then $A$ is order-isomorphic to $\Omega T/\sqsubseteq_A$.

Proof. By theorem 2.4.11, the following diagram commutes:

\[
\begin{array}{ccc}
\Omega T & \xrightarrow{qt} & \Omega T/\sqsubseteq_A \\
\downarrow M_A & & \downarrow M_A/\sqsubseteq_A \\
A & & 
\end{array}
\]
It is sufficient to show that $M_A/\subseteq_A$ is a surjective order-embedding. The surjectivity of $M_A/\subseteq_A$ follows from the surjectivity of $M_A$, and $M_A/\subseteq_A$ is an order-embedding since $\text{qt}$ is surjective and $\subseteq_\alpha = \subseteq_A$. □

Theorem 2.4.15 [Courcelle and Raoult] Let $A$ be a complete ordered algebra and $\leq$ a substitutive inductive pre-ordering over $A$. There is a complete ordered algebra $B$, together with a continuous homomorphism $f: A \to B$ with the property that $\leq = \leq_f$, such that if $C$ is a complete ordered algebra and $g: A \to C$ is a continuous homomorphism with the property that $\leq \subseteq \leq_g$ then there is a unique continuous homomorphism $h: B \to C$ such that $g = h \circ f$:

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow g & & \downarrow h \\
C & & 
\end{array}
\]

Proof. By theorem 2.4.11, we know that $\text{qt}: A \to A/\leq$ is a surjective monotonic homomorphism ($\subseteq_A \subseteq \leq$ since $\leq$ is inductive). Define a family of subsets $\Gamma$ of $A/\leq$ by

$$
\Gamma_s = \{ \text{qt}_s A' \mid A' \subseteq A_s \text{ is a directed set} \}.
$$

If $a \in A_s$, $s \in S$, then

$$
\{ \text{qt}_s a \} = \text{qt}_s \{ a \} \in \Gamma_s.
$$

If $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \to s'$ and $A_i \subseteq A_{s_i}$, $1 \leq i \leq n$, are directed sets then

$$
\sigma((\text{qt}_{s_1} A_1) \times \cdots \times (\text{qt}_{s_n} A_n)) = \text{qt}_{s'} \sigma(A_1 \times \cdots \times A_n) \in \Gamma_{s'}.
$$

Thus $\Gamma$ is well-defined. Next, we show that $A/\leq$ is $\Gamma$-complete. Suppose $A' \subseteq A_s$, $s \in S$, is a directed set; we show that $\text{qt}_s \bigcup A' = \bigcup \text{qt}_s A'$. Clearly $\text{qt}_s \bigcup A'$ is an ub of $\text{qt}_s A'$. Suppose $\text{qt}_s a$ is an ub of $\text{qt}_s A'$. Then $A' \subseteq \alpha a$ and, since
\( \leq \) is inductive, \( \sqcup A' \leq s a \). Thus \( qt_s \sqcup A' \subseteq s qt_s a \), as required. Suppose \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \rightarrow s' \) and \( A_i \subseteq A_{s_i}, \ 1 \leq i \leq n \), are directed sets. Then,

\[
\begin{align*}
\sigma(\sqcup qt_{s_1} A_1, \ldots, \sqcup qt_{s_n} A_n) &= \sigma(qt_{s_1} \sqcup A_1, \ldots, qt_{s_n} \sqcup A_n) \\
&= qt_{s'} \sigma(\sqcup A_1, \ldots, \sqcup A_n) \\
&= qt_{s'} \sqcup \sigma(A_1 \times \cdots \times A_n) \\
&= \sqcup qt_{s'} \sigma(A_1 \times \cdots \times A_n) \\
&= \sqcup \sigma((qt_{s_1} A_1) \times \cdots \times (qt_{s_n} A_n)).
\end{align*}
\]

By theorem 2.4.2, we know that \( \epsilon m: A/\leq \rightarrow (A/\leq)^{r} \) is a \( \Gamma \)-continuous order-embedding into a complete ordered algebra. We take \( (A/\leq)^{r} \) as our candidate for \( B \) and \( \epsilon m \circ qt \) as our candidate for \( f \). Clearly \( f \) is a monotonic homomorphism. For continuity, let \( A' \subseteq A_{s}, s \in S \), be a directed set. Then,

\[
\epsilon m_{s}(qt_{s} \sqcup A') = \epsilon m_{s} \sqcup qt_{s} A' = \sqcup \epsilon m_{s}(qt_{s} A'),
\]

since \( qt_{s} A' \in \Gamma_{s} \). To show that \( \leq = \leq_{f} \), let \( a_1, a_2 \in A_{s}, s \in S \). Then,

\[
a_1 \leq s a_2 \text{ iff } qt_{s} a_1 \subseteq s qt_{s} a_2 \text{ iff } \epsilon m_{s}(qt_{s} a_1) \subseteq s \epsilon m_{s}(qt_{s} a_2),
\]

since \( \epsilon m \) is an order-embedding.

It remains to show the universal property of \((f, B)\). Let \( C \) and \( g \) be as in the statement of the theorem.

\[
\begin{array}{ccc}
A & \xrightarrow{qt} & A/\leq & \xrightarrow{em} & (A/\leq)^{r} \\
\downarrow{g} & & \downarrow{g/\leq} & & \downarrow{(g/\leq)^{r}} \\
C & & & &
\end{array}
\]

By theorem 2.4.11, we know that \( (\dagger) g/\leq \) is the unique monotonic homomorphism from \( A/\leq \) to \( C \) such that \( g = (g/\leq) \circ qt \). To see that \( g/\leq \) is \( \Gamma \)-continuous,
let \( A' \subseteq A_s, s \in S \), be a directed set. Then,

\[
\begin{align*}
(g/\leq)_s \sqcup q_t s \ A' \\
= (g/\leq)_s (q_t s \sqcup A') \\
= g s \sqcup A' \\
= \sqcup g_s A' \\
= \sqcup (g/\leq)_s (q_t s \ A'),
\end{align*}
\]

since \( g \) is continuous. Thus, by theorem 2.4.2, we know that (†) \((g/\leq)^{\Gamma}\) is the unique continuous homomorphism from \((A/\leq)^{\Gamma}\) to \( C \) such that \( g/\leq = (g/\leq)^{\Gamma} \circ em \). We take \((g/\leq)^{\Gamma}\) as our candidate for \( h: B \rightarrow C \). Clearly \( g = h \circ f \), i.e., \( g = (g/\leq)^{\Gamma} \circ em \circ qt \). For uniqueness, suppose \( h': B \rightarrow C \) is a continuous homomorphism such that \( g = h' \circ f \), i.e., \( g = h' \circ em \circ qt \). Then \( g/\leq = h' \circ em \), by (†), and thus \( h' = h \), by (†) and the continuity of \( h' \).

We now give some notation that is based upon theorem 2.4.15.

**Definition 2.4.16** Let \( A \) be a complete ordered algebra and \( \leq \) a substitutive inductive pre-ordering over \( A \). We write \( A//\leq \) (the inductive quotient of \( A \) by \( \leq \)) and \( qt/\leq \) for the complete ordered algebra \( B \) and the continuous homomorphism \( f \), respectively, that are given by the proof of theorem 2.4.15. If \( g: A \rightarrow C \) is a continuous homomorphism with the property that \( \leq \subseteq \leq_s \) then we write \( g//\leq \) for the unique continuous homomorphism from \( A//\leq \) to \( C \) such that \( g = (g//\leq) \circ qt/\leq \).

The section concludes with the lemma that inductive quotients of inductively reachable complete ordered algebras are themselves inductively reachable.

**Lemma 2.4.17** If \( A \) is an inductively reachable complete ordered algebra and \( \leq \) is a substitutive inductive pre-ordering over \( A \) then \( A//\leq \) is also inductively reachable.

**Proof.** By lemma 2.3.33, it is sufficient to show that \( A//\leq \) and \( R(A//\leq) \) are order-isomorphic. Let \( i \) be the inclusion from \( R(A//\leq) \) to \( A//\leq \), so that
i is a continuous homomorphism from $R(A//\leq)$ to $A//\leq$. By lemma 2.3.32, $qt: A \rightarrow A//\leq$ is also a continuous homomorphism from $A$ to $R(A//\leq)$, and if $a_1 \leq a_2$, for $a_1, a_2 \in A_3$, $s \in S$, then

$$qt_3 a_1 \subseteq (A//\leq)_3 qt_3 a_2,$$

and thus

$$qt_3 a_1 \subseteq R(A//\leq)_3 qt_3 a_2.$$

Then, by theorem 2.4.15, we may let $h: A//\leq \rightarrow R(A//\leq)$ be the unique continuous homomorphism such that $qt = h \circ qt$.

By lemma 2.3.31, $h \circ i = id_{R(A//\leq)}$. Also by lemma 2.3.31, it follows that $(i \circ h) \circ qt = qt$, and thus $i \circ h = id_{(A//\leq)}$, since, by theorem 2.4.15, $id_{(A//\leq)}$ is the unique continuous homomorphism over $A//\leq$ such that $qt = id_{(A//\leq)} \circ qt$.

$\square$
Chapter 3

Full Abstraction and Least Fixed Point Models

This chapter is devoted to the definitions and elementary results concerning full abstraction and least fixed point models. This material is based upon the universal algebra of the previous chapter, and the combination of these two chapters forms the foundation upon which the remainder of the thesis is built.

Although we will apply this material to several programming languages in subsequent chapters, it is convenient to have an example programming language available in this chapter, in order to motivate the various definitions and results. For this purpose, we consider an imperative programming language skeleton with null, sequencing, conditional and iteration statements. Formally, consider a signature $\Sigma$ over a single sort, $\star$, that contains the following operators, where $\text{Exp}$ is some unspecified set of boolean expressions:

- $\Omega_{\star}$ and skip of type $\star$;
- while $E$ do— od of type $\star \rightarrow \star$, for all $E \in \text{Exp}$;
- ; and if $E$ then— else— fi of type $\star \times \star \rightarrow \star$, for all $E \in \text{Exp}$.

Since there is only one sort, we drop the sort subscripts from carriers, relations, etc., when considering this language below.
3.1 Full Abstraction

In this section, we formalize what it means for an algebra or ordered algebra to be correct or fully abstract. We actually consider three kinds of correctness and full abstraction: equational, inequational and contextual. The first and third are relations between algebras and congruences over $T$, whereas the second is a relation between ordered algebras and $\Omega$-least substitutive pre-orderings over $T$. As usual, we think of these congruences and pre-orderings over the term algebra as notions of program equivalence and ordering, respectively.

**Definition 3.1.1** Let $\approx$ be a congruence over $T$ and $A$ be an algebra. Then $A$ is $\approx$-equationally correct (or simply $\approx$-correct) if $\equiv_A \subseteq \approx$, and $\approx$-equationally fully abstract (or simply $\approx$-fully abstract) iff $\equiv_A = \approx$.

**Definition 3.1.2** Let $\preceq$ be an $\Omega$-least substitutive pre-ordering over $T$ and $A$ be an ordered algebra. Then $A$ is $\preceq$-inequationally correct iff $\sqsubseteq_A \subseteq \preceq$, and $\preceq$-inequationally fully abstract iff $\sqsubseteq_A = \preceq$.

It is easy to see that equational (respectively, inequational) full abstraction implies equational (respectively, inequational) correctness, but that the converse, in general, fails. Note that if $\preceq$ is an $\Omega$-least substitutive pre-ordering over $T$ and $A$ is a $\preceq$-inequationally fully abstract (respectively, $\preceq$-inequationally correct) ordered algebra then $A$ is $\approx$-fully abstract (respectively, $\approx$-correct), where $\approx$ is the congruence over $T$ induced by $\preceq$: $\approx = \preceq \cap \succeq$.

Suppose that we are given a notion of program equivalence $\approx$ for our example programming language, i.e., a congruence over $T$, with the expected property that

\[
\text{while } E \text{ do } t \text{ od } \approx \text{ if } E \text{ then } t; \text{ while } E \text{ do } t \text{ od else skip } fi,
\]

for all boolean expressions $E \in Exp$ and terms $t \in T$. Then every while-loop will have the same meaning as its expansion in any $\approx$-fully abstract algebra $A$,
and thus for all $E \in \text{Exp}$, the equation

$$
\text{while } E \text{ do } a \text{ od } = \text{if } E \text{ then } a; \text{ while } E \text{ do } a \text{ od } \text{ else skip fi}
$$

holds for all elements of $A$ that are definable by terms. But it is also reasonable to ask that this equation hold for all $a \in A$, i.e., that the unary derived operations

$$
\text{while } E \text{ do } v \text{ od}[v]
$$

and

$$
\text{if } E \text{ then } v; \text{ while } E \text{ do } v \text{ od } \text{ else skip fi}[v]
$$

be equal, for all $E \in \text{Exp}$. This suggests that we consider the following generalization of equational full abstraction from terms to contexts, or, more precisely, to derived operators.

**Definition 3.1.3** Let $\approx$ be a congruence over $\mathcal{T}$ and $A$ be an algebra. Then $A$ is $\approx$-contextually correct iff for all derived operators $c_1[v_1, \ldots, v_n]$ and $c_2[v_1, \ldots, v_n]$ of type $s_1 \times \cdots \times s_n \rightarrow s'$,

if $c_{1A} = c_{2A}$ then for all $t_i \in T_{s_i}, 1 \leq i \leq n$, $c_1(t_1, \ldots, t_n) \approx_{s'} c_2(t_1, \ldots, t_n),$

and $A$ is $\approx$-contextually fully abstract iff for all derived operators $c_1[v_1, \ldots, v_n]$ and $c_2[v_1, \ldots, v_n]$ of type $s_1 \times \cdots \times s_n \rightarrow s'$,

$$
c_{1A} = c_{2A} \text{ iff for all } t_i \in T_{s_i}, 1 \leq i \leq n, c_1(t_1, \ldots, t_n) \approx_{s'} c_2(t_1, \ldots, t_n).
$$

Thus an algebra $A$ is equationally fully abstract with reference to a congruence $\approx$ iff ground equations (equations with no free variables) hold in $A$ exactly when they hold in $\approx$, and contextually fully abstract iff universally quantified equations hold in $A$ exactly when they hold in $\approx$.

Note that we could also define the notions of inequational contextual full abstraction and correctness, in the obvious way.

It is easy to see that contextual full abstraction implies contextual correctness but that the converse, in general, fails. Furthermore, contextual full
abstraction (respectively, contextual correctness) implies full abstraction (re-
respectively, correctness), since for every term $t$ of sort $s$, $t[]$ is a constant derived
operator of type $s$. The next two theorems show that full abstraction does not,
in general, imply contextual full abstraction, but that correctness does imply
contextual correctness.

Theorem 3.1.4 There is a signature $\Sigma$, a congruence $\approx$ over $T$, and a $\approx$-fully
abstract, complete ordered algebra that is not $\approx$-contextually fully abstract.

Proof. Let $\Sigma$ over $S = \{*\}$ have the following operators:

- $\Omega_*$ of type $*$;
- $f$ and $g$ of type $* \to *$.

Since there is only one sort, we drop the sort subscripts from carriers, relations,
etc., below. Let $\approx$ be the greatest congruence over $T$ (all terms are congruent).
Define a complete ordered algebra $A$ as follows. It’s domain $A$ is the two-point
cpo $\{\bot, \top\}$, where $\bot \subseteq T$. It’s operations are defined by:

\[
\begin{align*}
\Omega &= \bot \\
fa &= \bot \\
g_a &= \begin{cases} 
\bot & \text{if } a = \bot \\
\top & \text{if } a = \top 
\end{cases} 
\end{align*}
\]

It is easy to see that $Mt = \bot$, for all $t \in T$, and thus that $A$ is $\approx$-fully abstract.
If $v \in V$, $(f(v))[v]$ and $(g(v))[v]$ are unary derived operators, and

\[(f(v))(t) \approx (g(v))(t),\]

for all $t \in T$, but

\[(f(v))_A = f \neq g = (g(v))_A,\]

showing that $A$ is not contextually fully abstract. $\Box$
Note that the complete ordered algebra $A$ in the previous proof is not inductively reachable. In chapter 5, we will see that inductive reachability is a sufficient condition for full abstraction and contextual full abstraction to coincide.

**Theorem 3.1.5** Let $\approx$ be a congruence over $\mathcal{T}$. An algebra is $\approx$-correct if and only if it is $\approx$-contextually correct.

**Proof.** Let $A$ be an algebra. The "if" direction is trivial. For the "only if" direction, suppose $c_1[v_1,\ldots,v_n]$ and $c_2[v_1,\ldots,v_n]$ are derived operators of type $s_1 \times \cdots \times s_n \rightarrow s'$, and that $c_{1A} = c_{2A}$. Then, for all $t_i \in T_{s_i}$, $1 \leq i \leq n$,

\[
M_{s'} c_1(t_1,\ldots,t_n) = c_1(M_{s_1} t_1,\ldots,M_{s_n} t_n)
= c_2(M_{s_1} t_1,\ldots,M_{s_n} t_n)
= M_{s'} c_2(t_1,\ldots,t_n),
\]

and thus

\[c_1(t_1,\ldots,t_n) \approx_{s'} c_2(t_1,\ldots,t_n),\]

since $A$ is $\approx$-correct. $\square$

Mulmuley has constructed a fully abstract model of the combinatory logic version of PCF in which the standard equational axioms for the $S$ and $K$ combinators are not satisfied. These equations do hold, however, in the notion of program equivalence for PCF, and thus Mulmuley's model is not contextually fully abstract. It would be interesting to find other examples of fully abstract models that fail to be contextually fully abstract.

### 3.2 Least Fixed Point Models

In this section, we say what it means for a complete ordered algebra to be a least fixed point model. This is not an intrinsic property of complete ordered
algebras, but is expressed via the satisfaction of families of least fixed point constraints. We consider two kinds of least fixed point models: ordinary and contextual. The latter is the natural generalization of the former from terms to contexts, or, more precisely, to derived operators. We also consider the satisfaction of families of least fixed point constraints by $\Omega$-least substitutive pre-orderings over the term algebra.

We begin by considering our example imperative programming language again. Conventionally, a model $A$ of this language, i.e., a complete ordered algebra, should assign a while-loop \texttt{while }$E$ \texttt{do }$t$ \texttt{od} the meaning $\bigcup_{n \in \omega} w^n(E,t)$, where $w^n(E,t)$ is the $\omega$-chain in $A$ defined by

$$w^0(E,t) = \bot,$$
$$w^{n+1}(E,t) = \text{if } E \text{ then } (M t); w^n(E,t) \text{ else skip }.$$

This requirement can be expressed syntactically, as follows. Define an $\omega$-chain $W^n(E,t)$ in the ordered term algebra by

$$W^0(E,t) = \Omega,$$
$$W^{n+1}(E,t) = \text{if } E \text{ then } t; W^n(E,t) \text{ else skip },$$

so that $w^n(E,t) = M W^n(E,t)$, for all $n \in \omega$. Then we require that

$$M \text{ while } E \text{ do } t \text{ od } = \bigcup_{n \in \omega} M W^n(E,t).$$

This situation is quite general, and we are led to the following definitions.

\textbf{Definition 3.2.1} A family of least fixed point constraints $\Phi$ is an $S$-indexed family of sets such that for all $s \in S$, $\Phi_s \subseteq T_s \times PT_s$, and for all $(t,T') \in \Phi_s$, $T'$ is a directed set in $OT_s$. We write $t \equiv \bigsqcup T'$ instead of $(t,T')$ for elements of $\Phi_s$.

\textbf{Definition 3.2.2} Let $\Phi$ be a family of least fixed point constraints and $A$ be a complete ordered algebra. Then $A$ is a $\Phi$-least fixed point model (or $A$ satisfies $\Phi$) iff for all $t \equiv \bigsqcup T' \in \Phi_s$, $s \in S$, $M_s t = \bigsqcup M_s T'$. 

66
Note that if \( T' \subseteq OT_s \) is a directed set and \( A \) is an ordered algebra then \( M_s T' \subseteq A_s \) is also a directed set.

The family of least fixed point constraints \( \Phi \) for our example language would be

\[
\{ \text{while } E \text{ do } t \text{ od} \sqcup \bigcup \{ W^n(E, t) \mid n \in \omega \} \mid E \in \text{Exp}, t \in T \}.
\]

Next, we introduce a natural notion of closure, under the operations of the term algebra, for families of least fixed point constraints.

**Definition 3.2.3** A family of least fixed point constraints \( \Phi \) is closed iff for all \( \sigma \in \Sigma \) of type \( s_1 \times \cdots \times s_n \rightarrow s' \), if \( t_i \sqcup \bigcup T_i' \in \Phi s_i, 1 \leq i \leq n \), then

\[
\sigma(t_1, \ldots, t_n) \sqcup \bigcup \sigma(T_1' \times \cdots \times T_n') \in \Phi s'.
\]

We write \( \overline{\Phi} \) for the closure of \( \Phi \), i.e., the least closed family of least fixed point constraints containing \( \Phi \).

Since \( \overline{\Phi} \) is defined inductively, we can give proofs by induction over \( \overline{\Phi} \). The next lemma shows that \( \overline{\Phi} \) has the usual closure properties.

**Lemma 3.2.4**

1. \( \Phi \subseteq \overline{\Phi} \)

2. \( \overline{\overline{\Phi}} = \overline{\Phi} \)

3. if \( \Phi_1 \subseteq \Phi_2 \) then \( \overline{\Phi_1} \subseteq \overline{\Phi_2} \)

**Proof.** Parts 1 and 2 are immediate from the definition. For part 3, suppose \( \Phi_1 \subseteq \Phi_2 \). Then \( \Phi_1 \subseteq \overline{\Phi_2} \), by part 1, and so \( \overline{\Phi_2} \) is a closed family that contains \( \Phi_1 \). But \( \Phi_1 \) is the least such family, and thus \( \overline{\Phi_1} \subseteq \overline{\Phi_2} \). \( \square \)

Three lemmas concerning closed families of least fixed point constraints now follow. The first two concern “singleton” constraints of the form \( t \sqcup \bigcup \{ t \} \), and the third shows that if a complete ordered algebra satisfies a family of least fixed point constraints then it also satisfies the closure of that family of constraints.

**Lemma 3.2.5** If \( \Phi \) is a closed family of least fixed point constraints then \( t \sqcup \bigcup \{ t \} \in \Phi s \), for all \( t \in T_s, s \in S \).
Proof. By structural induction over $T$. Define $T' \subseteq T$ by $T'_s = \{ t \in T_s \mid t \equiv \bigcup \{ t \} \in \Phi_s \}$. Suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$, and $t_i \equiv \bigcup \{ t_i \} \in \Phi_{s_i}$, $1 \leq i \leq n$. Then $t_i \equiv \bigcup \{ t_i \} \in \Phi_{s_i}$, $1 \leq i \leq n$, and, since $\Phi$ is closed,

$$
\sigma(t_1, \ldots, t_n) \equiv \bigcup \{ \sigma(t_1, \ldots, t_n) \}
$$

$$
\sigma(t_1, \ldots, t_n) \equiv \bigcup \{ \sigma(t_1 \times \cdots \times \{ t_n \}) \}
$$

$$
\in \Phi_{s'}.
$$

Thus $\sigma(t_1, \ldots, t_n) \in T'_s$, as required. □

Lemma 3.2.6 The family of least fixed point constraints $\Phi$ defined by

$$
\Phi_s = \{ t \equiv \bigcup \{ t \} \mid t \in T_s \}
$$

is the least closed family of least fixed point constraints, i.e., $\Phi = \overline{\Phi}$.

