A Domain Equation For Bisimulation

Samson Abramsky

Department of Computing

Imperial College of Science, Technology and Medicine

To appear in *Information and Computation*

January 2, 1990

Abstract

Some basic topics in the theory of concurrency are studied from the point of view of denotational semantics, and particularly the "Domain theory in logical form" developed by the author. A domain of synchronization trees is defined by means of a recursive domain equation involving the Plotkin powerdomain. The logical counterpart of this domain is described, and shown to be related to it by Stone duality. The relationship of this domain logic to the standard Hennessy-Milner logic for transition systems is studied; the domain logic can be seen as a rational reconstruction of Hennessy-Milner logic from the standpoint of a very general and systematic theory. Finally, a denotational semantics for SCCS based on the domain of synchronization trees is given, and proved fully abstract with respect to bisimulation.

1 Introduction

Our aim in this paper is to treat some basic topics in the theory of concurrency from the point of view of denotational semantics and "domain logic" in the sense of [Abr87b, Abr87a, Abr88a]. This will serve as a major case study for the general theory developed in [Abr87a]. Our aim is not only to exemplify the general theory, but to *apply* it in order to shed some new light on concurrency. In particular, we shall study *bisimulation* [Par81, Mil83, HM85]. This notion has emerged as one of the more stable and mathematically natural concepts to have been formulated in the study of concurrency over the past decade. It is commonly accepted as the *finest* extensional or behavioural equivalence on processes one would want to impose. To date, bisimulation has been studied almost exclusively from the operational and logical points of view. Our aim is to show that this notion can be captured elegantly in the setting of domain theory, using Plotkin's powerdomain construction [Plo76]. Moreover, we shall make extensive use of the logical form of domain theory developed in [Abr87b, Abr87a, Abr88a]. Thus our motivation can be summarised as follows:

- To show that more can be done in the sphere of concurrency using domain-theoretic and denotational methods than seems to be commonly realised.
- To analyze the apparently *ad hoc* and "application oriented" notions of bisimulation over labelled transition systems and Hennessy-Milner logic by means of the general, mathematically basic, and "reusable" notions of domain theory, specifically type constructions and the solution of recursive domain equations.
- To form part of our general programme of connecting

  1. Domain theory and operational notions of observability
  2. Denotational semantics and program logics.

  This programme is made systematic by using the information conveyed in the syntactic description of domains by type expressions. It can be argued that a full domain-theoretic analysis of some computational situation is only obtained when we have written down an explicit type expression, rather than using some *ad hoc* construction of a cpo. At any rate, the benefits which flow from having such a description are very considerable. Using the ideas developed in [Abr87b, Abr87a, Abr88a], we can derive a propositional theory from the type expression, and use this to explore the "observational logic" of the computational situation.

We now summarise the further contents of the paper. After reviewing some basic notions on transition systems etc., we introduce a domain of synchronisation trees defined by means of a domain equation (recursive type expression). Then we present a domain logic for transition systems, which is derived from this domain equation in the sense of [Abr87b, Abr87a, Abr88a]. The main result of section 4 is that the finitary part of this logic is the Stone dual of our domain of synchronisation trees.

In section 5, we present a number of applications of this logic. It is shown to be equivalent to Hennessy-Milner logic in the infinitary case, and hence to characterise bisimulation. In the finitary case, it more powerful than Hennessy-Milner logic, and we obtain a more satisfactory characterisation result for it; namely, it is shown to characterise the "finitary part" of bisimulation for *all* transition systems.

We also develop an extension of Hennessy-Milner logic which is equivalent to the finitary domain logic. The infinitary domain logic is then used to *axiomatize* a suitable notion of "finitary transition system". These systems are shown indeed to be finitary in a strong sense — their bisimulation preorders are algebraic. Finally, the domain of synchronisation trees (i.e. the spectral space of the logic) is shown to be finitary *qua* transition system, and moreover to be *final* in a suitable category of such systems. This yields a syntax-free "universal semantics" for transition systems, which is fully abstract with respect to bisimulation.