Proof. By lemma 3.2.5 it is sufficient to show that $\Phi$ is closed. Suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$, and $t_i \equiv \bigcup \{ t_i \} \in \Phi_{s_i}$, $1 \leq i \leq n$. Then,

$$
\sigma(t_1, \ldots, t_n) \equiv \bigcup \sigma(\{ t_1 \} \times \cdots \times \{ t_n \})
$$

$$
= \sigma(t_1, \ldots, t_n) \equiv \bigcup \{ \sigma(t_1, \ldots, t_n) \}
$$

$$
\in \Phi_{s'},
$$

as required. □

Lemma 3.2.7 Let $\Phi$ be a family of least fixed point constraints and $A$ be a complete ordered algebra. If $A$ satisfies $\Phi$ then $A$ satisfies $\overline{\Phi}$.

Proof. By induction over $\Phi$. Define $\Phi' \subseteq \overline{\Phi}$ by

$$
\Phi'_s = \{ t \equiv \bigcup T' \in \overline{\Phi}_s \mid M_s t = \bigcup M_s T' \};
$$

we must show that $\Phi'$ is closed (clearly it contains $\Phi$). Suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$, and $t_i \equiv \bigcup T'_i \in \Phi'_{s_i}$, $1 \leq i \leq n$. Then,

$$
M_{s'} \sigma(t_1, \ldots, t_n).
$$
\[
\begin{align*}
= \sigma(M_{s_1} t_1, \ldots, M_{s_n} t_n) \\
= \sigma(\bigsqcup M_{s_1} T'_1, \ldots, \bigsqcup M_{s_n} T'_n) \\
= \bigsqcup \sigma((M_{s_1} T'_1) \times \cdots \times (M_{s_n} T'_n)) \\
= \bigsqcup M_{s_i} \sigma(T'_1 \times \cdots \times T'_n),
\end{align*}
\]
showing that
\[
\sigma(t_1, \ldots, t_n) \equiv \bigsqcup \sigma(T'_1 \times \cdots \times T'_n) \in \Phi'_s,
\]
and thus that \( \Phi' \) is indeed closed. \( \square \)

Next, we consider the generalization of least fixed point models from terms to contexts, or, more precisely, to derived operators.

**Definition 3.2.8** A family of contextual least fixed point constraints \( \Delta \) is an \( S \)-indexed family of sets such that for all \( s \in S \), \( \Delta_s \) consists of a set of triples
\[
\langle (v_1, \ldots, v_n), c, C' \rangle,
\]
where the \( v_i \in V_{s'_i} \), \( 1 \leq i \leq n \), \( n \geq 0 \), are distinct context variables, \( c \in T(\{v_1, \ldots, v_n\})_s \), and \( C' \subseteq OT(\{v_1, \ldots, v_n\})_s \) is a directed set. We write \( c= v_1, \ldots, v_n \cup C' \) instead of \( \langle (v_1, \ldots, v_n), c, C' \rangle \) for elements of \( \Delta_s \).

Sometimes we write \( c= \bigcup C' \) instead of \( c= v_1, \ldots, v_n \cup C' \), when the variables are clear from the context.

**Definition 3.2.9** Let \( \Delta \) be a family of contextual least fixed point constraints and \( A \) be a complete ordered algebra. Then \( A \) is a \( \Delta \)-contextually least fixed point model (or \( A \) satisfies \( \Delta \)) iff for all \( c= v_1, \ldots, v_n \cup C' \in \Delta_s \), where \( v_i \in V_{s'_i} \), \( 1 \leq i \leq n \), \( c_{s_i}[v_1, \ldots, v_n] \) is the lub of \( \{ c'_{s_i}[v_1, \ldots, v_n] \mid c' \in C' \} \) in \( A_{s'_1} \times \cdots \times A_{s'_n} \rightarrow_c A_s \).

Note that \( \{ c'_{s_i}[v_1, \ldots, v_n] \mid c' \in C' \} \) is a directed set, by lemma 2.3.23.

A suitable family of contextual least fixed point constraints \( \Delta \) for our example imperative language is
\[
\{ \text{while } E \text{ do } v \text{ od} \equiv \bigsqcup \{ W^n(E, v) \mid n \in \omega \} \mid E \in \text{Exp} \},
\]
where \( v \in V \) is an arbitrary context variable, and \( W^n(E, v) \) is the \( \omega \)-chain in \( OT(\{v\}) \) defined by

\[
W^0(E, v) = \Omega, \\
W^{n+1}(E, v) = \text{if } E \text{ then } v; W^n(E, v) \text{ else skip fi}.
\]

Let \( A \) be a complete ordered algebra, and define an \( \omega \)-chain \( w^n(E, a) \) in \( A \), for \( E \in \text{Exp} \) and \( a \in A \), by

\[
w^0(E, a) = \bot, \\
w^{n+1}(E, a) = \text{if } E \text{ then } a; w^n(E, a) \text{ else skip fi},
\]

so that \( w^n(E, a) = W^n(E, v)_A(a) \), for all \( n \in \omega \). Thus for all \( E \in \text{Exp} \),

\[
\text{while } E \text{ do } v \text{ od}_A = \bigsqcup_{n \in \omega} W^n(E, v)_A \\
\text{iff while } E \text{ do } a \text{ od} = \bigsqcup_{n \in \omega} W^n(E, v)(a), \text{ for all } a \in A \\
\text{iff while } E \text{ do } a \text{ od} = \bigsqcup_{n \in \omega} w^n(E, a), \text{ for all } a \in A,
\]

showing that \( A \) is a \( \Delta \)-contextually least fixed point model iff for all \( E \in \text{Exp} \) and \( a \in A \), \( \text{while } E \text{ do } a \text{ od} \) is the lub of the \( \omega \)-chain \( w^n(E, a) \). In contrast, \( A \) satisfies the family of least fixed point constraints \( \Phi \) of our example language iff \( \text{while } E \text{ do } a \text{ od} \) is the lub of \( w^n(E, a) \), for all \( a \in A \) that are definable by terms and \( E \in \text{Exp} \).

Next, we consider the natural family of least fixed point constraints generated by a family of contextual least fixed point constraints.

**Definition 3.2.10** If \( \Delta \) is a family of contextual least fixed point constraints then \( \Delta^* \), the family of least fixed point constraints generated by \( \Delta \), is defined by: \( \Delta^*_g \) is the set of all

\[
c(t_1, \ldots, t_n) = \bigsqcup \{ c'(t_1, \ldots, t_n) \mid c' \in C' \}
\]

such that \( c \equiv v_1, \ldots, v_n \cup C' \in \Delta_s \), \( v_i \in V_{s_i} \), \( 1 \leq i \leq n \), and \( t_i \in T_{s_i} \), \( 1 \leq i \leq n \).
Lemma 2.3.23 shows that $\Delta^*$ is well-defined. It is easy to see that the families $\Phi$ of least fixed point constraints and $\Delta$ of contextual least fixed point constraints that we have defined for our example language are related by $\Phi = \Delta^*$.

Lemma 3.2.11 If $\Delta$ is a family of contextual least fixed point constraints and $A$ is a complete ordered algebra that satisfies $\Delta$ then $A$ also satisfies the family of least fixed point constraints $\Delta^*$.

Proof. Let $c \equiv v_1, \ldots, v_n \cup C' \in \Delta_S, s \in S$, where $v_i \in V_{s'_i}$, $1 \leq i \leq n$, and $t_i \in T_{s'_i}$, $1 \leq i \leq n$. We must show that

$$M_\delta \ c(t_1, \ldots, t_n) = \bigcup M_\delta \{ c'(t_1, \ldots, t_n) \mid c' \in C' \},$$

i.e.,

$$c(M_{s'_1} t_1, \ldots, M_{s'_n} t_n) = \bigcup_{c' \in C'} c'(M_{s'_1} t_1, \ldots, M_{s'_n} t_n),$$

and this follows from the assumption that $A$ satisfies $\Delta$. $\Box$

On the other hand, $A$ may satisfy $\Delta^*$ yet fail to satisfy $\Delta$. We omit the proof, which is similar to that of lemma 3.1.4. In chapter 5 we will see that if $A$ is inductively-reachable and satisfies $\Delta'$ then it also satisfies $\Delta$.

This section concludes with the definition of when an $\Omega$-least substitutive pre-ordering over $T$ satisfies a family of least fixed point constraints. We will use this definition in chapter 5 when we give conditions for the existence of fully abstract, least fixed point models.

Definition 3.2.12 Let $\Phi$ be a family of least fixed point constraints and $\leq$ be an $\Omega$-least substitutive pre-ordering over $T$. Then $\leq$ satisfies $\Phi$ iff for all $t \equiv \bigcup T' \in \Phi_s, s \in S, t$ is a lub of $T'$ in $(T_s, \leq_S)$.

Note that if $T' \subseteq OT_s$ is a directed set and $\leq$ is an $\Omega$-least substitutive pre-ordering over $T$ then $T'$ is a directed set in $(T_s, \leq_S)$, since $OT_s = (T_s, \leq_0)$.
and \( \preceq^\Omega \subseteq \preceq \). The following lemma shows that an \( \Omega \)-least substitutive pre-ordering may satisfy a family of least fixed point constraints without satisfying its closure.

**Lemma 3.2.13** There is a signature \( \Sigma \), an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( \mathcal{T} \), and a family of least fixed point constraints \( \Phi \) such that \( \preceq \) satisfies \( \Phi \) but does not satisfy \( \Phi^\preceq \).

**Proof.** Let \( \Sigma \) over \( S = \{\ast\} \) have the following operators:

- \( \Omega \) and \( x \) of type \( \ast \);
- \( f \) of type \( \ast \to \ast \).

Since there is only one sort, we drop the sort subscripts from relations, etc., below. Define \( \preceq \) over \( \mathcal{T} \) by

\[
\begin{align*}
\vdots \\
f(fx) \\
f \downarrow \\
x \\
\vdots \\
f(f\Omega) \\
f \downarrow \\
\Omega
\end{align*}
\]

and let \( x \equiv \bigcup \{\Omega, f\Omega, f(f\Omega), \ldots\} \) be the only element of \( \Phi \). Clearly \( \preceq \) satisfies \( \Phi \). On the other hand, \( (fx) \equiv \bigcup \{f\Omega, f(f\Omega), \ldots\} \) is an element of \( \Phi^\preceq \), but \( (fx) \) is not a lub of \( \{f\Omega, f(f\Omega), \ldots\} \) in \( \preceq \). \( \Box \)
Chapter 4

Example Correct Models

In this chapter, we study two programming languages within our framework. The first is PCF, and the second is TIE, an imperative language with explicit storage allocation and higher and recursive types. We give denotational semantics for both of these languages, define notions of program ordering and equivalence from these models in a uniform manner, and show that the models are inequationally correct with reference to these notions of program ordering. In contrast, the model of PCF is already known not to be fully abstract and we conjecture that neither is our model of the second language.

A comprehensive treatment of these languages would include the characterization of their notions of program ordering and equivalence in terms of operational semantics. This was done for PCF in [Plotkin1] and [Berry], and appears to be feasible for our second language.

4.1 Defining Notions of Program Ordering

We begin by describing the technique for defining notions of program ordering and equivalence, as abstractions of models, that we use in sections 4.3 and 4.4, and that forms the basis for our positive results of chapter 7. Given a complete ordered algebra $\mathcal{A}$, an $\Omega$-least substitutive pre-ordering over $\mathcal{T}$ is defined as
follows. First, a set of program sorts $P \subseteq S$ is selected, and the terms of sort $p \in P$ are designated as programs. Next, a notion of program behaviour is defined by giving a continuous function $h: A|P \to B$, for a $P$-indexed family of cpo's $B$ of program behaviours, and defining the behaviour of a program $t$ of sort $p$ to be $h_p(M_p t)$. Finally, one term is defined to be less than another iff the behaviour of the first is less than that of the second, in all program contexts. We then take the congruence over $T$ that is induced by this substitutive pre-ordering as our notion of program equivalence, so that two terms are equivalent iff they have the same behaviour in all program contexts.

The following lemma formalizes this technique.

**Lemma 4.1.1** Suppose $A$ is a complete ordered algebra and $h: A|P \to B$ is a continuous function, for $P \subseteq S$ and $B$ a $P$-indexed family of cpo's. Define a pre-ordering $\leq$ over $A|P$ by

$$a_1 \leq_p a_2 \iff h_p a_1 \sqsubseteq h_p a_2,$$

and a pre-ordering $\preceq$ over $T|P$ by

$$t_1 \preceq_p t_2 \iff M_p t_1 \leq_p M_p t_2.$$

Then $\preceq^C$ is an $\Omega$-least substitutive pre-ordering over $T$, $\preceq^C$ is a unary-substitutive inductive pre-ordering over $A$, and

$$t_1 \preceq^C_S t_2 \iff M_S t_1 \leq^C_S M_S t_2,$$

for all $t_1, t_2 \in T_S$, $s \in S$. Furthermore, $A$ is $\preceq^C$-inequationally correct.

**Proof.** Everything except the final claim follows from lemma 2.3.36, since $\preceq = \preceq_h$ is inductive. For the inequational correctness of $A$, simply note that if $M_S t_1 \sqsubseteq A_S M_S t_2$ then $M_S t_1 \leq^C_S M_S t_2$ (as $\preceq^C$ is inductive), and thus $t_1 \preceq^C_S t_2$. □

The unary-substitutive inductive pre-ordering $\preceq^C$ can be seen as the semantic analogue of $\preceq^C$, and its existence forms the basis for the positive results of
chapter 7. Note that if $\approx$ is the equivalence relation over $T \mid P$ that is induced by $\leq$ then $\approx^c$ is the congruence over $T$ induced by $\leq^c$, and thus $\mathcal{A}$ is also $\approx^c$-correct.

4.2 A Metalanguage for Denotational Semantics

In sections 4.3 and 4.4, we make use of a mostly standard metalanguage for defining cpo's and their elements that is taken from [Plotkin3], with minor variations. The following brief description is mainly intended to fix notation.

If $D_1$ and $D_2$ are cpo's then $D_1 \rightarrow D_2$ is the cpo of continuous functions from $D_1$ to $D_2$ (i.e., $D_1 \rightarrow_c D_2$). If $x$ is a variable of type $D_1$ and $E$ is an expression of type $D_2$ then $\lambda x : D_1 . E$ is the usual lambda abstraction of type $D_1 \rightarrow D_2$. If $E_1$ has type $D_1 \rightarrow D_2$ and $E_2$ has type $D_1$ then $E_1 E_2$ is the application of $E_1$ to $E_2$ of type $D_2$. Function space formation associates to the right and function application associates to the left.

If $D_1, \ldots, D_n$, $n \geq 0$, are cpo's then $D_1 \times \cdots \times D_n$ is their product (see definition 2.3.5) and $D_1 + \cdots + D_n$ is their separated sum (the least elements are not identified). We use tupling notation $\langle E_1, \ldots, E_n \rangle$ and the projection functions $\pi_i$ to construct and select, respectively, elements of $D_1 \times \cdots \times D_n$. In addition, for an expression $E$ of type $D_1 \times \cdots \times D_n$, we write

$$\text{let } x_1 : D_1, \ldots, x_n : D_n \text{ be } E \text{ in } E'$$

as an abbreviation for

$$(\lambda x_1 : D_1. \cdots \lambda x_n : D_n . E') (\pi_1 E) \cdots (\pi_n E).$$

As usual, $\mathbf{in}_i : D_i \rightarrow D_1 + \cdots + D_n$ is the $i$'th (nonstrict) injection function, and if $f_i : D_i \rightarrow D'$, $1 \leq i \leq n$, are continuous functions then $[f_1, \ldots, f_n] : D_1 + \cdots + D_n \rightarrow D'$ is the strict continuous function such that
\[ [f_1, \ldots, f_n](\text{in}_i d) = f_i d, \text{ for all } 1 \leq i \leq n. \] For an expression \( E \) of type \( D_1 + \cdots + D_n \) and expressions \( E'_1 \) of type \( D', 1 \leq i \leq n \),

\[
\text{case } E \text{ in } x_1: D_1, E'_1, \ldots, x_n: D_n, E'_n
\]
is an abbreviation for

\[
[\lambda x_1: D_1, E'_1, \ldots, \lambda x_n: D_n, E'_n] E.
\]

We also consider the product \( \Pi_{x \in X} D_x \) of arbitrary \( X \)-indexed families of cpo's \( D_x \) for index sets \( X \). As usual, \( \pi_x : \Pi_{x \in X} D_x \rightarrow D_x \) is the \( x \)'th projection function. We often write \( \rho[x] \) for \( \pi_x \rho \). For \( x \in X \),

\[
[-/x] : (\Pi_{x \in X} D_x) \times D_x \rightarrow (\Pi_{x \in X} D_x)
\]
is defined by

\[
\pi_y \rho[d/x] = \begin{cases} 
  d & \text{if } y = x \\
  (\pi_y \rho) & \text{otherwise}
\end{cases}
\]

For a set \( S \), \( S \bot \) is the flat cpo \( S \cup \{ \bot \} \), for some \( \bot \not\in S \). For an operation \( f: S_1 \times \cdots \times S_n \rightarrow S' \) over sets, we also write \( f \) for the unique extension of \( f \) to \( S_1 \bot \times \cdots \times S_n \bot \rightarrow S'_\bot \) that is strict in each argument, individually. In particular, we make use of the bistrict extensions of addition, \(+: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}\), and the equality operation over the natural numbers, \(=:\mathbb{N} \times \mathbb{N} \rightarrow \mathsf{Tr} \). Define a predecessor function \( \text{pred} : \mathbb{N} \rightarrow \mathbb{N} \) by

\[
\text{pred } x = \begin{cases} 
  x - 1 & \text{if } x \in (\mathbb{N} - \{0\}) \\
  \bot & \text{if } x = \bot \text{ or } x = 0
\end{cases}
\]

We can use the theory developed in [SmythPlotkin] (and also [Plotkin3]) in order to solve recursive domain equations involving \( \rightarrow, \times, + \) and \( \Pi \), up to order-isomorphism.

For any cpo \( D \),

\[
\text{if-then-else} : \mathsf{Tr}_\bot \times D \times D \rightarrow D
\]

76
is the usual conditional function (strict in its first argument), and for cpo's $D_1$ and $D_2$,

$$\text{ifdef } - \text{then } - : D_1 \times D_2 \rightarrow D_2$$

is defined by

$$\text{ifdef } d_1 \text{ then } d_2 = \begin{cases} \bot & \text{if } d_1 = \bot \\ d_2 & \text{otherwise} \end{cases}.$$

Finally, $\mu x: D.E$ is an abbreviation for $(\text{fix } \lambda x: D.E)$, where $\text{fix}: (D \rightarrow D) \rightarrow D$ is the usual least fixed point operation, and $\text{let } x: D \text{ be } E \text{ in } E'$ is an abbreviation for $((\lambda x: D.E') E)$.

4.3 The Programming Language PCF

In this section, we study the programming language PCF within our framework. Of the variants of PCF considered in the literature, we choose that of [Berry], in favour of those of [Plotkin1] and [Milner2]. There are only superficial differences between Berry's and Plotkin's variants, both of which are based upon the typed lambda calculus. Berry's has been studied more systematically and recently than Plotkin's, and has the advantage for us of containing the required undefined constants $\Omega$. Milner's language is significantly different and less natural than both of the others, since it is based upon combinatory logic.

We begin by defining the syntax of PCF, i.e., its signature. The sorts of this signature consist of PCF's types. Let the set of sorts $S$ be least such that

- $\text{nat} \in S$,
- $\text{bool} \in S$,
- $s_1 \rightarrow s_2 \in S$ if $s_1 \in S$ and $s_2 \in S$.

Let $I$ be an $S$-indexed family of disjoint countably-infinite sets of identifiers.

We confuse the family $I$ with the set of all identifiers $\bigcup_{s \in S} I_s$. For an identifier $x \in I$, we write $\text{sort}(x)$ for the unique $s \in S$ such that $x \in I_s$. 

77
Define a signature $\Sigma$ over $S$ with operators

- $\Omega_s$ of type $s$,
- $x$ of type $s$, for $x \in I_s$,
- $\lambda x, s_2$ of type $s_2 \rightarrow (s_1 \rightarrow s_2)$, for $x \in I_{s_1}$,
- $\cdot s_1, s_2$ of type $(s_1 \rightarrow s_2) \times s_1 \rightarrow s_2$,
- $Y_s$ of type $(s \rightarrow s) \rightarrow s$,
- $tt$ and $ff$ of type $bool$,
- $n$ of type $nat$, for $n \in \omega$,
- $succ$ and $pred$ of type $(nat \rightarrow nat)$,
- $zero?$ of type $(nat \rightarrow bool)$,
- $if_{nat}$ of type $(bool \rightarrow nat \rightarrow nat \rightarrow nat)$,
- $if_{bool}$ of type $(bool \rightarrow bool \rightarrow bool \rightarrow bool)$,

for all $s, s_i \in S$, where the compound sorts are parenthesized in order to avoid confusion. Thus $\lambda$ and $Y$ are unary operators, $\cdot$ is binary, and all of the other operators are constants. We drop the sort suffixes from the operators when they are clear from the context, and let $\cdot$ associate to the left.

The operator $\lambda$ binds identifiers in terms, and we have the usual notions of free and bound occurrences of identifiers in terms, and of open and closed terms. We write $[t_1/x]t_2$ for the substitution of $t_1$ for all of the free occurrences of $x$ in $t_2$, where bound variables are renamed, when necessary, to avoid capturing.

Next, we present the natural continuous function model $E$ of PCF, beginning with its semantic domains. The $S$-indexed family of cpo's $Val$ of values is defined
by

\[ Val_{\text{nat}} = N_\perp, \]
\[ Val_{\text{bool}} = Tr_\perp, \]
\[ Val_{s_1 \rightarrow s_2} = Val_{s_1} \rightarrow Val_{s_2}, \]

and the cpo \( Env \) of environments is

\[ \Pi_{x \in I} Val_{\text{sort}(x)}. \]

Now, define a complete ordered algebra \( \mathcal{E} \) as follows. Its carrier \( E \) is defined by

\[ E_s = Env \rightarrow Val_s, \]

for all \( s \in S \). Its operations are defined by:

\[ \Omega_s = \perp_{E_s}, \]
\[ x = \lambda \rho: Env. \rho[x] \]
\[ (x \in I_s), \]
\[ \lambda x, s_2 e = \lambda \rho: Env. \lambda v: Val_{s_1}.(e \rho[v/x]) \]
\[ (x \in I_{s_1}), \]
\[ e_1 \cdot s_1, s_2 e_2 = \lambda \rho: Env.(e_1 \rho)(e_2 \rho), \]
\[ Y_s e = \lambda \rho: Env. \mu v: Val_s.(e \rho)v, \]
\[ tt = \lambda \rho: Env.tt, \]
\[ ff = \lambda \rho: Env.ff, \]
\[ n = \lambda \rho: Env.n \]
\[ (n \in \omega), \]
\[ \text{succ} = \lambda \rho: Env. \lambda n: N_\perp.n+1, \]
\[
pred = \lambda \rho: \text{Env}.\text{pred},
\]
\[
\text{zero?} = \lambda \rho: \text{Env}.\lambda n: N_\bot.n=0,
\]
\[
\text{if nat} = \lambda \rho: \text{Env}.\lambda b: \text{Tr}_\bot.\lambda n_1: N_\bot.\lambda n_2: N_\bot.\text{if } b \text{ then } n_1 \text{ else } n_2,
\]
\[
\text{if bool} = \lambda \rho: \text{Env}.\lambda b: \text{Tr}_\bot.\lambda b_1: \text{Tr}_\bot.\lambda b_2: \text{Tr}_\bot.\text{if } b \text{ then } b_1 \text{ else } b_2.
\]

As is usual for models of languages with block structure, terms are assigned values in \( \mathcal{E} \) with the help of environments. Furthermore, the cpo of values of \( \mathcal{E} \) consists of functions with the pointwise ordering, or, more abstractly, is order-extensional. Thus, in the terminology of [Berry], \( \mathcal{E} \) is an order-extensional model of PCF. It is important to note that complete ordered algebras, in general, will not have any of this additional structure. This is a significant limitation of our theory.

As a guide to the definition of a family of contextual least fixed point constraints for PCF, we now give an equivalent definition of the fixed point operation \( Y \), which does not explicitly involve environments.

**Definition 4.3.1** If \( A \) is an ordered algebra and \( a \in A_s \rightarrow s, s \in S \), then an \( \omega \)-chain \( a^n(\bot) \) in \( A_s \) is defined by

\[
\begin{align*}
    a^0(\bot) &= \bot, \\
    a^{n+1}(\bot) &= a \cdot_s s a^n(\bot).
\end{align*}
\]

**Lemma 4.3.2** An equivalent definition of the operation \( Y_s \) is

\[
Y_s e = \mu e': E_s.(e \cdot_s s e').
\]

**Proof.** A simple induction on \( n \) shows that for all \( n \in \omega, e \in E_s \rightarrow s \) and \( \rho \in \text{Env} \),

\[
(e \rho)^n \bot = e^n(\bot) \rho.
\]

80
For all $e \in E_{S \rightarrow S}$ and $\rho \in Env$,

$$\mu v : Val_{S} (e \rho) v$$

$$= \bigcup_{n \in \omega} (e \rho)^{n} \bot$$

$$= \bigcup_{n \in \omega} (e^{n}(\bot) \rho)$$

$$= (\bigcup_{n \in \omega} e^{n}(\bot)) \rho$$

$$= (\mu e' : E_{S} (e \cdot e')) \rho.$$

Thus for all $e \in E_{S \rightarrow S}$,

$$\lambda \rho : Env, \mu v : Val_{S} (e \rho) v$$

$$= \lambda \rho : Env, (\mu e' : E_{S} (e \cdot e')) \rho$$

$$= \mu e' : E_{S} (e \cdot e'),$$

by $\eta$-conversion. □

Next, define a family of contextual least fixed point constraints $\Delta$ by:

$$\Delta_{S} = \{(Y v) \equiv \bigcup \{ Y^{n} \mid n \in \omega \}) \},$$

for some $v \in V_{S \rightarrow S}$, and where the $\omega$-chain $Y^{n}$ in $OT(\{ v \})_{S}$ is defined by:

$$Y^{0} = \Omega,$$

$$Y^{n+1} = v \cdot Y^{n}.$$

The next lemma shows that a complete ordered algebra satisfies $\Delta$ iff the constant $Y$ is the usual least fixed point operation. An immediate consequence is that $E$ is a $\Delta$-contextually least fixed point model.

**Lemma 4.3.3** A complete ordered algebra $A$ is a $\Delta$-contextually least fixed point model iff for all $a \in A_{S \rightarrow S}$, $s \in S$,

$$Y a = \bigcup_{n \in \omega} a^{n}(\bot).$$

81
Proof. A simple induction on \( n \) shows that for all \( n \in \omega \), \( Y^n(a) = a^n(\bot) \), for all \( a \in AS \rightarrow s \). Thus,

\[
(Yv)_A = \bigsqcup_{n \in \omega} Y^n_A
\]

iff \( (Ya) = \bigsqcup_{n \in \omega} Y^n(a) \), for all \( a \in AS \rightarrow s \)

iff \( (Ya) = \bigsqcup_{n \in \omega} a^n(\bot) \), for all \( a \in AS \rightarrow s \),

as required. \( \Box \)

Note that \( \mathcal{E} \) also satisfies the family of least fixed point constraints \( \Delta^* \), by lemma 3.2.11.

Next, we define notions of program ordering and equivalence for PCF, following [Berry]. We take the closed terms of sort \( \text{nat} \) as programs, \( N_\bot \) as the cpo of program behaviours, and define the behaviour of a program \( t \in T_{\text{nat}} \) to be \( M_{\text{nat}} t \bot \), the result of evaluating \( t \) in the undefined environment. (As \( t \) is closed, any environment would suffice, and we choose \( \bot \) simply for convenience.) Define a pre-ordering \( \preceq_{\text{PCF}} \) over \( T \) by:

\[
t_1 \preceq_{\text{PCF}} t_2 \iff \text{for all derived operators } c[v] \text{ of type } s \rightarrow \text{nat} \text{ such that both } c(t_1) \text{ and } c(t_2) \text{ are closed,}
\]

\[
M_{\text{nat}} c(t_1) \bot \subseteq M_{\text{nat}} c(t_2) \bot.
\]

Let \( \approx_{\text{PCF}} \) be the equivalence relation over \( T \) that is induced by \( \preceq_{\text{PCF}} \). From corollaries 4.1.7 and 4.3.2 of [Berry], it is easy to see that \( \preceq_{\text{PCF}} \) and \( \approx_{\text{PCF}} \) are indeed the notions of program ordering and equivalence, respectively, that are defined in section 2.2 of [Berry].

It is easy to see that the terms 0 and \( (\text{if nat } \cdot tt \cdot 0 \cdot x) \) of sort \( \text{nat} \) are assigned equal meanings in \( \mathcal{E} \), and thus are equivalent under \( \approx_{\text{PCF}} \). This shows that programs can be equivalent to nonprograms, or, in other words, that the property of being a program is not preserved by \( \approx_{\text{PCF}} \), but must be explicitly verified to hold after applying \( \approx_{\text{PCF}} \) transformations to a program. This situation is normal for languages with block structure.
It would perhaps be more natural to also take the closed terms of sort `bool` as programs, as in [Plotkin1]. It is easy to see, however, that the two choices yield equal notions of program ordering and equivalence, and we stick with Berry’s choice, since it is somewhat more convenient to work with.

Unfortunately, lemma 4.1.1, and thus much of the theory developed in chapters 5 and 7, does not directly apply to \( \preceq_{\text{PCF}} \) and \( \simeq_{\text{PCF}} \), since this lemma makes no mention of identifiers and their scopes. It turns out, however, that alternative definitions of these relations via this lemma are possible. Take the set of all terms of sort `nat` as `programs`, \( N_\bot \) as the cpo of `program behaviours`, and let the `behaviour` of a program \( t \in T_{\text{nat}} \) be \( M_{\text{nat}} t \bot \). (Here it is essential that the undefined environment be used.) Define a pre-ordering \( \preceq \) over \( E|\{\text{nat}\} \) by

\[
e_1 \preceq_{\text{nat}} e_2 \text{ iff } e_1 \bot \subseteq e_2 \bot,
\]

and a pre-ordering \( \preceq \) over \( T|\{\text{nat}\} \) by

\[
t_1 \preceq_{\text{nat}} t_2 \text{ iff } M_{\text{nat}} t_1 \preceq_{\text{nat}} M_{\text{nat}} t_2.
\]

Then, by lemma 4.1.1, \( \preceq^\epsilon \) is an \( \Omega \)-least substitutive pre-ordering over \( T \), \( \preceq^\epsilon \) is a unary-substitutive inductive pre-ordering over \( \mathcal{E} \),

\[
t_1 \preceq^\epsilon_s t_2 \text{ iff } M_s t_1 \preceq^\epsilon_s M_s t_2,
\]

for all \( t_1, t_2 \in T_s, s \in S \), and \( \mathcal{E} \) is \( \preceq^\epsilon \)-inequationally correct. Let \( \simeq \) be the equivalence relation over \( T|\{\text{nat}\} \) that is induced by \( \preceq \). Then \( \simeq^\epsilon \) is the congruence over \( T \) induced by \( \preceq^\epsilon \), and \( \mathcal{E} \) is \( \simeq^\epsilon \)-correct.

**Lemma 4.3.4** For every finite set of identifiers \( X \subseteq I \) and sort \( s \in S \), there is a derived operator \( c^X[v] \) of type \( s \rightarrow s \) such that for all \( t \in T_s \), none of the identifiers in \( X \) are free in \( c^X(t) \), all of the free identifiers (if any) of \( c^X(t) \) are also free in \( t \), and

\[
M_s c^X(t) \bot = M_s t \bot.
\]
Proof. By induction on the size of $X$. For the case $|X| = 0$, simply let $c^X[v] = v$. For the induction step, suppose $X = Y \cup \{z\}$, for $z \in I_s'$, and let $c^X[v]$ be

$$(\lambda z, s \cdot c^Y) \cdot s', s \cdot \Omega_{s'}.$$ 

Clearly $c^X$ has the desired identifier closure properties, and for all $t \in T_s$,

$$M_s c^X(t) \perp$$

$$= ((\lambda z (M_s c^Y(t))) \cdot \perp) \perp$$

$$= M_s c^Y(t) \perp[\perp/z]$$

$$= M_s c^Y(t) \perp$$

$$= M_s t \perp,$$

as required. $\square$

Lemma 4.3.5 $\preceq_{\text{PCF}} = \preceq_{\text{C}}$ and $\approx_{\text{PCF}} = \approx_{\text{C}}$

Proof. The latter equality will follow from the former, and clearly $\preceq_{\text{C}} \subseteq \preceq_{\text{PCF}}$.

For the opposite inclusion, suppose that $t_1 \preceq_{\text{PCF}} t_2$, and let $c[v]$ be a derived operator of type $s \to \text{nat}$. By lemma 4.3.4, there is a derived operator $c'[v']$ of type $\text{nat} \to \text{nat}$ such that both $c'(c(t_1))$ and $c'(c(t_2))$ are closed, and

$$M_{\text{nat}} c'(c(t_i)) \perp = M_{\text{nat}} c(t_i) \perp,$$

for $i = 1, 2$. Thus $c'(c)[v]$ is a derived operator of type $s \to \text{nat}$, and

$$M_{\text{nat}} c(t_1) \perp = M_{\text{nat}} (c'(c))(t_1) \perp \subseteq M_{\text{nat}} (c'(c))(t_2) \perp = M_{\text{nat}} c(t_2) \perp,$$

by the assumption that $t_1 \preceq_{\text{PCF}} t_2$. $\square$

Let $\preceq_{\text{PCF}}$ be $\preceq_{\text{C}}$. From all of the above, we can conclude that $\preceq_{\text{PCF}}$ is an $\Omega$-least substitutive pre-ordering over $T$, $\preceq_{\text{PCF}}$ is a unary-substitutive inductive pre-ordering over $\mathcal{E}$,

$$t_1 \preceq_{\text{PCF}} t_2 \text{ iff } M_s t_1 \preceq_{\text{PCF}} M_s t_2,$$
for all \( t_1, t_2 \in T_s, s \in S \), and \( \approx_{PCF} \) is the congruence over \( T \) induced by \( \preceq_{PCF} \).

Furthermore, \( \mathcal{E} \) is \( \preceq_{PCF} \)-inequationally correct, and thus \( \approx_{PCF} \)-correct.

We now recall the theorem of Plotkin that \( \mathcal{E} \) is not \( \approx_{PCF} \)-fully abstract, and thus is not \( \preceq_{PCF} \)-inequationally fully abstract, since the "parallel or" (por) function is not definable in PCF. Let \( binbool \) be the sort \( bool \rightarrow bool \rightarrow bool \).

Define terms \( portest_i \in T_{binbool \rightarrow nat} \), for \( i = 1, 2 \), by

\[
portest_i = \lambda x (\text{if } nat \cdot (x \cdot tt \cdot \Omega_{bool})
\cdot (\text{if } nat \cdot (x \cdot \Omega_{bool} \cdot tt)
\cdot (\text{if } nat \cdot (x \cdot ff \cdot ff)
\cdot \Omega_{nat}
\cdot i)
\cdot \Omega_{nat})
\cdot \Omega_{nat}).
\]

Let \( por \in Val_{binbool} \) be unique such that

\[
\begin{align*}
por tt \perp &= tt, \\
por \perp tt &= tt, \\
por ff ff &= ff,
\end{align*}
\]

and let \( POR \in E_{binbool} \) be \( \lambda p: Env.por \). It is easy to see that for all \( i \in \{1, 2\} \) and \( e \in E_{binbool} \),

\[
(M_{binbool \rightarrow nat portest_i} \cdot e = M_{nat} i \text{ iff } e = POR.
\]

**Theorem 4.3.6** \( \mathcal{E} \) is not \( \approx_{PCF} \)-fully abstract.

**Proof.** The terms \( portest_1 \) and \( portest_2 \) are distinguished by \( \equiv_{\mathcal{E}} \), since they yield different values when applied to \( POR \), but are identified by \( \approx_{PCF} \), as can be seen from the stability theorem (3.6.7) and the context lemma (4.1.5) of [Berry]. \( \square \)
4.4 TIE: A Typed Imperative Programming Language

In this section, we study a programming language called TIE, for Typed Imperative Expressions. TIE is strongly typed, expression-oriented and imperative: every term in the language is an expression of a fixed type, (potentially) yielding a value of that type, but expressions can have side effects, and thus are evaluated in a fixed order. The language has higher and recursive types, as well as reference types and explicit storage allocation. Procedures, i.e., values of higher type, can be returned as the results of other procedures, as well as stored in storage locations of appropriate type. Thus an implementation of TIE cannot follow a simple stack discipline, in the sense of [Halpern], but must retain scopes in a heap. With the exception of not including nondeterminism, our language is thus a good deal more general and uniform than the typed imperative language of [Halpern].

We begin by defining TIE's syntax, i.e., its signature. The sorts of this signature consist of TIE's types. Let $SVar$ be a countably infinite set of sort variables. The set $SExp$ of sort expressions is least such that

- $1 \in SExp$,
- $\nu \in SExp$ if $\nu \in SVar$,
- $\text{ref } s \in SExp$ if $s \in SExp$,
- $s_1 \times s_2, s_1 + s_2, s_1 \rightarrow s_2 \in SExp$ if $s_1, s_2 \in SExp$,
- $\mu \nu. s \in SExp$ if $\nu \in SVar$ and $s \in SExp$.

Here $\mu$ is a variable binding operator, and we have the usual notions of free and bound occurrences of variables in expressions, as well as open and closed expressions. We write $[s_1/\nu]s_2$ for the substitution of $s_1$ for all of the free
occurrences of $\nu$ in $s_2$, where bound variables are renamed, as necessary, to avoid capturing. In the following, we identify sort expressions up to the renaming of bound variables, in the usual way. Thus, e.g., $\mu \nu_1. (1 + \nu_1)$ and $\mu \nu_2. (1 + \nu_2)$ are equal expressions.

The set of sorts $S$ consists of the closed sort expressions, and the set of program sorts $P \subseteq S$ consists of the sorts that do not involve the sort constructors $\text{ref}$ and $\rightarrow$, i.e., the ones built up from $1$, $\times$, $+$ and recursion.

Let $I$ be an $S$-indexed family of disjoint countably infinite sets of identifiers. We confuse the family $I$ with the set of all identifiers $\bigcup_{s \in S} I_s$. For an identifier $x \in I$, we write $\text{sort}(x)$ for the $s \in S$ such that $x \in I_s$.

Define a signature $\Sigma$ over $S$ with the following operators,

- $\Omega_s$ of type $s$,
- $x$ of type $s$, for $x \in I_s$,
- $\ast$ of type $1$,
- $\text{new}_s$ of type $s \rightarrow (\text{ref } s)$,
- $:=_s$ of type $(\text{ref } s) \times s \rightarrow s$,
- $\text{cont}_s$ of type $(\text{ref } s) \rightarrow s$,
- $\equiv_s$ of type $(\text{ref } s) \times (\text{ref } s) \rightarrow (1 + 1)$,
- $\text{pair}_{s_1, s_2}$ of type $s_1 \times s_2 \rightarrow (s_1 \times s_2)$,
- $\text{first}_{s_1, s_2}$ of type $(s_1 \times s_2) \rightarrow s_1$,
- $\text{second}_{s_1, s_2}$ of type $(s_1 \times s_2) \rightarrow s_2$,
- $\text{infirst}_{s_1, s_2}$ of type $s_1 \rightarrow (s_1 + s_2)$,
- $\text{insecond}_{s_1, s_2}$ of type $s_2 \rightarrow (s_1 + s_2)$,
- case—first_x—second_y—esac_{s_3} of type \((s_1 + s_2) \times s_3 \times s_3 \rightarrow s_3\), for \(x \in I_{s_1}\) and \(y \in I_{s_2}\),

- \(\lambda_x, s_2\) of type \(s_2 \rightarrow (s_1 \rightarrow s_2)\), for \(x \in I_{s_1}\),

- \(s_1, s_2\) of type \((s_1 \rightarrow s_2) \times s_1 \rightarrow s_2\),

- \(in_{\mu \nu . s}\) of type \(((\mu \nu . s / \nu) \mu) \rightarrow (\mu \nu . s)\), for \(\mu \nu . s \in S\),

- \(out_{\mu \nu . s}\) of type \((\mu \nu . s) \rightarrow ((\mu \nu . s / \nu) \mu)\), for \(\mu \nu . s \in S\),

- \(s_1, s_2\) of type \(s_1 \times s_2 \rightarrow s_2\),

- \(rec_x\) of type \(s \rightarrow s\), for \(x \in I_s\),

for all \(s, s_i \in S\), where compound sorts are parenthesized in order to avoid confusion. Thus, e.g., \(first\) and \(\lambda\) are unary operators, whereas \(pair\) and \(\cdot\) are binary operators. We drop the sort subscripts from the operators when no confusion can occur, and let \(\cdot\) and \(;\) associate to the left and right, respectively.

The operators \(case, \lambda\) and \(rec\) bind identifiers:

\[
\text{case } t_1 \text{ first}_x t_2 \text{ second}_y t_3 \text{ esac}
\]

binds \(x\) in \(t_2\) and \(y\) in \(t_3\), and \(\lambda_x t\) and \(rec_x t\) bind \(x\) in \(t\). We have the usual notions of bound and free occurrences of identifiers in terms, and of open and closed terms. We write \([t_1 / x] t_2\) for the substitution of \(t_1\) for all of the free occurrences of \(x\) in \(t_2\), where bound variables are renamed, when necessary, to avoid capturing.

The sort \(1\) is intended to contain a single element, \(*\). Elements of reference sorts, \(ref s\), are pointers to storage locations, which are created (and initialized) by \(new\), modified by assignment (:=), and accessed by \(cont\) (contents). The product (\(\times\)), sum (+) and function (\(\rightarrow\)) sorts have their usual meanings and associated operators, where function application (\(\cdot\)) is intended to be by-name, instead of by-value. The sort \(bool = 1 + 1\) can be seen as the booleans; \(\equiv_s\) of type
(ref s) × (ref s) → bool is a test for equality between pointers to storage locations. Recursive sorts are defined via μ, and, e.g., nat = μv.(1 + v) is the natural numbers. The in and out operators are used to package and unpackage elements of recursive sorts. The sequencing operator (;) evaluates its first argument, discards its value (but not its side effects) and yields the result of evaluating its second argument. Finally, the operator rec is used to give recursive definitions in the usual way. For example, rec x x and Ω are intended to be equivalent. With the exception of the the case, λ and · operators, the arguments of operators are evaluated from left to right. Only one of the second and third arguments of the case operator is evaluated, depending upon the value of the first, and neither the only argument of λ nor the second argument of · is ever evaluated (the latter, since application is by-name).

The usual operators over the derived sorts bool and nat can be defined as derived operators in TIE. For example, infirst * and insecond * are the nullary derived operators of type bool that stand for true and false, respectively, and for any s ∈ S, a derived operator if − then − else − fi s of type bool x s x s → s can be defined by

\[(\lambda w(\lambda y case w first z x second z y esac)) \cdot v_1 \cdot v_2 \cdot v_3,\]

for arbitrary identifiers w ∈ Ibool, x, y ∈ Is and z ∈ I1. The case expression must be abstracted and then applied to the context variables in order to prevent the capture of any occurrences of the identifier z in the second and third arguments of the derived conditional. The suitability of this definition is thus dependent upon application being by-name instead of by-value.

Next, we define a model L of TIE, beginning with its semantic domains. The S-indexed family of cpo's Val of values, together with the S-indexed family of order-isomorphisms α, is the initial solution, in the sense of [SmythPlotkin], of the infinite system of simultaneous isomorphism equations

\[\alpha_1: Val_1 \cong \{\ast\}_1,\]
\[
\begin{align*}
\alpha_{\text{ref}\ s} & : \text{Val}_{\text{ref}\ s} \equiv N_\perp, \\
\alpha_{s_1 \times s_2} & : \text{Val}_{s_1 \times s_2} \equiv \text{Val}_{s_1} \times \text{Val}_{s_2}, \\
\alpha_{s_1 + s_2} & : \text{Val}_{s_1 + s_2} \equiv \text{Val}_{s_1} + \text{Val}_{s_2}, \\
\alpha_{s_1 \rightarrow s_2} & : \text{Val}_{s_1 \rightarrow s_2} \equiv \text{Comp}_{s_1 \rightarrow s_2}, \\
\alpha_{\mu \nu \cdot s} & : \text{Val}_{\mu \nu \cdot s} \equiv \text{Val}_{[\mu \nu \cdot s/\nu]s},
\end{align*}
\]

for all \( s, s_i \in S \), where \( \text{Comp}_s \), for computation, is

\[
\text{Sto} \rightarrow (\text{Val}_s \times \text{Sto}),
\]

and \( \text{Sto} \), for store, is

\[
\prod_{s \in S} [(N_\perp \rightarrow \text{Val}_s) \times N_\perp].
\]

Define the cpo \( \text{Env} \) of environments to be

\[
\prod_{x \in I} \text{Comp}_{\text{sort}(x)}.
\]

The names of storage locations are, simply, natural numbers. A store \( \sigma \in \text{Sto} \) consists, for each \( s \in S \), of a pair \( \langle f, n \rangle \), where \( f : N_\perp \rightarrow \text{Val}_s \) and \( n \in N_\perp \). In the semantics given below, we follow the convention that \( n \) is the least available location in \( f \). We write \( \text{empty} \) for the store with no locations allocated:

\[
\text{empty}[s] = (\perp, 0), \text{ for all } s \in S.
\]

Define a complete ordered algebra \( L \) as follows. Its carrier \( L \) is defined by

\[
L_s = \text{Env} \rightarrow \text{Comp}_s = \text{Env} \rightarrow \text{Sto} \rightarrow (\text{Val}_s \times \text{Sto}),
\]

for all \( s \in S \). Thus a term \( t \in T_s \), when evaluated in an environment \( \rho \in \text{Env} \) and a store \( \sigma \in \text{Sto} \), produces a value \( v \in \text{Val}_s \) and a new store \( \sigma' \in \text{Sto} \). Divergence (nontermination) is indicated by \( \sigma' \) being \( \perp \); the value \( v \) is only meaningful when \( \sigma' \neq \perp \).

The operations of \( L \) are now defined below:

\[
\Omega_s = \perp_{L_s},
\]

90
\[ x = \lambda \rho: \text{Env}.\rho[x] \]
\[(x \in I_s),\]

\[ \ast = \lambda \rho: \text{Env}.\lambda \sigma: \text{Sto}.(\alpha^{-1}_{-1} \ast, \sigma),\]

\[ \text{new}_s l = \lambda \rho: \text{Env}.\lambda \sigma: \text{Sto}. \]
\[
\begin{aligned}
&\text{let } v: \text{Val}_s, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
&\text{in } \text{ifdef } \sigma' \\
&\quad \text{then let } f: N_\perp \to \text{Val}_s, n: N_\perp \text{ be } \sigma'[s] \\
&\quad \text{in } (\alpha^{-1}_{\text{ref } s}) n, \\
&\quad \sigma'[\langle \lambda n': N_\perp. \text{if } n' = n \text{ then } v \text{ else } (f n'), \\
&\quad \ n + 1 \rangle[s]), \\
&\quad l_1 := l_2 = \lambda \rho: \text{Env}.\lambda \sigma: \text{Sto}. \\
&\quad \text{let } v_1: \text{Val}_{\text{ref } s}, \sigma': \text{Sto} \text{ be } (l_1 \rho \sigma) \\
&\quad \text{in } \text{let } v_2: \text{Val}_s, \sigma'': \text{Sto} \text{ be } (l_2 \rho \sigma') \\
&\quad \text{in } \text{ifdef } \sigma'' \\
&\quad \quad \text{then let } f: N_\perp \to \text{Val}_s, n: N_\perp \text{ be } \sigma''[s] \\
&\quad \quad \text{in } (v_2, \\
&\quad \quad \quad \sigma''[\langle \lambda n': N_\perp. \text{if } n' = (\alpha_{\text{ref } s} v_1) \text{ then } v_2 \text{ else } (f n'), \\
&\quad \quad \quad \ n \rangle[s]), \\
&\quad \text{cont}_s l = \lambda \rho: \text{Env}.\lambda \sigma: \text{Sto}. \\
&\quad \text{let } v: \text{Val}_{\text{ref } s}, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
&\quad \text{in } \text{let } f: N_\perp \to \text{Val}_s, n: N_\perp \text{ be } \sigma'[s] \\
&\quad \text{in } \langle f (\alpha_{\text{ref } s} v), \sigma'\rangle, \\
&\quad l_1 \equiv_s l_2 = \lambda \rho: \text{Env}.\lambda \sigma: \text{Sto}. \\
&\quad \text{let } v_1: \text{Val}_{\text{ref } s}, \sigma': \text{Sto} \text{ be } (l_1 \rho \sigma) \\
&\quad \text{in } \text{let } v_2: \text{Val}_{\text{ref } s}, \sigma'': \text{Sto} \text{ be } (l_2 \rho \sigma') \\
&\quad \text{in } (\alpha^{-1}_{1+1} (\text{if } (\alpha_{\text{ref } s} v_1) = (\alpha_{\text{ref } s} v_2) 
\]

91
\[
\text{then } \in_1(\alpha_1^{-1} \ast) \\
\text{else } \in_2(\alpha_1^{-1} \ast),
\]
\[
\sigma''),
\]
\[
l_{1 \text{ pair}}_{s_1, s_2} l_2 = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v_1: \text{Val}_{s_1}, \sigma': \text{Sto} \text{ be } (l_1 \rho \sigma) \\
\text{in let } v_2: \text{Val}_{s_2}, \sigma'': \text{Sto} \text{ be } (l_2 \rho \sigma') \\
\text{in } \langle \alpha_3^{-1} x s_2 (v_1, v_2), \sigma'' \rangle,
\]
\[
\text{first}_{s_1, s_2} l = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v: \text{Val}_{s_1} x s_2, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
\text{in } \langle \pi_1(\alpha s_1 x s_2 v), \sigma' \rangle,
\]
\[
\text{second}_{s_1, s_2} l = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v: \text{Val}_{s_1} x s_2, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
\text{in } \langle \pi_2(\alpha s_1 x s_2 v), \sigma'' \rangle,
\]
\[
\text{infirst}_{s_1, s_2} l = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v: \text{Val}_{s_1}, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
\text{in } \langle \alpha_3^{-1} s_1 x s_2 (in_1 v), \sigma'' \rangle,
\]
\[
\text{insecond}_{s_1, s_2} l = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v: \text{Val}_{s_2}, \sigma': \text{Sto} \text{ be } (l \rho \sigma) \\
\text{in } \langle \alpha_3^{-1} s_1 x s_2 (in_2 v), \sigma'' \rangle,
\]
\[
\text{case } l_1 \text{ first}_{x} \text{ l_2 \text{ second}_{y} \text{ l_3 esac}}_{s_3} = \lambda \rho: \text{Env}. \lambda \sigma: \text{Sto}. \\
\text{let } v: \text{Val}_{s_1} + s_2, \sigma': \text{Sto} \text{ be } (l_1 \rho \sigma) \\
\text{in case } (\alpha_3 s_1 + s_2 v) \\
\text{in } v_1: \text{Val}_{s_1}.(l_2 \rho[\lambda \sigma: \text{Sto}.(v_1, \sigma)/x] \sigma'), \\
v_2: \text{Val}_{s_2}.(l_3 \rho[\lambda \sigma: \text{Sto}.(v_2, \sigma)/y] \sigma') \\
(x \in I_{s_1}, y \in I_{s_2}),
\]
\[
92
\]
\[ \lambda x, s_2 l = \lambda \rho: \text{Env.} \lambda \sigma: \text{Sto.} \]
\[ (\alpha_{s_1}^{-1} \rightarrow s_2 (\lambda \kappa: \text{Comp}_{s_1}.(l \rho[\kappa/x])), \sigma) \]
\[ (x \in I_{s_1}), \]

\[ l_1 \cdot s_1, s_2 l_1 = \lambda \rho: \text{Env.} \lambda \sigma: \text{Sto.} \]
\[ \text{let } v: \text{Val}_{s_1} \rightarrow s_2, \sigma': \text{Sto be } (l_1 \rho \sigma) \]
\[ \text{in } (\alpha_{s_1} \rightarrow s_2 v (l_2 \rho) \sigma'), \]

\[ \text{in}_{\mu \nu . s} l = \lambda \rho: \text{Env.} \lambda \sigma: \text{Sto.} \]
\[ \text{let } v: \text{Val}_{[\mu \nu . s / \nu]s}, \sigma': \text{Sto be } (l \rho \sigma) \]
\[ \text{in } (\alpha_{\mu \nu . s} v, \sigma'), \]

\[ \text{out}_{\mu \nu . s} l = \lambda \rho: \text{Env.} \lambda \sigma: \text{Sto.} \]
\[ \text{let } v: \text{Val}_{\mu \nu . s}, \sigma': \text{Sto be } (l \rho \sigma) \]
\[ \text{in } (\alpha_{\mu \nu . s} v, \sigma'), \]

\[ l_1 ; s_1, s_2 l_2 = \lambda \rho: \text{Env.} \lambda \sigma: \text{Sto.} \]
\[ \text{let } v: \text{Val}_{s_1}, \sigma': \text{Sto be } (l_1 \rho \sigma) \]
\[ \text{in } (l_2 \rho \sigma'), \]

\[ \text{rec}_x l = \mu l': L_S. \lambda \rho: \text{Env.}(l \rho[(l' \rho)/x]) \]
\[ (x \in I_S). \]

Note that for all \( x \in I_S, s \in S \), the elements \( \Omega_s \) and \( \text{rec}_x x \) of \( L_S \) are equal.

The obvious principle of extensionality under application is not valid in \( L \), as the following example shows. Let \( x \in I_S, s \in S \), and consider the elements \( \lambda x, s \Omega_S \) and \( \Omega_S \rightarrow s \) of \( L_S \rightarrow s \). They are unequal, since for any \( \rho \in \text{Env} \) and \( \sigma \in \text{Sto} \), \( \pi_2((\lambda x \Omega) \rho \sigma) = \sigma \). One the other hand, for any \( l \in L_S, \rho \in \text{Env} \) and \( \sigma \in \text{Sto} \),

\[ ((\lambda x \Omega) \cdot l) \rho \sigma \]
and thus, for all $l \in L_s$, $(\lambda x \Omega) \cdot l = \Omega \cdot l$, by extensionality in the metalanguage. I view the lack of extensionality as an expected property of models of TIE, rather than as a defect of $\mathcal{L}$.

Since application is by-name instead of by-value, we can give an equivalent definition of the operation $\text{rec}$ that does not explicitly mention environments.

**Lemma 4.4.1** An equivalent definition of the operation $\text{rec}$ is

$$\text{rec}_x l = \mu l': L_s.(\lambda x, s l) \cdot s, s l'$$

$(x \in I_s)$.

**Proof.** For $l, l' \in L_s$, $\rho \in \text{Env}$ and $\sigma \in \text{Sto}$,

$$((\lambda x l) \cdot l') \rho \sigma$$

$$= \alpha_s \rightarrow_s (\alpha_s^{-1} \rightarrow_s \lambda \kappa: \text{Comp}_s.(l \rho[\kappa/x])) \cdot (l' \rho) \sigma$$

$$= l \rho[(l' \rho)/x] \rho \sigma.$$ 

Thus for all $l, l' \in L_s$,

$$((\lambda x l) \cdot l')$$

$$= \lambda \rho: \text{Env}.((\lambda x l) \cdot l') \rho$$

$$= \lambda \rho: \text{Env}.(l \rho[(l' \rho)/x]),$$

by extensionality and $\eta$-conversion. $\square$

As a consequence of this lemma, it is appropriate to define a family of contextual least fixed point constraints $\Delta$ for TIE by:

$$\Delta_s = \{ \text{rec}_x v \equiv \bigcup \{ \text{rec}_x^n | n \in \omega \} | x \in I_s \},$$
for some $v \in V_s$, and where the $\omega$-chain $\text{rec}_x^n$ in $OT\{\{v\}\}_s$ is defined by:

$$\text{rec}_x^0 = \Omega,$$
$$\text{rec}_x^{n+1} = (\lambda x \, v) \cdot \text{rec}_x^n.$$

The next lemma shows that a complete ordered algebra is a $\Delta$-contextually least fixed point model iff $\text{rec}$ is the expected least fixed point operation. An immediate consequence is that $L$ is a $\Delta$-contextually least fixed point model.

**Lemma 4.4.2** A complete ordered algebra $A$ is a $\Delta$-contextually least fixed point model iff for all $x \in I_s$, $s \in S$, 

$$\text{rec}_x a = \bigsqcup_{n \in \omega} r_x^n(a), \text{ for all } a \in A_s,$$

where the $\omega$-chain $r_x^n(a)$ in $A_s$ is defined by:

$$r_x^0(a) = \bot,$$
$$r_x^{n+1}(a) = (\lambda x \, a) \cdot r_x^n(a).$$

**Proof.** A simple induction over $n$ shows that for all $n \in \omega$, 

$$\text{rec}_x^n(a) = r_x^n(a),$$

for all $a \in A_s$. Thus,

$$(\text{rec}_x v)_A = \bigsqcup_{n \in \omega} \text{rec}_x^n,$$

iff 

$$(\text{rec}_x a) = \bigsqcup_{n \in \omega} \text{rec}_x^n(a), \text{ for all } a \in A_s$$

iff 

$$(\text{rec}_x a) = \bigsqcup_{n \in \omega} r_x^n(a), \text{ for all } a \in A_s,$$

as required. ∎

Now, we define notions of program ordering and equivalence for TIE. It is natural to take the closed terms of program sort as programs, $Val | P$ as the cpo of program behaviours, and to define the behaviour of a program to be the result of evaluating it in the undefined environment and empty store. More precisely, define a continuous function $h: L | P \to Val | P$ by:
\( h_p \cdot l = \text{let } v: \text{Val}_p, \sigma: \text{Sto be} (l \perp \text{empty}) \text{ in } \text{ifdef } \sigma \text{ then } v. \)

The behaviour of a program \( t \in T_p \) is then \( h_p(M_p t) \). Define a pre-ordering \( \leq^{\text{TIE}} \) over \( T \) by: \( t_1 \leq^{\text{TIE}}_s t_2 \) iff for all derived operators \( c[v] \) of type \( s \rightarrow p, p \in P \), such that both \( c(t_1) \) and \( c(t_2) \) are closed,

\[
h_p(M_p c(t_1)) \sqsubseteq_{\text{val}_p} h_p(M_p c(t_2)).
\]

Let \( \approx^{\text{TIE}} \) be the equivalence relation over \( T \) that is induced by \( \leq^{\text{TIE}} \).

Unfortunately, lemma 4.1.1, and thus much of the theory developed in chapters 5 and 7, does not directly apply to \( \leq^{\text{TIE}} \) and \( \approx^{\text{TIE}} \), since this lemma makes no mention of identifiers and their scopes. It turns out, however, that alternative definitions of these relations via this lemma are possible. Take the set of all terms of program sort as programs, \( \text{Val | P} \) (again) as the cpo of program behaviours, and define the behaviour of a program \( t \in T_p \) to be \( h_p(M_p t) \), for the function \( h \) defined above. Define a pre-ordering \( \leq \) over \( L | P \) by

\[
l_1 \leq_p l_2 \text{ iff } h_p l_1 \sqsubseteq_{\text{val}_p} h_p l_2,
\]

and a pre-ordering \( \leq \) over \( T | P \) by

\[
t_1 \leq_p t_2 \text{ iff } M_p t_1 \leq_p M_p t_2.
\]

Then, by lemma 4.1.1, \( \leq^c \) is an \( \Omega \)-least substitutive pre-ordering over \( T \), \( \leq^c \) is a unary-substitutive inductive pre-ordering over \( L \),

\[
t_1 \leq^c_s t_2 \text{ iff } M_s t_1 \leq^c_s M_s t_2,
\]

for all \( t_1, t_2 \in T_s, s \in S \), and \( L \) is \( \leq^c \)-inequationally correct. Let \( \approx \) be the equivalence relation over \( T | P \) that is induced by \( \leq \). Then \( \approx^c \) is the congruence over \( T \) that is induced by \( \leq^c \), and \( L \) is \( \approx^c \)-correct.

**Lemma 4.4.3** For every finite set of identifiers \( X \subseteq I \) and sort \( s \in S \), there is a derived operator \( c^X[v] \) of type \( s \rightarrow s \) such that for all \( t \in T_s \), none of the
identifiers in X are free in c^X(t), all of the free identifiers (if any) of c^X(t) are also free in t, and

\[ M_s c^X(t) \perp = M_s t \perp. \]

**Proof.** By induction on the size of X. For the case |X| = 0, simply let c^X[v] = v. For the induction step, suppose X = Y \cup \{z\}, for z \in I_{s'}, and let c^X[v] be

\[(\lambda z, s c^Y) \cdot s', s \Omega_{s'}.\]

Let t \in T_s. Clearly c^X has the desired identifier closure properties, and for all \( \sigma \in St_{\sigma}, \)

\[
M_s c^X(t) \perp \sigma \\
= ((\lambda z (M_s c^Y(t))) \cdot \perp) \perp \sigma \\
= \alpha_{s'} \to s (\alpha^{-1}_{s'} \to s (\lambda x: Comp_{s'} . M_s c^Y(t) \perp [k/z])) \perp \sigma \\
= M_s c^Y(t) \perp [\perp / z] \sigma \\
= M_s c^Y(t) \perp \sigma \\
= M_s t \perp \sigma.
\]

The lemma then follows by extensionality. \( \square \)

**Lemma 4.4.4** \( \preceq_{\text{TIE}} = \preceq_c \) and \( \approx_{\text{TIE}} = \approx_c \)

**Proof.** The latter equality will follow from the former, and clearly \( \preceq_c \subseteq \preceq_{\text{TIE}}. \)

For the opposite inclusion, suppose that \( t_1 \preceq_{\text{TIE}} t_2, \) and let \( c[v] \) be a derived operator of type \( s \to p, p \in P. \) By lemma 4.4.3, there is a derived operator \( c'[v'] \) of type \( p \to p \) such that both \( c'(c(t_1)) \) and \( c'(c(t_2)) \) are closed, and

\[ M_p c'(c(t_i)) \perp = M_p c(t_i) \perp, \]

for \( i = 1, 2. \) Thus \( c'(c)[v] \) is a derived operator of type \( s \to p, \) and

\[ h_p(M_p c(t_1)) = h_p(M_p (c'(c))(t_1)) \subseteq h_p(M_p (c'(c))(t_2)) = h_p(M_p c(t_2)), \]

97
by the assumption that $t_1 \preceq^\text{TIE} t_2$. □

Let $\preceq^\text{TIE}$ be $\preceq^C$. From all of the above, we can conclude that $\preceq^\text{TIE}$ is an \(\Omega\)-least substitutive pre-ordering over \(T\), $\preceq^\text{TIE}$ is a unary-substitutive inductive pre-ordering over \(L\),

$$t_1 \preceq^\text{TIE} t_2 \iff M_s t_1 \preceq^\text{TIE} M_s t_2,$$

for all $t_1, t_2 \in T_s$, $s \in S$, and $\approx^\text{TIE}$ is the congruence over \(T\) induced by $\preceq^\text{TIE}$. Furthermore, \(L\) is $\preceq^\text{TIE}$-inequationally correct, and thus $\approx^\text{TIE}$-correct. On the other hand, I conjecture that \(L\) is not $\approx^\text{TIE}$-fully abstract (and thus not $\preceq^\text{TIE}$-inequationally fully abstract) since

$$M_1 (\text{new } *); * \neq M_1 *,$$

but it appears that

$$(\text{new } *); * \approx^\text{TIE} _1 *.$$

In the remainder of this section, we investigate a call-by-value version of TIE. First, the isomorphism equations for $\rightarrow$ that are used in the definition of \(Val\) should be changed to

$$\alpha_{s_1} \rightarrow s_2: Val_{s_1} \rightarrow s_2 \cong Val_{s_1} \rightarrow Comp_{s_2},$$

for all $s_1, s_2 \in S$. Second, the definitions of the operations $\lambda$ and $\cdot$ should be changed to

$$\lambda x, s_2 l = \lambda \rho: Env. \lambda \sigma: Sto.$$

$$(\alpha_{s_1}^{-1} \rightarrow s_2 (\lambda v: Val_{s_1}.(l \rho[\lambda \sigma: Sto.(v, \sigma)/x]))),$$

$$\sigma)$$

$$(x \in I_{s_1}),$$

and

$$l_1 \cdot_{s_1, s_2} l_2 = \lambda \rho: Env. \lambda \sigma: Sto.$$
Now both arguments of \( \cdot \) are evaluated, the first followed by the second.

Unfortunately, the change from call-by-name to call-by-value has at least three unpleasant consequences. The first is that the derived conditional operator (given immediately after the definition of TIE’s signature) is no longer suitable, and I conjecture that no replacement exists.

The second is that we lose lemma 4.4.1, and thus the family of contextual least fixed point constraints \( \Delta \) is not appropriate for the changed language; again, there does not appear to be a suitable replacement. As a partial solution to this problem, we might consider making do with a family of (ordinary) least fixed point constraints. Unfortunately, we run into problems again, since the following “definition” of a family of least fixed point constraints \( \Phi \) is invalid:

\[
\Phi_s = \{ \text{rec}_x t = \cup \{ \text{rec}_x^n t \mid n \in \omega \} \mid x \in I_s, t \in T_s \},
\]

where \( \text{rec}_x^n t \) is defined by

\[
\text{rec}_x^0 t = \Omega,
\]

\[
\text{rec}_x^{n+1} t = [\text{rec}_x^n t/x]t.
\]

Due to the renaming of identifiers that is involved in substitution, \( \text{rec}_x^n t \) is not always an \( \omega \)-chain in \( OT_s \).

The third problem is that we lose the proofs of lemmas 4.4.3 and 4.4.4, and the relations \( \approx^{\text{TIE}} \) and \( \approx^c \) appear, in fact, to be unequal, as the following example suggests. Consider the terms \( * \) and \( x; * \) of sort \( 1 \), for \( x \in I_{\mu\nu\mu \nu} \). It is easy to see that they are distinguished by \( \approx^c_1 \), since

\[
h_1(M_1(x; *)) = \bot \neq (\alpha_1^{-1} * ) = h_1(M_1 * ).
\]
On the other hand, I see no way of causing \( x; * \) to diverge in a closed context, without also causing * to diverge in that context, and thus it seems that they are equivalent under \( \approx_{\text{TIE}} \). (Here, it is essential that there does not exist a closed, convergent, term \( t \) of sort \( \mu \nu \nu \); if there did exist such a \( t \) then our pair of terms would be distinguished by the context \( c[v] = \text{rec}_x (v; t) \).)

I hope that at least some of these problems can be solved by giving identifiers and their scopes formal significance in signatures, and working with models that have environments as part of their formal structure. In such a theory, terms would be identified up to the renaming of bound variables, solving the problem with \( \Phi \), and a version of lemma 4.1.1 that is directly applicable to notions of program ordering such as \( \preceq_{\text{TIE}} \) could be developed. Perhaps the problem of defining derived operators can be solved by working with two kinds of derived operators: ones that can capture free identifiers and ones that cannot. The problem of specifying contextually least fixed point properties of models is more difficult, and will require more changes, but I hope that this proposal is a step in the right direction.
Chapter 5

Conditions for the Existence of Fully Abstract Models

In this chapter, we give necessary and sufficient conditions for the existence of correct and fully abstract, least fixed point, complete ordered algebras. As usual, we consider the three kinds of correctness and full abstraction, equational (ordinary), inequational and contextual, and the two kinds of least fixed point models, ordinary and contextual. The condition for the existence of inequationally fully abstract, (ordinarily) least fixed point, complete ordered algebras is the cornerstone of these results: it is developed first, using a general term model construction, and the other conditions are derived from it. The condition for the existence of equationally fully abstract, least fixed point models will be employed in chapter 6 to show that such models do not exist for two natural nondeterministic programming languages. The condition for the existence of inequationally fully abstract, least fixed point models will be used in chapter 7 to develop a useful model-theoretic condition, which is used to show the existence of inequationally fully abstract models for the languages introduced in chapter 4.

We also prove theorems concerning the existence of initial objects and the nonexistence of terminal objects in various categories of correct and fully ab-
stract, least fixed point, complete ordered algebras, and show the existence of nonisomorphic inductively reachable, inequationally fully abstract, least fixed point, complete ordered algebras.

As an aid to understanding and appreciating these results, we begin by considering the simpler case of inequationally correct and fully abstract ordered algebras. In the following, let $\leq$ be an $\omega$-least substitutive pre-ordering over $T$. Clearly $OT$ is initial in the category of $\leq$-inequationally correct ordered algebras, together with monotonic homomorphisms, and, by theorem 2.4.11, $OT/\leq$ is initial in the full subcategory of $\leq$-inequationally fully abstract ordered algebras. By corollary 2.4.14, every reachable ordered algebra $A$ is order-isomorphic to $OT/\leq_A$. Thus $OT/\leq$ is the unique (up to order-isomorphism) reachable, $\leq$-inequationally fully abstract, ordered algebra and, again by theorem 2.4.11, it is terminal in the category of reachable, $\leq$-inequationally correct, ordered algebras, together with monotonic morphisms.

As we will see in the following sections, the situation is considerably more complicated for least fixed point, complete ordered algebras and continuous homomorphisms.

5.1 Inequational Full Abstraction

In this section, we give a necessary and sufficient condition for the existence of $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebras, and show that if the category of such ordered algebras and continuous homomorphisms is nonempty that it has an initial object.

Theorem 5.1.4 is the main result: a $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra exists iff $\leq$ satisfies $\Phi$. The "only if" direction of this theorem is straightforward. For the "if" direction, we construct a $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra $I(\leq, \Phi)$ via the quotenting and completion constructions of section 2.4. The
ordered algebra $OT/\preceq$ is $\preceq$-inequationally fully abstract and satisfies the constraints of $\Phi$ but is not, in general, complete. Thus, we embed $OT/\preceq$ into a complete ordered algebra in such a way that exactly the lub's corresponding to the constraints of $\Phi$ are preserved.

It is sometimes the case that $\preceq$ satisfies $\Phi$ but it is impossible to embed $OT/\preceq$ into a complete ordered algebra in such a way that all existing lub's are preserved. Thus our construction is a refinement of the use of a "conservative completion" in section 5.1 of [Berry], which relies upon preserving all existing lub's. Furthermore, by preserving only the necessary lub's, we succeed in producing an initial object in the category of $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebras, together with monotonic homomorphisms.

Lemma 5.1.1 Suppose $\Phi$ is a closed family of least fixed point constraints and $\preceq$ is an $\Omega$-least substitutive pre-ordering over $T$ that satisfies $\Phi$. Define an $S$-indexed family $\Gamma$ of sets of subsets of $OT/\preceq$ by

$$\Gamma_s = \{ qt_s T' \mid t\equiv\bigcup T' \in \Phi_s \}.$$ 

Then $\Gamma$ is a family of subsets of $OT/\preceq$, and $OT/\preceq$ is $\Gamma$-complete.

Proof. Clearly $\Gamma$ consists of sets of directed subsets of $OT/\preceq$. Let $a \in (OT/\preceq)_s, s \in S$; we must show that $\{a\} \in \Gamma_s$. Since $qt$ is surjective, there is a $t \in OT_s$ such that $qt_s t = a$. Furthermore, by lemma 3.2.5, $t\equiv\bigcup\{t\} \in \Phi_s$, and thus

$$\{a\} = \{qt_s t\} = qt_s \{t\} \in \Gamma_s.$$ 

Now, suppose $\sigma \in \Sigma$ has type $s_1 \times \cdots \times s_n \rightarrow s'$ and $t_i\equiv\bigcup T'_i \in \Phi_{s_i}, 1 \leq i \leq n$. Then,

$$\sigma((qt_{s_1} T'_1) \times \cdots \times (qt_{s_n} T'_n))$$

$$= qt_s \sigma(T'_1 \times \cdots \times T'_n)$$

$$\in \Gamma_{s'},$$
since
\[ \sigma(t_1, \ldots, t_n) = \bigcup \sigma(T'_1 \times \cdots \times T'_n) \in \Phi_{s'}. \]

Thus, \( \Gamma \) is indeed a family of subsets of \( OT/\leq \).

Suppose \( t = \bigcup T' \in \Phi_s \), \( s \in S \); we must show that \( qt_s T' \) has a lub in \( (OT/\leq_s) \).

By assumption, \( t \) is a lub of \( T' \) in \( (T_s, \leq_s) \), and thus, by corollary 2.4.13, \( qt_s t \) is the lub of \( qt_s T' \) in \( (OT/\leq_s) \). Suppose \( \sigma \in \Sigma \) has type \( s_1 \times \cdots \times s_n \rightarrow s' \) and \( t_i = \bigcup T'_i \in \Phi_{s_i}, 1 \leq i \leq n \). Then,

\[
\begin{align*}
\sigma(\bigcup qt_{s_1} T'_1, \ldots, \bigcup qt_{s_n} T'_n) & = \sigma(qt_{s_1} t_1, \ldots, qt_{s_n} t_n) \\
& = qt_{s'} \sigma(t_1, \ldots, t_n) \\
& = \bigcup qt_{s'} \sigma(T'_1 \times \cdots \times T'_n) \\
& = \bigcup \sigma((qt_{s_1} T'_1) \times \cdots \times (qt_{s_n} T'_n)).
\end{align*}
\]

Thus \( OT/\leq \) is indeed \( \Gamma \)-complete. \( \square \)

We now give a definition that is based upon lemma 5.1.1 and theorem 2.4.2.

**Definition 5.1.2** Let \( \Phi \) be a closed family of least fixed point constraints and \( \leq \) be an \( \Omega \)-least substitutive pre-ordering over \( T \) that satisfies \( \Phi \). The complete ordered algebra \( I(\leq, \Phi) \) is defined to be \( (OT/\leq)^\Gamma \), where \( \Gamma \) is defined as in the statement of lemma 5.1.1.

**Theorem 5.1.3** Suppose \( \Phi \) is a closed family of least fixed point constraints and \( \leq \) is an \( \Omega \)-least substitutive pre-ordering over \( T \) that satisfies \( \Phi \).

1. \( I(\leq, \Phi) \) is a \( \leq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra.

2. If \( A \) is a \( \leq' \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra, for an \( \Omega \)-least substitutive pre-ordering \( \leq' \) over \( T \) such that \( \leq \subseteq \leq' \), then there is a unique continuous homomorphism \( h: I(\leq, \Phi) \rightarrow A \).
Proof. Let $\Gamma$ be the family of subsets of $\mathcal{O}/\leq$ that was defined in the statement of lemma 5.1.1.

\[
\begin{array}{c}
\mathcal{O} \\
\downarrow \quad \downarrow \\
\mathcal{O}/\leq \\
\downarrow \\
\mathcal{M}/\leq \\
\downarrow \\
(A)
\end{array}
\]

We begin by showing part 1. Clearly $(\mathcal{O}/\leq)^\Gamma$ is a complete ordered algebra. To see that it is $\leq$-inequationally fully abstract, let $t_1, t_2 \in T_s, s \in S$. Then,

\[
t_1 \leq s^t t_2
\]

iff \( qt_s t_1 \subseteq s \cap qt_s t_2 \)

iff \( em_s(qt_s t_1) \subseteq s \cap em_s(qt_s t_2) \),

since $em$ is an order-embedding. To see that $(\mathcal{O}/\leq)^\Gamma$ satisfies $\Phi$, suppose that $t = \bigsqcup T' \in \Phi_s, s \in S$. By assumption, $t$ is a lub of $T'$ in $(T_s, \leq_s)$, and thus, by corollary 2.4.13, $qt_s t = \bigsqcup qt_s T'$. Then,

\[
em_s(qt_s t) = em_s \bigsqcup qt_s T' = \bigsqcup em_s(qt_s T'),
\]

since $em$ is $\Gamma$-continuous.

Next, we consider part 2. If $t_1, t_2 \in S, s \in S$, then

\[
t_1 \leq s^t t_2 \Rightarrow t_1 \leq s^t t_2 \Rightarrow M_s t_1 \subseteq s M_s t_2.
\]

Thus (†) there is a unique monotonic homomorphism $M_{\mathcal{A}/\leq}$ from $\mathcal{O}/\leq$ to $\mathcal{A}$ such that $(M_{\mathcal{A}/\leq}) \circ qt = M_{\mathcal{A}}$. Furthermore, $M_{\mathcal{A}/\leq}$ is $\Gamma$-continuous, since if $t = \bigsqcup T' \in \Phi_s, s \in S$, then

\[
(M_{\mathcal{A}/\leq}) s \bigsqcup qt_s T' = (M_{\mathcal{A}/\leq}) s(qt_s t).
\]

105
Thus (†) there is a unique continuous homomorphism $(M_A/\leq)^F$ from $(\mathcal{OT}/\leq)^F$ to $\mathcal{A}$ such that $(M_A/\leq)^F \circ \text{em} = M_A/\leq$. For uniqueness, suppose $h: (\mathcal{OT}/\leq)^F \to \mathcal{A}$ is a continuous homomorphism. By the initiality of $\mathcal{OT}$, we know that $(h \circ \text{em}) \circ \text{qt} = M_A$, and thus by (†) that $h \circ \text{em} = M_A/\leq$. The fact that $h = (M_A/\leq)^F$ then follows from (†). □

Theorem 5.1.4 Suppose $\Phi$ is a family of least fixed point constraints and $\leq$ is an $\Omega$-least substitutive pre-ordering over $\mathcal{T}$. A $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra exists iff $\leq$ satisfies $\overline{\Phi}$.

Proof. The "if" direction follows immediately from theorem 5.1.3. For the "only if" direction, suppose $\mathcal{A}$ is a $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra. By lemma 3.2.7, $\mathcal{A}$ also satisfies $\overline{\Phi}$. Suppose $t \equiv \bigcup T' \in \overline{\Phi}_s$, $s \in S$. Then $M_S t = \bigcup M_S T'$, and thus $t$ is an ub of $T'$ in $(T_S, \leq_S)$. Suppose $t''$ is also an ub of $T'$. Then $M_S t''$ is an ub of $M_S T'$, and so $M_S t \sqsubseteq_A M_S t''$. But this, in turn, implies that $t \leq_S t''$, showing that $t$ is a lub of $T'$ in $(T_S, \leq_S)$, as required. □

Corollary 5.1.5 If the category of $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebras, together with continuous homomorphisms, is nonempty then it has an initial object, $I(\leq, \Phi)$.

Proof. Immediate from theorems 5.1.3 and 5.1.4. □

We conclude this section with the corollary that $I(\leq, \overline{\Phi})$ is always inductively reachable. Thus, if the category of inductively reachable $\leq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebras, together with continuous homomorphisms, is nonempty then it has $I(\leq, \overline{\Phi})$ as an initial object.
Corollary 5.1.6 If $\Phi$ is a closed family of least fixed point constraints and $\preceq$ is an $\Omega$-least substitutive pre-ordering over $T$ that satisfies $\Phi$ then $I(\preceq, \Phi)$ is inductively reachable.

Proof. By lemma 2.3.33, it is sufficient to show that $I(\preceq, \Phi)$ and $R(I(\preceq, \Phi))$ are order-isomorphic. Since $I(\preceq, \Phi)$ is initial in the category of $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebras, together with continuous homomorphisms (corollary 5.1.5), it is sufficient to show that $R(I(\preceq, \Phi))$ is also initial in this category. It is easy to see that $R(I(\preceq, \Phi))$ is a $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra, since $R(I(\preceq, \Phi)) \preceq I(\preceq, \Phi)$. Let $i$ be the inclusion from $R(I(\preceq, \Phi))$ to $I(\preceq, \Phi)$, so that $i$ is a continuous homomorphism from $R(I(\preceq, \Phi))$ to $I(\preceq, \Phi)$.

Suppose $A$ is a $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra, and let $h: I(\preceq, \Phi) \to A$ be the unique continuous homomorphism. Then $h \circ i$ is the unique continuous homomorphism from $R(I(\preceq, \Phi))$ to $A$, by lemma 2.3.31. $\square$

5.2 More Existence Results

This section consists of two corollaries of theorem 5.1.4. In the first, we show that inequationally fully abstract, complete ordered algebras always exist and give a necessary and sufficient condition for the existence of inequationally correct, least fixed point, complete ordered algebras. In the second we give necessary and sufficient conditions for the existence of equationally fully abstract (respectively, equationally correct), least fixed point, complete ordered algebras, and equationally fully abstract, complete ordered algebras, as well as showing that equationally correct, complete ordered algebras always exist.

Corollary 5.2.1 Let $\preceq$ be an $\Omega$-least substitutive pre-ordering over $T$ and $\Phi$ a family of least fixed point constraints.
1. A \( \preceq \)-inequationally fully abstract, complete ordered algebra exists.

2. A \( \preceq \)-inequationally correct, \( \Phi \)-least fixed point, complete ordered algebra exists iff there exists an \( \Omega \)-least substitutive pre-ordering \( \preceq' \) over \( T \) such that \( \preceq' \subseteq \preceq \) and \( \preceq' \) satisfies \( \Phi \).

**Proof.**

1. By lemma 3.2.6, the least closed family of least fixed point constraints, \( \overline{\mathcal{O}} \), consists of exactly the singleton constraints \( t = \bigcup \{ t \} \), \( t \in T \), \( s \in S \). Thus \( \preceq \) satisfies \( \overline{\mathcal{O}} \), and the result follows by theorem 5.1.4.

2. \( (\Rightarrow) \) Suppose \( A \) is a \( \preceq \)-inequationally correct, \( \Phi \)-least fixed point, complete ordered algebra. Then \( \subseteq_A \subseteq \preceq \). Further, \( A \) is \( \subseteq_A \)-inequationally fully abstract, and thus, by theorem 5.1.4, \( \subseteq_A \) satisfies \( \Phi \). \( (\Leftarrow) \) By theorem 5.1.4, there is a \( \preceq' \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra, and, since \( \preceq' \subseteq \preceq \), \( A \) is \( \preceq \)-inequationally correct.

\( \Box \)

It is easy to find artificial examples of \( \preceq \) and \( \Phi \) such that no \( \preceq \)-inequationally correct, \( \Phi \)-least fixed point, complete ordered algebras exist. It would be quite surprising, however, if natural examples existed.

**Corollary 5.2.2** Let \( \approx \) be a congruence over \( T \) and \( \Phi \) be a family of least fixed point constraints.

1. A \( \approx \)-fully abstract, \( \Phi \)-least fixed point, complete ordered algebra exists iff there is an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( T \) such that \( \approx = \subseteq \cap \preceq \) and \( \preceq \) satisfies \( \Phi \).

2. A \( \approx \)-fully abstract, complete ordered algebra exists iff there is an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( T \) such that \( \approx = \subseteq \cap \preceq \).

3. A \( \approx \)-correct, \( \Phi \)-least fixed point, complete ordered algebra exists iff there is an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( T \) such that \( \subseteq \cap \preceq \subseteq \approx \) and \( \preceq \) satisfies \( \Phi \).
4. A $\approx$-correct, complete ordered algebra exists.

Proof.

1. ($\Rightarrow$) Suppose $A$ is a $\approx$-fully abstract, $\Phi$-least fixed point, complete ordered algebra. Then $\approx = \subseteq_A \cap \supseteq_A$. Further, $A$ is $\subseteq_A$-inequationally fully abstract, and thus, by theorem 5.1.4, $\subseteq_A$ satisfies $\overline{\Phi}$. ($\Leftarrow$) By theorem 5.1.4, there exists a $\approx$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra $A$, and, since $\approx = \subseteq \cap \supseteq$, $A$ is $\approx$-fully abstract.

2. Follows from part 1, with $\Phi = \emptyset$.

3. ($\Rightarrow$) Suppose $A$ is a $\approx$-correct, $\Phi$-least fixed point, complete ordered algebra. Then $\equiv_A \subseteq \approx$. Further, $A$ is $\equiv_A$-fully abstract, and thus, by part 1, there is an $\Omega$-least substitutive pre-ordering $\leq$ over $T$ such that

$$\leq \cap \supseteq = \equiv_A \subseteq \approx$$

and $\leq$ satisfies $\overline{\Phi}$. ($\Leftarrow$) Let $\approx' = \leq \cap \supseteq$. By part 1, there exist a $\approx'$-fully abstract, $\Phi$-least fixed point, complete ordered algebra $A$, and, since

$$\approx' = \leq \cap \supseteq \subseteq \approx,$$

$A$ is $\approx$-correct.

4. Since $\leq^\Omega$ is a partial ordering, $\leq^\Omega \cap \supseteq^\Omega$ is the least congruence over $T$ (no distinct terms are congruent). Thus $\leq^\Omega \cap \supseteq^\Omega \subseteq \approx$, and the result follows by applying part 3, with $\leq = \leq^\Omega$ and $\Phi = \emptyset$.

By lemma 2.2.13, we know that not every congruence $\approx$ over $T$ is induced by an $\Omega$-least substitutive pre-ordering, and thus, by part 2 of corollary 5.2.2, $\approx$-fully abstract, complete ordered algebras do not always exist. It is unclear whether there are naturally occurring congruences that are not induced by such
pre-orderings. Similarly, it is not difficult to find artificial examples of $\approx$ and $\Phi$ such that no $\approx$-correct, $\Phi$-least fixed point, complete ordered algebras exist. It would be surprising, however, if natural examples existed.

Part 1 of corollary 5.2.2 is the basis for the negative results of chapter 6.

5.3 Contextual Full Abstraction and Least Fixed Point Models

In this section, we show that inductively reachable, $\approx$-fully abstract, $\Delta^*$-least fixed point, complete ordered algebras, for congruences $\approx$ over $\mathcal{T}$ and families of contextual least fixed point constraints $\Delta$, are also $\approx$-contextually fully abstract, $\Delta$-contextually least fixed point, complete ordered algebras. Thus $\approx$-contextually fully abstract, $\Delta$-contextually least fixed point, complete ordered algebras exist exactly when $\approx$-fully abstract, $\Delta^*$-least fixed point, complete ordered algebras do.

Theorem 5.3.1 Suppose $A$ is an inductively reachable complete ordered algebra and $\approx$ is a congruence over $\mathcal{T}$. Then $A$ is $\approx$-fully abstract iff $A$ is $\approx$-contextually fully abstract.

Proof. The "if" direction is obvious. (The hypothesis of inductive reachability is not needed.) For the "only if" direction, first note that, by theorem 3.1.5, $A$ is $\approx$-contextually correct. Thus, we need only show that for all derived operators $c_1[v_1, \ldots, v_n]$ and $c_2[v_1, \ldots, v_n]$ of type $s_1 \times \cdots \times s_n \rightarrow s'$: if $c_1(t_1, \ldots, t_n) \approx_{g'} c_2(t_1, \ldots, t_n)$, for all $t_i \in T_{S_i}, 1 \leq i \leq n$, then $c_1_A = c_2_A$. We show this by induction on the arity $n$ of $c_1$ and $c_2$. The case $n = 0$ holds since $A$ is $\approx$-fully abstract. For the induction step, suppose that $c_1[v_1, \ldots, v_{n+1}]$ and $c_2[v_1, \ldots, v_{n+1}]$ are derived operators of type $s_1 \times \cdots \times s_{n+1} \rightarrow s'$, and that $c_1(t_1, \ldots, t_{n+1}) \approx_{g'} c_2(t_1, \ldots, t_{n+1})$, for all $t_i \in T_{S_i}, 1 \leq i \leq n + 1$. We show by
induction over $A_{s_{n+1}}$ that for all $a_{n+1} \in A_{s_{n+1}}$,

$$c_1(a_1, \ldots, a_{n+1}) = c_2(a_1, \ldots, a_{n+1}), \text{ for all } a_i \in A_{s_i}, 1 \leq i \leq n. \quad (5.2)$$

Let $A'$ be the set of all $a_{n+1} \in A_{s_{n+1}}$ such that (5.2). Suppose $t \in T_{s_{n+1}}$; we must show that $M_{s_{n+1}} t \in A'$. Then $(c_1(v_1, \ldots, v_n, t))[v_1, \ldots, v_n]$ and $(c_2(v_1, \ldots, v_n, t))[v_1, \ldots, v_n]$ are derived operators of type $s_1 \times \cdots \times s_n \rightarrow s'$, and, by the inductive hypothesis on $n$, for all $a_i \in A_{s_i}, 1 \leq i \leq n$,

$$c_1(a_1, \ldots, a_{n}, M_{s_{n+1}} t) = (c_1(v_1, \ldots, v_n, t))[a_1, \ldots, a_n] = (c_2(v_1, \ldots, v_n, t))[a_1, \ldots, a_n] = c_2(a_1, \ldots, a_{n}, M_{s_{n+1}} t).$$

Now, suppose $A'' \subseteq A'$ is a directed set; we must show that $\bigcup A'' \in A'$. Suppose $a_i \in A_{s_i}, 1 \leq i \leq n$. Then,

$$c_1(a_1, \ldots, a_{n}, \bigcup A'') = \bigcup c_1(\{a_1\} \times \cdots \times \{a_n\} \times A'') = \bigcup c_2(\{a_1\} \times \cdots \times \{a_n\} \times A'') = c_2(a_1, \ldots, a_{n}, \bigcup A''),$$

since $A$ is complete. \(\square\)

**Theorem 5.3.2** If $\Delta$ is a family of contextual least fixed point constraints and $A$ is an inductively reachable complete ordered algebra then $A$ satisfies $\Delta$ iff $A$ satisfies the family of least fixed point constraints $\Delta^*$.

**Proof.** The “only if” direction follows by lemma 3.2.11. For the “if” direction, it is sufficient to show that for all distinct context variables $v_i \in V_{s_i}, 1 \leq i \leq n$, $c \in T(\{v_1, \ldots, v_n\})_s$ and directed sets $C' \subseteq OT(\{v_1, \ldots, v_n\})_s$: if

$$M_{s} c(t_1, \ldots, t_n) = \bigcup_{c' \in C'} M_{s} c'(t_1, \ldots, t_n),$$

111
for all $t_i \in T_{s'_i}, 1 \leq i \leq n$, then

$$c_A = \bigcup_{c' \in C'} c'_A.$$ 

We show this by induction on $n$. The case $n = 0$ is trivial. For the induction step, suppose that $v_i \in V_{s'_i}, 1 \leq i \leq n + 1$, $c \in T(\{v_1, \ldots, v_{n+1}\})$, $C' \subseteq OT(\{v_1, \ldots, v_{n+1}\})$ is a directed set, and

$$M_{s'} c(t_1, \ldots, t_{n+1}) = \bigcup_{c' \in C'} M_{s'} c'(t_1, \ldots, t_{n+1}),$$

for all $t_i \in T_{s'_i}, 1 \leq i \leq n + 1$. We show by induction over $A_{s'_{n+1}}$ that for all $a_{n+1} \in A_{s'_{n+1}},$

$$c(a_1, \ldots, a_{n+1}) = \bigcup_{c' \in C'} c'(a_1, \ldots, a_{n+1}),$$

for all $a_i \in A_{s'_i}, 1 \leq i \leq n$. \hspace{1cm} (5.3)

Let $A'$ be the set of all $a_{n+1} \in A_{s'_{n+1}}$ such that (5.3). Suppose $t' \in T_{s'_{n+1}}$; we must show that $M_{s'_{n+1}} t' \in A'$. Then,

$$c(v_1, \ldots, v_n, t') \in T(\{v_1, \ldots, v_n\})_s,$$

and

$$\{ c'(v_1, \ldots, v_n, t') \mid c' \in C' \} \subseteq OT(\{v_1, \ldots, v_n\}_s$$

is a directed set. Further, for all $t_i \in T_{s'_i}, 1 \leq i \leq n,$

$$M_{s'} (c(v_1, \ldots, v_n, t'))(t_1, \ldots, t_n)$$

$$= M_{s'} c(t_1, \ldots, t_n, t')$$

$$= \bigcup_{c' \in C'} M_{s'} c'(t_1, \ldots, t_n, t')$$

$$= \bigcup_{c' \in C'} M_{s'} (c'(v_1, \ldots, v_n, t'))(t_1, \ldots, t_n).$$

Thus, by the inductive hypothesis on $n,$

$$c(a_1, \ldots, a_n, M_{s'_{n+1}} t')$$

$$= (c(v_1, \ldots, v_n, t'))(a_1, \ldots, a_n)$$

$$= \bigcup_{c' \in C'} (c'(v_1, \ldots, v_n, t'))(a_1, \ldots, a_n)$$

$$= \bigcup_{c' \in C'} c'(a_1, \ldots, a_n, M_{s'_{n+1}} t'),$$
for all \( a_i \in A_{\delta_i} \), \( 1 \leq i \leq n \). Now, suppose \( A'' \subseteq A' \) is a directed set; we must show that \( \bigcup A'' \in A' \). Let \( a_i \in A_{\delta_i} \), \( 1 \leq i \leq n \). Then,

\[
\begin{align*}
c(a_1, \ldots, a_n, \bigcup A'') & = \bigcup_{a'' \in A''} c(a_1, \ldots, a_n, a'') \\
& = \bigcup_{a'' \in A''} \bigcup_{c' \in C'} c'(a_1, \ldots, a_n, a'') \\
& = \bigcup_{c' \in C'} \bigcup_{a'' \in A''} c'(a_1, \ldots, a_n, a'') \\
& = \bigcup_{c' \in C'} c'(a_1, \ldots, a_n, \bigcup A''),
\end{align*}
\]

as required. \( \Box \)

**Corollary 5.3.3** Suppose \( \approx \) is a congruence over \( \mathcal{T} \) and \( \Delta \) is a family of contextual least fixed point constraints. If \( A \) is a \( \approx \)-fully abstract, \( \Delta^*- \)least fixed point, complete ordered algebra then \( R(A) \) is a \( \approx \)-contextually fully abstract, \( \Delta \)-contextually least fixed point, complete ordered algebra.

**Proof.** Since \( R(A) \preceq A \), \( R(A) \) is also a \( \approx \)-fully abstract, \( \Delta^*- \)least fixed point, complete ordered algebra. The result then follows from theorems 5.3.1 and 5.3.2. \( \Box \)

**Corollary 5.3.4** Suppose \( \approx \) is a congruence over \( \mathcal{T} \) and \( \Delta \) is a family of contextual least fixed point constraints. Then, there exists a \( \approx \)-contextually fully abstract, \( \Delta \)-contextually least fixed point, complete ordered algebra iff there exists a \( \approx \)-fully abstract, \( \Delta^*- \)least fixed point, complete ordered algebra iff there exists an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( \mathcal{T} \) such that \( \approx = \preceq \cap \succeq \) and \( \preceq \) satisfies \( \Delta^c \).

**Proof.** Immediate from lemma 3.2.11, corollary 5.3.3 and part 1 of corollary 5.2.2. \( \Box \)
5.4 Categorical Properties

In this section, we prove theorems concerning the existence of initial objects and the nonexistence of terminal objects in various categories of correct and fully abstract, least fixed point, complete ordered algebras, and show the existence of nonisomorphic inductively reachable, inequationally fully abstract, least fixed point, complete ordered algebras.

To begin with, we name the categories we will be considering. Let $L(\Phi)$ be the category of $\Phi$-least fixed point, complete ordered algebras, together with continuous homomorphisms. Define the following full subcategories of $L(\Phi)$.

<table>
<thead>
<tr>
<th>Category</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C(\simeq, \Phi)$</td>
<td>$\simeq$-correct</td>
</tr>
<tr>
<td>$FA(\simeq, \Phi)$</td>
<td>$\simeq$-fully abstract</td>
</tr>
<tr>
<td>$IC(\preceq, \Phi)$</td>
<td>$\preceq$-inequationally correct</td>
</tr>
<tr>
<td>$IFA(\preceq, \Phi)$</td>
<td>$\preceq$-inequationally fully abstract</td>
</tr>
</tbody>
</table>

In addition, let $RL(\Phi), RC(\simeq, \Phi), RFA(\simeq, \Phi), RIC(\preceq, \Phi), \text{ and } RIFA(\preceq, \Phi)$ be the full subcategories of $L(\Phi), C(\simeq, \Phi)$, etc., whose objects are inductively reachable. Note that $FA(\simeq, \Phi)$ (respectively, $RFA(\simeq, \Phi)$) is a subcategory of $C(\simeq, \Phi)$ (respectively, $RC(\simeq, \Phi)$), and $IFA(\preceq, \Phi)$ (respectively, $RIFA(\preceq, \Phi)$) is subcategory of $IC(\preceq, \Phi)$ (respectively, $RIC(\preceq, \Phi)$).

In section 5.1, we learned that if the category $IFA(\preceq, \Phi)$ is nonempty then it has an initial object, $I(\preceq, \Phi)$. We now prove analogous theorems for our other categories. Theorem 5.4.1 shows that $L(\Phi)$ always has an initial object $A$, and that if $C(\simeq, \Phi)$ (respectively, $IC(\preceq, \Phi)$) is nonempty then it also has $A$ as an initial object.

**Theorem 5.4.1** Suppose $\Phi$ is a family of least fixed point constraints and let $\preceq^0$ be the least $\Omega$-least substitutive pre-ordering over $\mathcal{T}$ that satisfies $\overline{\Phi}$.

1. $I(\preceq^0, \Phi)$ is initial in $L(\Phi)$. 

114
2. If $C(\simeq, \Phi)$ is nonempty, for a congruence $\simeq$ over $T$, then it has $I(\preceq^0, \Phi)$ as an initial object.

3. If $IC(\preceq, \Phi)$ is nonempty, for an $\Omega$-least substitutive pre-ordering $\preceq$ over $T$, then it has $I(\preceq^0, \Phi)$ as an initial object.

Proof. We begin by showing that such a $\preceq^0$ exists. Let $X$ be the set of all $\preceq$ such that $\preceq$ is an $\Omega$-least substitutive pre-ordering over $T$ that satisfies $\Phi$. Then $X$ is nonempty, since the greatest $\Omega$-least substitutive pre-ordering over $T$ (every term is less than every other term) satisfies $\Phi$, and it is easy to see that $\cap X$ is the least $\Omega$-least substitutive pre-ordering over $T$ that satisfies $\Phi$.

1. Clearly $I(\preceq^0, \Phi)$ is an object of $L(\Phi)$. Suppose $A$ is also a $\Phi$-least fixed point, complete ordered algebra. Then $A$ is $\subseteq A$-inequationally fully abstract, and, by lemma 3.2.7, $A$ satisfies $\Phi$. By theorem 5.1.4, $\subseteq A$ satisfies $\Phi$, and thus, by the leastness of $\preceq^0$, $\preceq^0 \subseteq \subseteq A$. The existence of the unique continuous homomorphism from $I(\preceq^0, \Phi)$ to $A$ then follows from theorem 5.1.3.

2. By part 1, it is sufficient to show that $I(\preceq^0, \Phi)$ is an object of $C(\simeq, \Phi)$, i.e., that it is $\simeq$-correct. Let $\simeq^0 = \preceq^0 \cap \succeq^0$. Then $I(\preceq^0, \Phi)$ is $\simeq^0$-fully abstract, and thus it is sufficient to show that $\simeq^0 \subseteq \simeq$. By part 3 of corollary 5.2.2, there is an $\Omega$-least substitutive pre-ordering $\preceq$ over $T$ such that $\preceq \cap \succeq \subseteq \simeq$ and $\preceq$ satisfies $\Phi$. Then, by the leastness of $\preceq^0$, $\preceq^0 \subseteq \preceq$, and thus

$$\simeq^0 = \preceq^0 \cap \succeq^0 \subseteq \preceq \cap \succeq \subseteq \simeq.$$

3. By part 1, it is sufficient to show that $I(\preceq^0, \Phi)$ is an object of $IC(\preceq, \Phi)$, i.e., that it is $\preceq$-inequationally correct. Thus it is sufficient to show that $\preceq^0 \subseteq \preceq$. By part 2 of corollary 5.2.1, there is an $\Omega$-least substitutive pre-ordering $\preceq'$ over $T$ such that $\preceq' \subseteq \preceq$ and $\preceq'$ satisfies $\Phi$. Then, by the leastness of $\preceq^0$, $\preceq^0 \subseteq \preceq'$, and thus $\preceq^0 \subseteq \preceq$. 

115
Theorem 5.4.2 Suppose \( \Phi \) is a family of least fixed point constraints and \( \approx \) is a congruence over \( \mathcal{T} \). If \( \text{FA}(\approx, \Phi) \) is nonempty then it has \( I(\preceq^0, \Phi) \) as an initial object, where \( \preceq^0 \) is the least \( \Omega \)-least substitutive pre-ordering over \( \mathcal{T} \) such that \( \preceq^0 \) satisfies \( \Phi \) and \( \approx = \preceq^0 \cap \succeq^0 \).

Proof. We begin by showing that such a \( \preceq^0 \) exists. Let \( X \) be the set of all \( \Omega \)-least substitutive pre-orderings over \( \mathcal{T} \) that satisfy \( \Phi \) and induce \( \approx \). Then \( X \) is nonempty, by part 1 of corollary 5.2.2, and it is easy to see that \( \bigcap X \) may be taken as \( \preceq^0 \).

Clearly \( I(\preceq^0, \Phi) \) is an object of \( \text{FA}(\approx, \Phi) \). Suppose \( A \) is also a \( \approx \)-fully abstract, \( \Phi \)-least fixed point, complete ordered algebra. Then \( \approx = \preceq_A \cap \succeq_A \), and, by theorem 5.1.4, \( \preceq_A \) satisfies \( \Phi \). By the leastness of \( \preceq^0 \), \( \preceq^0 \subseteq \preceq_A \), and thus, by theorem 5.1.3, there is a unique continuous homomorphism from \( I(\preceq^0, \Phi) \) to \( A \). \( \square \)

We now turn our attention to the subcategories of inductively reachable objects: \( \text{RC}(\approx, \Phi) \), \( \text{RFA}(\approx, \Phi) \), \( \text{RIC}(\preceq, \Phi) \) and \( \text{RIFA}(\preceq, \Phi) \). Since \( I(\preceq, \Phi) \) is always inductively reachable, all of these categories have initial objects whenever they are nonempty.

The next theorem shows, perhaps surprisingly, that \( \text{RIFA}(\preceq, \Phi) \) can have nonisomorphic objects.

Theorem 5.4.3 There is a signature \( \Sigma \), an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( \mathcal{T} \) and a family of least fixed point constraints \( \Phi \) such that \( \text{RIFA}(\preceq, \Phi) \) has nonisomorphic objects.

Proof. Let \( \Sigma \) over \( S = \{*, \} \) consist of the following nullary operators: \( \Omega_*, x \) and \( n, n \in \omega \). Since there is only one sort, we drop the sort subscripts from carriers, relations, etc., below. Define ordered algebras \( A \) and \( B \) as follows. Their carriers are defined by
so that $x = \bigcup_A \omega$ and $y = \bigcup_B \omega$. Their operations are interpreted by themselves. It is easy to see that $A$ and $B$ are non order-isomorphic inductively reachable, complete ordered algebras. Furthermore, $\subseteq_A = \subseteq_B$. Thus the theorem holds with $\leq = \subseteq_A$ and $\Phi = \emptyset$. $\Box$

We now consider the existence of terminal objects in our categories of inductively reachable objects. Theorem 5.4.4 shows that $\mathbf{RFA}(\approx, \Phi)$ can be nonempty yet lack a terminal object. Thus, even when $\mathbf{RFA}(\approx, \Phi)$ is nonempty, $\mathbf{RC}(\approx, \Phi)$ can lack a terminal object. The situation is less clear for $\mathbf{RIFA}(\preceq, \Phi)$ and $\mathbf{RIC}(\preceq, \Phi)$. Theorem 5.4.5 shows that $\mathbf{RIC}(\preceq, \Phi)$ can lack a terminal object, even when $\mathbf{RIFA}(\preceq, \Phi)$ is nonempty. It is open, however, whether $\mathbf{RIFA}(\preceq, \Phi)$ always has a terminal object whenever it is nonempty; I conjecture that it always does.

**Theorem 5.4.4** There is a signature $\Sigma$, a congruence $\approx$ over $\mathcal{T}$ and a family of least fixed point constraints $\Phi$ such that $\mathbf{RFA}(\approx, \Phi)$ is nonempty but lacks a terminal object.

**Proof.** Let $\Sigma$ over $S = \{\ast\}$ consist of the nullary operators $\Omega_\ast$, $x$ and $y$. Since there is only one sort, we drop the sort subscripts from carriers, relations, etc., below. Define ordered algebras $A$ and $B$ as follows. Their carriers are defined by

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>$\Omega$</td>
</tr>
</tbody>
</table>

117
and their operations are interpreted by themselves. Clearly \( A \) and \( B \) are inductively reachable complete ordered algebras. Furthermore, \( \equiv_A = \equiv_B \), and, in particular, \( x \not\equiv_A y \). Thus \( A \) and \( B \) are \( \text{RFA}(\simeq, \Phi) \) objects, where \( \simeq = \equiv_A \) and \( \Phi = \emptyset \). Suppose, toward a contradiction, that \( C \) is terminal in \( \text{RFA}(\simeq, \Phi) \), and let \( f: A \to C \) and \( g: B \to C \) be the unique continuous homomorphisms. But then

\[
M_C x = f x \subseteq_C f y = M_C y
\]

and

\[
M_C y = g y \subseteq_C g x = M_C x,
\]

showing that \( M_C x = M_C y \)—a contradiction. \( \Box \)

**Theorem 5.4.5** There is a signature \( \Sigma \), an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( \mathcal{T} \) and a family of least fixed point constraints \( \Phi \) such that \( \text{RIFA}(\preceq, \Phi) \) is nonempty but \( \text{RIC}(\preceq, \Phi) \) lacks a terminal object. In particular, there is a \( \text{RIC}(\preceq, \Phi) \) object that cannot be collapsed, via a continuous homomorphism, to any \( \text{RIFA}(\preceq, \Phi) \) object.

**Proof.** Let \( \Sigma \) over \( S = \{\ast\} \) consist of the following nullary operators: \( \Omega_\ast \), \( x \), \( y \) and \( n \), \( n \in \omega \). Since there is only one sort, we drop the sort subscripts from carriers, relations, etc., below. Define \( \preceq \) over \( \mathcal{T} \) by
and let $\Phi = \emptyset$. Then $I(\leq, \Phi)$ is a RIFA($\leq, \Phi$) object. Define an ordered algebra $A$ as follows. Its carrier is defined by

and its operations are interpreted by themselves. It is easy to see that $A$ is an inductively reachable complete ordered algebra. Furthermore, $A$ is $\leq$-inequationally correct, and thus is a RIC($\leq, \Phi$) object. Suppose, toward a contradiction, that $B$ is a RIFA($\leq, \Phi$) object, and that $h: A \to B$ is a continuous homomorphism. Then $M_B y = \bigcup M_B \omega$. By inequational full abstraction, $M_B x$ is an ub of $M_B \omega$, and thus $M_B y \subseteq_B M_B x$. But this implies that $y \leq x$—a contradiction. Finally, suppose toward a contradiction that RIC($\leq, \Phi$) has a terminal object, $C$. Then $C$ is $\leq$-inequationally fully abstract, since RIFA($\leq, \Phi$) is nonempty. But, by the above, this yields a contradiction. \(\square\)

Conjecture 5.4.6 The category RIFA($\leq, \Phi$) always has a terminal object, whenever it is nonempty.

My reasons for making this conjecture are largely negative: my attempts at finding a counterexample have failed. To prove the conjecture, it would be
sufficient to show that inequational full abstraction is preserved by arbitrary coproducts in the category of inductively reachable complete ordered algebras, together with continuous homomorphisms. Then the terminal object would be the coproduct of representatives of all of the isomorphism classes in $\text{RIFA}(\preceq, \Phi)$. (The number of isomorphism classes in $\text{RIFA}(\preceq, \Phi)$ is bounded, since every element of an inductively reachable complete ordered algebra is the lub of a (not necessarily directed) set of elements that are definable by terms.)
Chapter 6

Negative Results

This chapter consists of proofs of the nonexistence of fully abstract models of two nondeterministic imperative programming languages: one with random assignment and the other with infinite output streams. We give operational semantics for these languages, define notions of program equivalence in terms of these semantics, and use the condition for the existence of equationally fully abstract, least fixed point, complete ordered algebras given in chapter 5 in order to prove the negative results. No model-theoretic reasoning is involved in these proofs.

The language with random assignment is taken from [AptPlotkin] (with minor variations). Our proof of the nonexistence of fully abstract models of this language is a simplification of theirs. Our treatment of the language with infinite output streams is motivated by Abramsky's negative result for a nondeterministic applicative language with infinite streams [Abramsky3].

6.1 A Language with Random Assignment

In this section, we study a nondeterministic imperative programming language with random assignment statements ($x := ?$), which nondeterministically choose natural numbers and assign them to identifiers. The language also includes binary nondeterministic choice ($or$), which nondeterministically selects one of
its arguments to be executed, as well as the usual null \((\text{skip})\), assignment \((x:=n,\) etc.), sequencing \((;)\), conditional and iteration statements. We begin by defining the language's syntax, i.e., its signature.

Let \(I\) be a countably infinite set of identifiers, and the set of boolean expressions \(\text{Exp}\) be

\[
\{ x \equiv 0 \mid x \in I \} \cup \{ x \not\equiv 0 \mid x \in I \}.
\]

Define a signature \(\Sigma\) over \(S = \{\ast\}\) with the following operators:

- \(\Omega_*\), \(\text{skip}\), \(x:=n\), \(x:=x+1\), \(x:=x-1\), \(x:=y\) and \(x:=?\) of type \(\ast\), for all \(x,y \in I\) and \(n \in \omega\);
- \(\text{while} E \text{do-od}\) of type \(\ast \rightarrow \ast\), for all \(E \in \text{Exp}\);
- \(;\), \(\text{or}\), and \(\text{if} E \text{then-else-fi}\) of type \(\ast \times \ast \rightarrow \ast\), for all \(E \in \text{Exp}\).

We let \(;\) and \(\text{or}\) associate to the right, and drop the single sort \(\ast\) from carriers, relations, etc., below.

Let the set of states \(\text{Sta}\) be \(I \rightarrow N\). For \(\sigma \in \text{Sta}, x \in I\) and \(n \in N\), define \(\sigma[x] \in N\) and \(\sigma[n/x] \in \text{Sta}\) by:

\[
\begin{align*}
\sigma[x] &= \sigma x, \\
\sigma[n/x] y &= \begin{cases} 
n & \text{if } y = x \\
\sigma y & \text{otherwise}
\end{cases}
\end{align*}
\]

Define an evaluation map for boolean expressions \(\mathcal{E}: \text{Exp} \rightarrow \text{Sta} \rightarrow \text{Tr}\) by:

\[
\begin{align*}
\mathcal{E} x \equiv 0 \sigma &= \begin{cases} 
ten & \text{if } \sigma[x] = 0 \\
\text{false} & \text{if } \sigma[x] \neq 0
\end{cases}, \\
\mathcal{E} x \not\equiv 0 \sigma &= \begin{cases} 
ten & \text{if } \sigma[x] \neq 0 \\
\text{false} & \text{if } \sigma[x] = 0
\end{cases}.
\end{align*}
\]

We define a transition system for our language as follows. Its set of configurations \(\Gamma\) is \((T \times \text{Sta}) \cup \text{Sta}\). Its transition relation \(\rightarrow\) is the least binary relation
over $\Gamma$ satisfying the following conditions, for all $x, y \in I$, $n \in \omega$, $E \in \text{Exp}$, $t, t_1, t_1', t_2 \in T$ and $\sigma, \sigma' \in \text{Sta}$:

$$\langle \Omega, \sigma \rangle \rightarrow \langle \Omega, \sigma \rangle,$$

$$\langle \text{skip}, \sigma \rangle \rightarrow \sigma,$$

$$\langle x:=n, \sigma \rangle \rightarrow \sigma[n/x],$$

$$\langle x:=x+1, \sigma \rangle \rightarrow \sigma[x + 1/x],$$

$$\langle x:=x-1, \sigma \rangle \rightarrow \sigma[x - 1/x] \quad (\sigma[x] \neq 0),$$

$$\langle x:=x-1, \sigma \rangle \rightarrow \sigma \quad (\sigma[x] = 0),$$

$$\langle x:=y, \sigma \rangle \rightarrow \sigma[y/x],$$

$$\langle x:=?, \sigma \rangle \rightarrow \sigma[n/x],$$

$$\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow \langle t; \text{ while } E \text{ do } t \text{ od}, \sigma \rangle \quad (\mathcal{E} \ E \sigma = \text{tt}),$$

$$\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow \sigma \quad (\mathcal{E} \ E \sigma = \text{ff}),$$

$$\langle t_1; t_2, \sigma \rangle \rightarrow \langle t_1', t_2', \sigma' \rangle \quad \frac{\langle t_1, \sigma \rangle \rightarrow \langle t_1', \sigma' \rangle}{\langle t_1; t_2, \sigma \rangle \rightarrow \langle t_1'; t_2, \sigma' \rangle},$$

$$\langle t_1 \text{ or } t_2, \sigma \rangle \rightarrow \langle t_1, \sigma \rangle, \quad \langle t_1 \text{ or } t_2, \sigma \rangle \rightarrow \langle t_2, \sigma \rangle,$$

$$\langle \text{if } E \text{ then } t_1 \text{ else } t_2, \sigma \rangle \rightarrow \langle t_1, \sigma \rangle \quad (\mathcal{E} \ E \sigma = \text{tt}),$$

$$\langle \text{if } E \text{ then } t_1 \text{ else } t_2, \sigma \rangle \rightarrow \langle t_2, \sigma \rangle \quad (\mathcal{E} \ E \sigma = \text{ff}).$$
Define a family \( \rightarrow_n, n \in \omega \), of binary relations over \( \Gamma \) by:

\[
\gamma_1 \rightarrow_0 \gamma_2 \quad \text{iff} \quad \gamma_1 = \gamma_2,
\]

\[
\gamma_1 \rightarrow_{n+1} \gamma_2 \quad \text{iff} \quad \gamma_1 \rightarrow_n \gamma' \rightarrow \gamma_2, \text{ for some } \gamma' \in \Gamma.
\]

Thus, \( \gamma_1 \rightarrow^* \gamma_2 \) iff there exists an \( n \in \omega \) such that \( \gamma_1 \rightarrow_n \gamma_2 \). We say that \( \gamma \) may diverge, written \( \gamma \uparrow \), iff there exists a \( \gamma_i \in \Gamma^\omega \) such that \( \gamma_0 = \gamma \), and \( \gamma_i \rightarrow \gamma_{i+1} \), for all \( i \in \omega \).

Next, we define a notion of program equivalence for our language. Define an evaluation map

\[
O : T \rightarrow \text{Sta} \rightarrow \mathcal{P}(\text{Sta} \cup \{\bot\})
\]

(for some \( \bot \not\in \text{Sta} \)) by:

\[
O(t, \sigma) = \{ \sigma' \mid (t, \sigma) \rightarrow^* \sigma' \} \cup \{ \bot \mid (t, \sigma) \uparrow \}.
\]

Define an equivalence relation \( \approx \) over \( T \) by:

\[
t_1 \approx t_2 \quad \text{iff} \quad O(t_1) = O(t_2).
\]

Then \( \approx^c \) is a congruence over \( T \). The next lemma shows that \( \approx \) is already a congruence.

Lemma 6.1.1 \( \approx^c = \approx \)

Proof. By lemma 2.2.26, it is sufficient to show that \( \approx \) is substitutive. We only show substitutivity under \( ; \) and \( \text{while } E \text{ do } - od \), leaving the \( \text{or} \) and \( \text{if } E \text{ then } - else - fi \) cases, which are simpler, to the reader.

For \( ; \), suppose that \( t_1 \approx t_1' \) and \( t_2 \approx t_2' \); we must show that \( t_1; t_2 \approx t_1'; t_2' \). By the symmetry of \( \approx \), it is sufficient to show that

\[
\frac{(t_1; t_2, \sigma) \rightarrow^* \sigma'}{(t_1'; t_2', \sigma) \rightarrow^* \sigma'}, \text{ for all } \sigma, \sigma' \in \text{Sta},
\]

and

\[
\frac{(t_1; t_2, \sigma) \uparrow}{(t_1'; t_2', \sigma) \uparrow}, \text{ for all } \sigma \in \text{Sta}.
\]
For (6.4), if \( \langle t_1; t_2, \sigma \rangle \rightarrow* \sigma' \) then there is a \( \sigma'' \) such that \( \langle t_1, \sigma \rangle \rightarrow* \sigma'' \) and \( \langle t_2, \sigma'' \rangle \rightarrow* \sigma'. \) Thus, from the assumption that \( t_i \approx t'_i, \ i = 1, 2, \) it follows that \( \langle t'_1, \sigma \rangle \rightarrow* \sigma'' \) and \( \langle t'_2, \sigma'' \rangle \rightarrow* \sigma', \) and thus that \( \langle t'_1; t'_2, \sigma \rangle \rightarrow* \sigma'. \) For (6.5), if \( \langle t_1; t_2, \sigma \rangle \uparrow \) then either \( \langle t_1, \sigma \rangle \uparrow \) or there is a \( \sigma' \) such that \( \langle t_1, \sigma \rangle \rightarrow* \sigma' \) and \( \langle t_2, \sigma' \rangle \uparrow. \) In the first case, \( \langle t'_1, \sigma \rangle \uparrow, \) by the assumption, and thus \( \langle t'_1; t'_2, \sigma \rangle \uparrow. \) In the second case, \( \langle t'_1, \sigma \rangle \rightarrow* \sigma' \) and \( \langle t'_2, \sigma' \rangle \uparrow, \) showing that \( \langle t'_1; t'_2, \sigma \rangle \uparrow. \)

For \( \text{while } E \text{ do } - \text{ od, } E \in \text{Exp, } \) suppose that \( t \approx t'; \) we must show that

\[
\text{while } E \text{ do } t \text{ od } \approx \text{while } E \text{ do } t' \text{ od.}
\]

By the symmetry of \( \approx, \) it is sufficient to show that

\[
\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow* \sigma', \quad \text{for all } \sigma, \sigma' \in \text{Sta},
\]

and

\[
\langle \text{while } E \text{ do } t' \text{ od}, \sigma \rangle \uparrow, \quad \text{for all } \sigma \in \text{Sta.}
\]

For (6.6), it is sufficient to show that for all \( n \in \omega, \)

\[
\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow_n \sigma', \quad \text{for all } \sigma, \sigma' \in \text{Sta.}
\]

We prove this by course of values induction over \( n. \) Suppose that

\[
\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow_n \sigma'.
\]

If \( \mathcal{E} E \sigma = \text{ff } \) then \( \sigma = \sigma' \) and

\[
\langle \text{while } E \text{ do } t' \text{ od}, \sigma \rangle \rightarrow* \sigma'.
\]

So, assume that \( \mathcal{E} E \sigma = \text{tt}. \) Then there is a \( \sigma'' \) and an \( m < n \) such that

\[
\langle t, \sigma \rangle \rightarrow* \sigma'',
\]

\[
\langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow \langle t; \text{while } E \text{ do } t \text{ od}, \sigma \rangle \rightarrow* \langle \text{while } E \text{ do } t \text{ od}, \sigma'' \rangle,
\]

125
and 

\[ \langle \text{while } E \text{ do } t \text{ od}, \sigma' \rangle \rightarrow_m \sigma'. \]

The result then follows from the assumption that \( t \approx t' \) and the inductive hypothesis on \( m \). For (6.7), note that for all \( \sigma \) and \( \gamma \in \Gamma^w \), if \( \gamma_0 = \langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \), and \( \gamma_i \rightarrow \gamma_{i+1} \), for all \( i \in \omega \), then either \( \langle t, \sigma \rangle \uparrow \) or there exists an \( i > 0 \) and a \( \sigma' \) such that \( \langle t, \sigma \rangle \rightarrow^* \sigma' \) and \( \gamma_i = \langle \text{while } E \text{ do } t \text{ od}, \sigma' \rangle \).

Thus, if \( \langle \text{while } E \text{ do } t \text{ od}, \sigma \rangle \uparrow \) then we can choose a \( \gamma \in \Gamma^w \) such that \( \gamma_0 = \langle \text{while } E \text{ do } t' \text{ od}, \sigma \rangle \), and \( \gamma_i \rightarrow \gamma_{i+1} \), for all \( i \in \omega \). \( \square \)

Let the family of least fixed point constraints \( \Phi \) be

\[
\{ \text{while } E \text{ do } t \text{ od} \equiv \bigcup \{ W^n(E, t) \mid n \in \omega \} \mid E \in \text{Exp}, t \in T \},
\]

where \( W^n(E, t) \) is the \( \omega \)-chain in \( OT \) defined by:

\[
W^0(E, t) = \Omega, \quad W^{n+1}(E, t) = \text{ if } E \text{ then } t; W^n(E, t) \text{ else skip } \]

Lemma 6.1.2 A complete ordered algebra \( A \) is a \( \Phi \)-least fixed point model iff for all \( E \in \text{Exp} \) and \( t \in T \),

\[
M \text{ while } E \text{ do } t \text{ od} = \bigsqcup_{n \in \omega} W^n(E, t),
\]

where \( W^n(E, t) \) is the \( \omega \)-chain in \( A \) defined by

\[
w^0(E, t) = \bot, \quad w^{n+1}(E, t) = \text{ if } E \text{ then } (M t); w^n(E, t) \text{ else skip }.
\]

Proof. A simple induction on \( n \) shows that for all \( n \in \omega \), \( MW^n(E, t) = w^n(E, t) \), and thus

\[
M \text{ while } E \text{ do } t \text{ od} = \bigsqcup_{n \in \omega} MW^n(E, t)
\]

iff \( M \text{ while } E \text{ do } t \text{ od} = \bigsqcup_{n \in \omega} w^n(E, t) \),

as required. \( \square \)
We can now prove the main result of this section: there is no fully abstract, least fixed point model for our programming language.

**Theorem 6.1.3** There does not exist a $\approx$-fully abstract, $\Phi$-least fixed point, complete ordered algebra.

**Proof.** Suppose, toward a contradiction, that such an ordered algebra does exist. Then, by part 1 of corollary 5.2.2, there is an $\Omega$-least substitutive pre-ordering $\preceq$ over $T$ such that $\approx = \preceq \cap \succeq$ and $\preceq$ satisfies $\Phi$. Let the term $t$ be

$$x:=?; \text{while } x\neq 0 \text{ do } x:=x-1 \text{ od},$$

and the $\omega$-chain $t'_n$ in $OT$ be defined by

$$x:=?; W^n(x\neq 0, x:=x-1).$$

Then $t$ is a lub of $t'_n$ in $\langle T, \preceq \rangle$, since $t \equiv \bigcup \{ t'_n \mid n \in \omega \} \in \Phi$ and $\preceq$ satisfies $\Phi$. But $t \approx x:=0, t'_0 \approx \Omega$ and $t'_{n+1} \approx x:=0 \text{ or } \Omega$, for all $n \in \omega$, which implies that $x:=0$ is a lub of $\{ \Omega, x:=0 \text{ or } \Omega \}$ in $\langle T, \preceq \rangle$, and thus that $x:=0 \approx x:=0 \text{ or } \Omega$—a contradiction. \(\Box\)

An apparently stronger result is actually proved in [AptPlotkin]: there does not exist a $\Phi$-least fixed point, complete ordered algebra $A$, together with a continuous full abstraction function, i.e., a continuous function $h$ from $A$ to a cpo $B$ with the property that

$$t_1 \approx t_2 \text{ iff } h(M t_1) = h(M t_2),$$

for all $t_1, t_2 \in T$. Corollary 7.1.2 shows, however, that if a full abstraction function exists for a least fixed point model of a programming language then a fully abstract, least fixed point model also exists for that language. Thus their result follows, by a language independent corollary, from theorem 6.1.3.

On the other hand, the negative result of [AptPlotkin] is stronger than ours in the following respect. As an essential part of our theory, we have included
the constant $\Omega$ in our language, and required that it be interpreted as the least element of any model. Furthermore, it is easy to see that any term that diverges in all states, such as

$$x:=0; \text{while } x\equiv 0 \text{ do skip od},$$

is equivalent to $\Omega$, and thus must also have the value $\bot$ in any model. Thus our theorem 6.1.3 leaves open the possibility that a fully abstract, least fixed point model exists in which such divergent terms have a non-$\bot$ meaning. The negative result of [AptPlotkin] shows, however, that no such models exist. See the conclusion for some more discussion of this point.

6.2 A Nondeterministic Language with Infinite Output Streams

In this section, we study a nondeterministic imperative programming language with output statements $(\text{output } x)$, which write the values of identifiers into potentially infinite length output streams. Otherwise the language is the same as that of section 6.1, with the exception that random assignment statements are not included. We begin by defining the language's syntax.

Let the sets $I$ of identifiers, $\text{Exp}$ of boolean expressions and $\text{Sta}$ of states be the same as in section 6.1. The signature $\Sigma$ is also the same, with the exception that the family of constants $x:=?, x \in I$, is replaced by the family $\text{output } x$, $x \in I$. We define a transition system for our language as follows. Its set of configurations $\Gamma$ is

$$(T \times \text{Sta} \times \mathbb{N}^*) \cup (\text{Sta} \times \mathbb{N}^*),$$

where the element $\delta \in \mathbb{N}^*$ in a configuration $\gamma$ is intended to be the output produced before reaching $\gamma$. Its transition relation $\rightarrow$ is the least binary relation over $\Gamma$ satisfying the following conditions, for all $x, y \in I$, $E \in \text{Exp}$, $n \in \omega$,
$t, t_1, t_2 \in T$, $\sigma, \sigma' \in Sta$ and $\delta, \delta' \in N^*$:

\[
\langle \Omega, \sigma, \delta \rangle \rightarrow \langle \Omega, \sigma, \delta \rangle,
\]

\[
\langle \text{skip}, \sigma, \delta \rangle \rightarrow \langle \sigma, \delta \rangle,
\]

\[
\langle x:=n, \sigma, \delta \rangle \rightarrow \langle \sigma[n/x], \delta \rangle,
\]

\[
\langle x:=x+1, \sigma, \delta \rangle \rightarrow \langle \sigma[x+1/x], \delta \rangle,
\]

\[
\langle x:=x-1, \sigma, \delta \rangle \rightarrow \langle \sigma[x-1/x], \delta \rangle \quad (\sigma[x] \neq 0),
\]

\[
\langle x:=x-1, \sigma, \delta \rangle \rightarrow \langle \sigma, \delta \rangle \quad (\sigma[x] = 0),
\]

\[
\langle x:=y, \sigma, \delta \rangle \rightarrow \langle \sigma[y/x], \delta \rangle,
\]

\[
\langle \text{output } x, \sigma, \delta \rangle \rightarrow \langle \sigma, \delta \langle \sigma[x] \rangle \rangle,
\]

\[
\langle \text{while } E \text{ do } t_1 \text{ od}, \sigma, \delta \rangle \rightarrow \langle t; \text{while } E \text{ do } t_1 \text{ od}, \sigma, \delta \rangle \quad (\mathcal{E} E \sigma = \mathcal{tt}),
\]

\[
\langle \text{while } E \text{ do } t_2 \text{ od}, \sigma, \delta \rangle \rightarrow \langle \sigma, \delta \rangle \quad (\mathcal{E} E \sigma = \mathcal{ff}),
\]

\[
\frac{\langle t_1, \sigma, \delta \rangle \rightarrow \langle t'_1, \sigma', \delta' \rangle}{\langle t_1 ; t_2, \sigma, \delta \rangle \rightarrow \langle t'_1 ; t_2, \sigma', \delta' \rangle}, \quad \frac{\langle t_1, \sigma, \delta \rangle \rightarrow \langle \sigma', \delta' \rangle}{\langle t_1 ; t_2, \sigma, \delta \rangle \rightarrow \langle t_2, \sigma', \delta' \rangle},
\]

\[
\langle \text{if } E \text{ then } t_1 \text{ else } t_2 \text{ fi}, \sigma, \delta \rangle \rightarrow \langle t_1, \sigma, \delta \rangle \quad (\mathcal{E} E \sigma = \mathcal{tt}),
\]

\[
\langle \text{if } E \text{ then } t_1 \text{ else } t_2 \text{ fi}, \sigma, \delta \rangle \rightarrow \langle t_2, \sigma, \delta \rangle \quad (\mathcal{E} E \sigma = \mathcal{ff}).
\]

Define $\text{out}: \Gamma \rightarrow N^*$ by:

\[
\text{out } \langle t, \sigma, \delta \rangle = \delta,
\]

\[
\text{out } \langle \sigma, \delta \rangle = \delta.
\]
It is easy to see that $\text{out } \gamma_1 \subseteq \text{out } \gamma_2$ if $\gamma_1 \rightarrow \gamma_2$. For $\gamma \in \Gamma$ and $\delta \in N^\infty$, we say that $\gamma$ may diverge with output $\delta$, written $\gamma \uparrow \delta$, iff there is a $\vec{\gamma} \in \Gamma^\omega$ such that $\vec{\gamma}_0 = \gamma$, $\vec{\gamma}_i \rightarrow \vec{\gamma}_{i+1}$, for all $i \in \omega$, and $\delta = \bigcup_{i \in \omega} \text{out } \vec{\gamma}_i$.

Next, we define a notion of program equivalence for our language. Define an evaluation map

$$O : T \rightarrow Sta \rightarrow P[(Sta \times N^*) \cup \{\bot \times N^\infty\}],$$

for some $\bot \notin Sta$, by:

$$O \cdot \sigma = \{ \langle \sigma', \delta \rangle \mid \langle t, \sigma, \{\} \rangle \rightarrow^* \langle \sigma', \delta \rangle \} \cup \{ \langle \bot, \delta \rangle \mid \langle t, \sigma, \{\} \rangle \uparrow \delta \}.$$

Define an equivalence relation $\approx$ over $T$ by:

$$t_1 \approx t_2 \text{ iff } O \cdot t_1 = O \cdot t_2.$$

Then $\approx^c$ is a congruence over $T$. The next lemma shows that $\approx$ is already a congruence.

Lemma 6.2.1 $\approx^c = \approx$

Proof. The proof is similar to that of lemma 6.1.1, and uses the fact that if $t_1 \approx t_2$ then

$$\langle t_1, \sigma, \delta \rangle \rightarrow^* \langle \sigma', \delta' \rangle \text{ iff } \langle t_2, \sigma, \delta \rangle \rightarrow^* \langle \sigma', \delta' \rangle$$

and

$$\langle t_1, \sigma, \delta \rangle \uparrow \delta'' \text{ iff } \langle t_2, \sigma, \delta \rangle \uparrow \delta'',$$

for all $\sigma, \sigma' \in Sta$, $\delta, \delta' \in N^*$ and $\delta'' \in N^\infty$. □

The while-loop approximations $W^n(E, t)$ and the family of least fixed point constraints $\Phi$ have the same formal definitions as in section 6.1. We can now prove the main result of this section: there is no fully abstract, least fixed point model of our programming language.

Theorem 6.2.2 There does not exist a $\approx$-fully abstract, $\Phi$-least fixed point, complete ordered algebra.
Proof. Suppose, toward a contradiction, that such an ordered algebra does exist. By part 1 of corollary 5.2.2, there is an \( \Omega \)-least substitutive pre-ordering \( \preceq \) over \( T \) such that \( \approx = \preceq \cap \succeq \) and \( \preceq \) satisfies \( \mathcal{F} \). Let the term \( t \) be

\[
x := 1;
y := 0;
while x \neq 0 do y := y + 1 or x := 0 od;
while y \neq 0 do output x; y := y - 1 od;
\]

so that \( O \cdot t \cdot \sigma = \{ (\bot, 0^n) \mid n \in \omega \} \), where \( 0^n \) is the sequence of zeroes of length \( n \). Let \( t' \) be

\[
x := 0;\ \text{while } x \equiv 0 \text{ do output } x \text{ od},
\]

and define an \( \omega \)-chain \( t''_n \) in \( OT \) by

\[
x := 0; W^n(x \equiv 0, \text{output } x).
\]

Then \( O \cdot t' \cdot \sigma = \{ (\bot, 0^\omega) \} \), where \( 0^\omega \) is the infinite sequence of zeroes, and \( O \cdot t''_n \cdot \sigma = \{ (\bot, 0^n) \} \), for all \( n \in \omega \). Now, \( t \or t' \) is a lub of the \( \omega \)-chain \( t \or t''_n \) in \( (T, \preceq) \), since

\[
(t \or t') \equiv \bigcup \{ t \or t''_n \mid n \in \omega \} \in \overline{\mathcal{F}},
\]

and \( \preceq \) satisfies \( \overline{\mathcal{F}} \). But \( t \or t''_n \approx t \), for all \( n \in \omega \), and thus \( t \or t' \approx t \)—a contradiction. \( \square \)
Chapter 7

Obtaining Fully Abstract Models from Correct Models

In this chapter, we investigate two approaches to obtaining fully abstract models from correct ones. In section 7.1, we use the condition for the existence of inequationally fully abstract models of chapter 5 in order to develop useful necessary and sufficient conditions involving the existence of correct models. In section 7.2, we consider the possibility of collapsing correct models, via continuous homomorphisms, to fully abstract ones. We show that this is not always possible—indeed the natural continuous function model of PCF provides a counterexample—but give a sufficient condition for its possibility. Both of these approaches yield fully abstract models for the languages introduced in chapter 4 and, more generally, for languages whose notions of program ordering and equivalence are defined as abstractions of models using the technique of section 4.1.
7.1 Model-Theoretic Conditions for the Existence of Fully Abstract Models

The following theorem gives two model-theoretic necessary and sufficient conditions for the existence of inequationally fully abstract, least fixed point, complete ordered algebras. Their necessity is obvious; theorem 5.1.4 is used to show their sufficiency. A corollary of this theorem gives two model-theoretic necessary and sufficient conditions for the existence of equationally fully abstract, least fixed point, complete ordered algebras.

Theorem 7.1.1 Let \( \leq \) be an \( \Omega \)-least substitutive pre-ordering over \( T \) and \( \Phi \) be a family of least fixed point constraints. The following conditions are equivalent.

1. A \( \leq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra exists.

2. There is a \( \Phi \)-least fixed point, complete ordered algebra \( A \), together with an inductive pre-ordering \( \leq \) over \( A \), such that

\[
t_1 \leq_s t_2 \text{ iff } M_s t_1 \leq_s M_s t_2,
\]

for all \( t_1, t_2 \in T_s, s \in S \).

3. There is a \( \Phi \)-least fixed point, complete ordered algebra \( A \), together with a continuous function \( h \) from \( A \) to a cpo \( B \), such that

\[
t_1 \leq_s t_2 \text{ iff } h_s(M_s t_1) \sqsubseteq_B h_s(M_s t_2),
\]

for all \( t_1, t_2 \in T_s, s \in S \).

Proof. We show that \( 2 \Rightarrow 1 \), \( 1 \Rightarrow 3 \) and \( 3 \Rightarrow 2 \).

\((2 \Rightarrow 1)\) By theorem 5.1.4, it is sufficient to show that \( \leq \) satisfies \( \overline{\Phi} \). Suppose \( t \equiv \bigcup T' \in \overline{\Phi}_s, s \in S \). By lemma 3.2.7, \( A \) satisfies \( \overline{\Phi} \), and thus \( M_s t = \bigcup M_s T' \). Since \( \sqsubseteq_A \subseteq \leq \), it then follows that \( M_s t \) is an ub of \( M_s T' \) in \( \langle A_s, \leq_s \rangle \), and thus
that \( t \) is an ub of \( T' \) in \( (T_3, \leq_S) \). Suppose \( t'' \) is also an ub of \( T' \) in \( (T_3, \leq_S) \). Then \( M_S t'' \) is an ub of \( M_S T' \) in \( (A_3, \leq_S) \), and, since \( \leq \) is inductive,

\[
M_S t = \bigcup M_S T' \leq M_S t''.
\]

Thus \( t \leq_S t'' \), showing that \( t \) is indeed a lub of \( T' \) in \( (T_3, \leq_S) \).

(1 \( \Rightarrow \) 3) Simply take \( A \) to be a \( \leq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra, and let \( h \) be the identity function from \( A \) to \( B = A \).

(3 \( \Rightarrow \) 2) Let \( \leq \) be \( \leq_h \). Then,

\[
t_1 \leq_S t_2 \quad \text{iff} \quad h_S(M_S t_1) \subseteq_B h_S(M_S t_2)
\]

\[
\text{iff} \quad M_S t_1 \leq_S M_S t_2,
\]

for \( t_1, t_2 \in T_3, s \in S \). \( \square \)

**Corollary 7.1.2** Let \( \approx \) be a congruence over \( T \) and \( \Phi \) be a family of least fixed point constraints. The following conditions are equivalent.

1. \( A \approx \)-fully abstract, \( \Phi \)-least fixed point, complete ordered algebra exists.

2. There is a \( \Phi \)-least fixed point, complete ordered algebra \( A \), together with an inductive pre-ordering \( \leq \) over \( A \), such that

\[
t_1 \approx_S t_2 \quad \text{iff} \quad M_S t_1 (\leq \cap \geq_S) M_S t_2,
\]

for all \( t_1, t_2 \in T_3, s \in S \).

3. There is a \( \Phi \)-least fixed point, complete ordered algebra \( A \), together with a continuous function \( h \) from \( A \) to a cpo \( B \), such that

\[
t_1 \approx_S t_2 \quad \text{iff} \quad h_S(M_S t_1) = h_S(M_S t_2),
\]

for all \( t_1, t_2 \in T_3, s \in S \).
Proof. We show that $2 \Rightarrow 1$, $1 \Rightarrow 3$ and $3 \Rightarrow 2$.

$(2 \Rightarrow 1)$ Define a pre-ordering $\preceq$ over $T$ by

$$t_1 \preceq t_2 \text{ iff } M_S t_1 \leq_S M_S t_2,$$

so that $\preceq$ induces $\simeq$. Then by lemma 2.3.36, $\preceq^C$ is an $\Omega$-least substitutive pre-ordering over $T$, $\preceq^C$ is a unary-substitutive inductive pre-ordering over $A$, and

$$t_1 \preceq^C t_2 \text{ iff } M_S t_1 \leq^C_S M_S t_2,$$

for all $t_1, t_2 \in T_S$, $s \in S$. Furthermore, by lemma 2.2.27, $\preceq^C$ also induces $\simeq$. Thus, by condition 2 of theorem 7.1.1, a $\simeq$-fully abstract, $\Phi$-least fixed point, complete ordered algebra exists.

$(1 \Rightarrow 3)$ Simply take $A$ to be a $\simeq$-fully abstract, $\Phi$-least fixed point, complete ordered algebra, and let $h$ be the identity function from $A$ to $B = A$.

$(3 \Rightarrow 2)$ Simply let $\preceq = \preceq_h$. □

Note the subtlety in the proof that condition 2 implies condition 1 of the corollary: the pre-ordering $\preceq$ is not necessarily substitutive, and thus $\preceq^C$, which also induces $\simeq$, must be used instead.

Condition 2 of theorem 7.1.1 is especially useful since it allows us to conclude that fully abstract models exist for the languages of chapter 4. Consider, e.g., the case of PCF. Let $\Sigma, \mathcal{E}, \Delta, \preceq^\text{PCF}, \leq^\text{PCF}$ and $\simeq^\text{PCF}$ be as in section 4.3. We can apply condition 2, with $\Delta^*, \mathcal{E}, \preceq^\text{PCF}$ and $\leq^\text{PCF}$ substituted for $\Phi, A, \preceq$ and $\leq$, respectively, and conclude that a $\preceq^\text{PCF}$-inequationally fully abstract, $\Delta^*$-least fixed point, complete ordered algebra exists. Such a model is also $\simeq^\text{PCF}$-fully abstract, and from corollary 5.3.4, it follows that a $\simeq^\text{PCF}$-contextually fully abstract, $\Delta$-contextually least fixed point, complete ordered algebra exists. Note, however, that we are still rather a long way from the theorem of [Berry] that an order-extensional fully abstract model of PCF exists.

The proof that a fully abstract model exists for the language TIE of section 4.4 is similar. Furthermore, we can apply condition 2 of theorem 7.1.1 to any
language whose notion of program ordering is defined via lemma 4.1.1.

Condition 3 of corollary 7.1.2 states that there exists a correct, least fixed point model, together with a continuous "full abstraction function", for a programming language. It was suggested in [AptPlotkin] that condition 3 might be weaker than condition 1; the corollary shows that this is false. See the end of section 6.1 for an application of this result.

7.2 Collapsing Correct Models into Fully Abstract Models

Given a correct, least fixed point, complete ordered algebra, it is natural to consider collapsing it, via a continuous homomorphism, into a fully abstract, least fixed point, complete ordered algebra. This, of course, is not always possible, since fully abstract models do not always exist. But, is it always possible when such models do exist? The answer is "no"; in fact neither of the following conditions are sufficient to guarantee that a \( \preceq \)-inequationally correct, \( \Phi \)-least fixed point, complete ordered algebra \( A \) can be continuously collapsed into a \( \preceq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra:

1. \( A \) is inductively reachable, and there exist \( \preceq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebras;

2. \( \preceq \) is related to an inductive pre-ordering \( \leq \) over \( A \) according to condition 2 of theorem 7.1.1, so that \( \preceq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebras exist.

We shall see, however, that the conjunction of these conditions is sufficient.

Theorem 5.4.5 shows that condition 1 is not sufficient, and we now give two proofs of the insufficiency of condition 2. The first is a simple artificial counterexample, whereas the second features the natural continuous function model
$\mathcal{E}$ of PCF. The artificial counterexample was discovered first; subsequently, Plotkin (unpublished) gave a proof that the continuous function model of the combinatorial logic version of PCF cannot be collapsed to a fully abstract model. It is unclear, however, how to extend his proof to our framework, and the proof of the noncollapsibility of $\mathcal{E}$ that is presented below uses a completely different and (at least for us) more perspicuous method.

Lemma 7.2.1 If $\approx$ is a congruence over $T$, $A$ is an algebra, and there does not exist a congruence $\equiv$ over $A$ such that

$$t_1 \approx_S t_2 \text{ iff } M_S t_1 \equiv_S M_S t_2,$$

for all $t_1, t_2 \in T_S, s \in S$, then there does not exist a homomorphism $h$ from $A$ to a $\approx$-fully abstract algebra $B$.

Proof. If such an $h$ does exist then $\equiv_h$ is a congruence over $A$ such that

$$t_1 \approx_S t_2$$

$$\text{iff } h_S(M_AS t_1) = M_{B_S} t_1 = M_{B_S} t_2 = h_S(M_AS t_2)$$

$$\text{iff } M_{A_S} t_1 \equiv_{h_S} M_{A_S} t_2,$$

for all $t_1, t_2 \in T_S, s \in S$. □

Theorem 7.2.2 There is a signature $\Sigma$, an $\Omega$-least substitutive pre-ordering $\preceq$ over $T$, a family of least fixed point constraints $\Phi$, a $\preceq$-inequationally correct, $\Phi$-least fixed point, complete ordered algebra $A$, and a unary-substitutive inductive pre-ordering $\preceq$ over $A$ with the property that for all $t_1, t_2 \in T_S, s \in S$,

$$t_1 \preceq_S t_2 \text{ iff } M_S t_1 \preceq_S M_S t_2,$$

but such that $A$ cannot be collapsed, via a homomorphism, to a $\approx$-fully abstract algebra, where $\approx$ is the congruence induced by $\preceq$. Thus, in particular, $A$ cannot be collapsed, via a continuous homomorphism, to a $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra.

137
Proof. Consider the $\Sigma$, $A$ and $\leq$ from lemma 2.3.34, and let $\Phi = \emptyset$, so that $A$ satisfies $\Phi$. Define a pre-ordering $\leq$ over $T$ by

$$t_1 \leq_S t_2 \text{ iff } M_S t_1 \leq_S M_S t_2,$$

for $t_1, t_2 \in T_S$, $s \in S$. Then by lemma 2.3.37, $\leq$ is an $\Omega$-least substitutive pre-ordering, and $A$ is $\preceq$-inequationally correct, since $\sqsubseteq_A \subseteq \leq$. Let $\approx = \leq \cap \succeq$ and $\equiv = \leq \cap \geq$, so that

$$t_1 \approx_S t_2 \text{ iff } M_S t_1 \equiv_S M_S t_2,$$

for all $t_1, t_2 \in T_S$, $s \in S$. The theorem then follows by lemmas 7.2.1 and 2.3.34.

For the following theorem, let $\Sigma$, $\mathcal{E}$, $\Delta$, $\preceq_{PCF}$, $\approx_{PCF}$, portest$_1$, and PORT be as in section 4.3.

Theorem 7.2.3 $\mathcal{E}$ cannot be collapsed, via a homomorphism, to a $\approx_{PCF}$-fully abstract algebra. In particular, $\mathcal{E}$ cannot be collapsed, via a continuous homomorphism, to a $\preceq_{PCF}$-inequationally fully abstract, $\Delta^*\text{-least fixed point, complete ordered algebra.}$

Proof. Suppose, toward a contradiction, that there exists a congruence $\equiv$ over $\mathcal{E}$ such that

$$t_1 \approx_{PCF} t_2 \text{ iff } M_S t_1 \equiv_S M_S t_2,$$

for all $t_1, t_2 \in T_S$, $s \in S$. Let $s'$ be the sort $(\text{bool} \rightarrow \text{bool} \rightarrow \text{bool}) \rightarrow \text{nat}$. Then

$$\text{portest}_1 \approx_{PCF} \text{portest}_2$$

$$\Rightarrow M_{s'} \text{portest}_1 \equiv_{s'} M_{s'} \text{portest}_2$$

$$\Rightarrow M_{\text{nat}} 1 = (M_{s'} \text{portest}_1) \cdot \text{PORT} \equiv_{\text{nat}} (M_{s'} \text{portest}_2) \cdot \text{PORT} = M_{\text{nat}} 2$$

$$\Rightarrow 1 \approx_{\text{nat}} 2,$$

which is a contradiction. The theorem then follows by lemma 7.2.1.
A consequence of this theorem is that \( \mathcal{E} \) and \( \leq_{\text{PCF}} \) (also of section 4.3) provide an alternative proof of lemma 2.3.34; in particular, \( \leq_{\text{PCF}} \) is unary-substitutive but not substitutive.

Next we show that, whenever they are possible, continuous collapses can be carried out using the inductive quotienting construction of section 2.4.

**Lemma 7.2.4** Let \( \leq \) be an \( \Omega \)-least substitutive pre-ordering over \( T \), \( \Phi \) be a family of least fixed point constraints, \( A \) be a \( \Phi \)-least fixed point, complete ordered algebra, and \( \leq \) be a substitutive inductive pre-ordering over \( A \) such that

\[
t_1 \leq_S t_2 \iff M_\Phi t_1 \leq_S M_\Phi t_2,
\]

for all \( t_1, t_2 \in T \), \( s \in S \). Then \( A \) can be collapsed, via the continuous homomorphism \( q_t \), to the \( \leq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra \( A//\leq \).

**Proof.** For the inequational full abstraction of \( A//\leq \), let \( t_1, t_2 \in T \), \( s \in S \). Then,

\[
t_1 \leq_S t_2 \iff M_\Phi t_1 \leq_S M_\Phi t_2 \iff q_t(M_\Phi t_1) \leq_{(A//\leq)_S} q_t(M_\Phi t_2) \iff M_{(A//\leq)_S} t_1 \leq_{(A//\leq)_S} M_{(A//\leq)_S} t_2.
\]

To see that \( A//\leq \) satisfies \( \Phi \), let \( t \equiv \bigcup T' \in \Phi_S \), \( s \in S \). Then,

\[
M_{(A//\leq)_S} t = q_t(M_\Phi t) = q_t \bigcup M_\Phi T' = \bigcup M_\Phi T' = \bigcup M_{(A//\leq)_S} T',
\]

as required. \( \Box \)
Lemma 7.2.5 Let $A$ be a $\Phi$-least fixed point, complete ordered algebra. The following two conditions are equivalent.

1. There is a $\preceq$-inequationally fully abstract, $\Phi$-least fixed point, complete ordered algebra $B$, together with a continuous homomorphism $h: A \rightarrow B$.

2. There is a substitutive inductive pre-ordering $\leq$ over $A$ such that for all $t_1, t_2 \in T_S$, $s \in S$,

$$t_1 \leq_s t_2 \text{ iff } M_s t_1 \leq_s M_s t_2.$$ 

Proof. For $1 \Rightarrow 2$, let $\leq$ be $\leq_h$. Then $\leq$ is a substitutive inductive pre-ordering over $A$, and for $t_1, t_2 \in T_S$, $s \in S$,

$$t_1 \leq_s t_2 \text{ iff } M_S t_1 \subseteq_B M_S t_2 \text{ iff } h_S(M_A t_1) \subseteq_B h_S(M_A t_2) \text{ iff } M_A t_1 \leq_s M_A t_2.$$ 

For $2 \Rightarrow 1$, simply apply lemma 7.2.4. □

Now we are able to give a sufficient condition for the possibility of collapsing inductively reachable, inequationally correct models, via continuous homomorphisms, to inequationally fully abstract models, and, more generally, for collapsing the reachable inductive subalgebras of inequationally correct models to inequationally fully abstract models.

Theorem 7.2.6 Suppose $\preceq$ is an $\Omega$-least substitutive pre-ordering over $T$, $\Phi$ is a family of least fixed point constraints, $A$ is a $\Phi$-least fixed point, complete ordered algebra, and $\leq$ is an inductive pre-ordering over $A$ with the property that

$$t_1 \leq_s t_2 \text{ iff } M_S t_1 \leq_s M_S t_2,$$

for all $t_1, t_2 \in T_S$, $s \in S$. Let $\leq'$ be the restriction of $\leq^c$ to $R(A)$. Then $R(A)$ can be collapsed, via the continuous homomorphism $q$, to the inductively
reachable, \( \leq \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra \( R(\mathcal{A})/\leq' \).

**Proof.** By lemmas 2.3.14, 2.3.12 and 2.3.35, \( \leq' \) is a substitutive inductive pre-ordering over \( R(\mathcal{A}) \), and, by lemma 2.3.37,

\[
t_1 \leq_5 t_2 \text{ iff } M_S t_1 \leq^c_5 M_S t_2 \text{ iff } M_S t_1 \leq'_5 M_S t_2,
\]

for all \( t_1, t_2 \in T_S, s \in S \). Thus, by lemma 7.2.4, \( R(\mathcal{A}) \) can be collapsed, via the continuous homomorphism \( q_t \), to the \( \leq' \)-inequationally fully abstract, \( \Phi \)-least fixed point, complete ordered algebra \( R(\mathcal{A})/\leq' \), and, by lemma 2.4.17, \( R(\mathcal{A})/\leq' \) is inductively reachable. \( \square \)

Note the following special cases of theorem 7.2.6. If \( \leq \) is already unary-substitutive then \( R(\mathcal{A}) \) can be collapsed to \( R(\mathcal{A})/\leq' \), where \( \leq' \) is simply the restriction of \( \leq \) to \( R(\mathcal{A}) \). If \( \leq \) is unary-substitutive and \( \mathcal{A} \) is inductively reachable then \( \mathcal{A} \) itself can be collapsed to \( \mathcal{A}/\leq \), since \( \leq \) is in fact substitutive.

Theorem 7.2.6 can be immediately applied to the languages of chapter 4 and, more generally, to languages whose notions of program ordering are defined via lemma 4.1.1. Consider, e.g., the case of PCF. Let \( \Sigma, \mathcal{E}, \Lambda, \leq_{\text{PCF}}, \leq'_{\text{PCF}} \) and \( \approx_{\text{PCF}} \) be as in section 4.3. Then \( R(\mathcal{E}) \) can be collapsed, via the continuous homomorphism \( q_t \), to the inductively reachable, \( \leq_{\text{PCF}} \)-inequationally fully abstract, \( \Lambda^* \)-least fixed point, complete ordered algebra \( R(\mathcal{E})/\leq' \), where \( \leq' \) is the restriction of \( \leq_{\text{PCF}} \) to \( R(\mathcal{E}) \). Furthermore, \( R(\mathcal{E})/\leq' \) is also \( \approx_{\text{PCF}} \)-fully abstract, and thus, by theorems 5.3.1 and 5.3.2, we can conclude that \( R(\mathcal{E})/\leq' \) is \( \approx_{\text{PCF}} \)-contextually fully abstract and \( \Lambda \)-contextually least fixed point. It would be interesting to know whether or not \( I(\leq_{\text{PCF}}, \Lambda^*) \), the initial \( \leq_{\text{PCF}} \)-inequationally fully abstract, \( \Lambda^* \)-least fixed point, complete ordered algebra and \( R(\mathcal{E})/\leq' \) are order-isomorphic.

We can also prove the following equational variant of theorem 7.2.6.

**Corollary 7.2.7** Suppose \( \approx \) is a congruence over \( \mathcal{T} \), \( \Phi \) is a family of least fixed point constraints, \( \mathcal{A} \) is a \( \Phi \)-least fixed point, complete ordered algebra, and \( h \) is...
a continuous function from $A$ to a cpo $B$, such that

$$t_1 \approx t_2 \text{ iff } h_s(M_s t_1) = h_s(M_s t_2),$$

for all $t_1, t_2 \in T_s$, $s \in S$. Then $R(A)$ can be collapsed, via the continuous homomorphism $q_t$, to the inductively reachable, $\approx$-fully abstract, $\Phi$-least fixed point, complete ordered algebra $R(A)/\leq'$, where $\leq'$ is the restriction of $(\leq_h)^c$ to $R(A)$.

**Proof.** Define a pre-ordering $\leq$ over $T$ by

$$t_1 \leq t_2 \text{ iff } M_s t_1 (\leq_h)^c M_s t_2,$$

so that $\leq$ induces $\approx$ (but $\leq$ may not be substitutive!). Then, by lemma 2.3.36, $\leq^c$ is an $\Omega$-least substitutive pre-ordering over $T$, $(\leq_h)^c$ is a unary-substitutive inductive pre-ordering over $A$,

$$t_1 \leq^c t_2 \text{ iff } M_s t_1 (\leq_h)^c M_s t_2,$$

for all $t_1, t_2 \in T_s$, $s \in S$, and, by lemma 2.2.27, $\leq^c$ also induces $\approx$. The desired result follows by theorem 7.2.6. □
Chapter 8

Conclusion

In the preceding chapters, we have developed a theory of fully abstract models of programming languages and applied this theory to several programming languages. On the basis of these examples, it seems likely that the theory will yield proofs of the existence or nonexistence of fully abstract models of a wide variety of programming languages. I expect, for example, that the existence of fully abstract models for the Algol-like language of [Halpern] can be shown using the methods of chapter 7, and that the nonexistence of fully abstract models of the fair parallel programming language of [Plotkin2] can be shown using the techniques of chapter 6. In this final chapter, we consider the theory's limitations and the corresponding possibilities for further research.

The cornerstone of the theory is its class of models: complete ordered algebras. This was a natural and rewarding choice, but there are many other important classes of models, narrower and wider, that should also be studied. Examples include: universal algebras whose carriers are cpo's with additional order-theoretic structure, e.g., consistently-complete $\omega$-algebraic cpo's; models based on weaker notions of continuity [Plotkin2]; categorical models [Lehmann] [Abramsky2]; and models definable in particular metalanguages (and thus, in a formal sense, natural). The extension of the theory to these classes of models will probably involve the development of new quotienting and completion
An essential feature of the theory is the inclusion of the undefined constants $\Omega$ in all signatures, and the corresponding requirements that they be interpreted as $\perp$ in models, and be least elements in notions of program ordering. Unfortunately, this feature limits the applicability of the theory. There are programming languages, such as the parallel programming language of [HennessyPlotkin1], whose notions of program ordering do not have least elements, and, thus, whose inequationally correct models cannot have definable least elements. There may even be naturally occurring languages for which equationally fully abstract models exist, but such that there do not exist such models with definable least elements. It is thus desirable to develop a theory in which the undefined constants are not required. This would be a radical departure from the current theory, however, and it is unclear how to proceed.

As we indicated in chapter 4, our treatment of programming languages with block structure, such as PCF and TIE, is only partially satisfactory, for the following reasons. First, we are unable to construct environment models for these languages, i.e., models that have identifier environments as formal components. Second, the theory is not directly applicable to notions of program ordering and equivalence that are defined in terms of the behaviour of closed terms of program sort, as opposed to all such terms. Third, there apparently do not exist suitable families of least fixed point constraints for certain languages with recursion, such as the call-by-value version of TIE. Removing the first of these defects, and giving program identifiers and their scopes formal status in signatures, is the first step toward the removal of the second and third defects. See the end of section 4.4 for more discussion of these points.

Notions of program equivalence are often defined as abstractions of operational semantics, as with the languages of chapter 6. Unfortunately, the condition for the existence of inequationally fully abstract models of section 7.1, which was the basis for our positive results, is model-theoretic in nature and is
expressed in terms of program orderings instead of equivalences. It would thus be useful to develop conditions for the existence of fully abstract models that are directly applicable to operationally defined program equivalences.

In section 7.2, we gave useful sufficient conditions for the possibility of collapsing inductively reachable correct models, via continuous homomorphisms, to fully abstract models, and, more generally, for collapsing the reachable inductive subalgebras of correct models to fully abstract models. We also showed that it is not always possible to collapse correct models in such a way. Useful sufficient conditions for the possibility of collapsing non-inductively reachable correct models should be developed.

In section 5.4, we began the study of various categories of correct and fully abstract models, proving theorems concerning the existence and nonexistence of initial and terminal objects, respectively. Much remains to be learned about the structure of these categories, and thus this study should continue. In particular, it would be nice to resolve conjecture 5.4.6.
Table of Notation

In the following lists, each symbol is followed by the number of its definition.

Many Sorted Algebras: $S, \Sigma, s_1 \times \cdots \times s_n \to s', \Omega \ (2.2.1); A, A \ (2.2.2); A \subseteq B \ (2.2.3); f A \ (2.2.4); \tau_E \ (2.2.5); M_\Lambda \ (2.2.7); \equiv_f \ (2.2.10); \equiv_A \ (2.2.11); \Sigma(X), \tau_E(X) \ (2.2.14); V_E \ (2.2.16); c[v_1, \ldots, v_n] \ (2.2.17); R^c \ (2.2.25).

Ordered Algebras: $D \rightarrow_m E, D \rightarrow_c E \ (2.3.6); A, A \ (2.3.7); A \leq B \ (2.3.8); \subseteq_A \ (2.3.10); \leq_f \ (2.3.11); \leq^0 \ (2.3.15); O\tau_E \ (2.3.18); O\tau_E(X) \ (2.3.20); [B] \ (2.3.26); R(A) \ (2.3.28).

Quotienting and Completion Constructions: $\Gamma \ (2.4.1); A^\Gamma, g^\Gamma, em \ (2.4.10); A/\leq, g/\leq, qt_\leq \ (2.4.12); A/\leq, g/\leq, qt_\leq \ (2.4.16).

Least Fixed Point Constraints: $\Phi, t\equiv\bigcup T' \ (3.2.1); \Phi \ (3.2.3); \Delta, c\equiv v_1, \ldots, v_n \bigcup C' \ (3.2.8); \Delta^* \ (3.2.10).

Term Model Construction: $I(\leq, \Phi) \ (5.1.2).$
Bibliography


Mac Lane, S. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

Markowsky, G. Chain-complete posets and directed sets with applications. *Algebra Univ.* 6, 1976.