In section 6, we give a conventional (syntax-directed) denotational semantics for the concurrent calculus SCCS [Mil83], based on our domain of synchronisation trees. A full abstraction result is proved for this semantics; as a by-product, our domain is shown to be isomorphic to Hennessy's term model [Hen81].

2 Transition Systems and Related Notions

We begin with the basic notion of a labelled transition system (with divergence), which abstracts from the operational semantics of many concurrent calculi.

Definition 2.1 A *transition system* is a structure

$$(\mathrm{Proc}, \mathsf{Act}, \rightarrow, \uparrow)$$

where:

- Proc is a set of *processes* or *agents*.
- Act is a set of atomic *actions* or *experiments*.
- $\rightarrow \subseteq \mathrm{Proc} \times \mathsf{Act} \times \mathrm{Proc}$ (notation: $p \xrightarrow{a} q$).
- $\uparrow \subseteq \mathrm{Proc}$ (notation: $p\uparrow$).

We write

$$p\downarrow \equiv \neg(p\uparrow).$$

We read $p \xrightarrow{a} q$ as "p has the capability to do a and become (i.e. change state to) q"; $p\uparrow$ as "p may diverge"; and $p\downarrow$ as "p definitely converges". We define

$$\mathsf{sort}(p) \equiv \{a \in \mathsf{Act} \mid \exists q, r.\, p \rightarrow^{\star} q \xrightarrow{a} r\}$$

where $p \rightarrow q \equiv \exists a \in \mathsf{Act}.\, p \xrightarrow{a} q$, and $\rightarrow^{\star}$ is the reflexive, transitive closure of $\rightarrow$.

We now define a number of finiteness conditions on transition systems:

image-finiteness	$\forall p \in \mathrm{Proc}, a \in \mathsf{Act}.\, \{q \mid p \xrightarrow{a} q\}$ is finite.
sort-finiteness	$\forall p \in \mathrm{Proc}.\, \mathsf{sort}(p)$ is finite.
finite-branching	$\forall p \in \mathrm{Proc}.\, \{q \mid p \rightarrow q\}$ is finite.
initials-finiteness	$\forall p \in \mathrm{Proc}.\, \{a \in \mathsf{Act} \mid \exists q.\, p \xrightarrow{a} q\}$ is finite.

Each of these properties has a weak form, obtained by making it conditional on convergence. For example:

weak image-finiteness $\quad \forall p \in \mathrm{Proc}, a \in \mathsf{Act}.\, p\downarrow \;\Rightarrow\; \{q \mid p \xrightarrow{a} q\}$ is finite.

We now introduce a particularly useful source of examples for transition systems, the *synchronisation trees*. Given a set Act of actions, $\mathsf{ST}_\infty(\mathsf{Act})$, the synchronisation trees over Act, are defined as the (proper) class of infinitary terms generated by the following inductive definition:

$$\frac{\{a_i \in \mathsf{Act}, t_i \in \mathsf{ST}_\infty(\mathsf{Act})\}_{i\in I}}{\sum_{i\in I} a_i t_i \; [+\Omega] \in \mathsf{ST}_\infty(\mathsf{Act})} \tag{1}$$

where $[+\Omega]$ means optional inclusion of Ω as a summand (i.e. there are really two clauses in this definition). We write

$$\begin{aligned} \mathbb{O} &\equiv \sum_{i\in\varnothing} a_i t_i \\ \Omega &\equiv \sum_{i\in\varnothing} a_i t_i + \Omega. \end{aligned}$$

The subclass of terms formed using only finite sums is denoted $\mathsf{ST}_\omega(\mathsf{Act})$. Given a synchronisation tree t formed according to (1), we stipulate:

- $t\uparrow$ iff Ω is included as a summand.

- $t \xrightarrow{a_i} t_i$ for each summand $a_i t_i$ $(i \in I)$.

This defines a (large) transition system $(\mathsf{ST}_\infty(\mathsf{Act}), \mathsf{Act}, \rightarrow, \uparrow)$; restriction to a subset of synchronisation trees yields a small transition system. In particular, by choosing a canonical system of representatives for $\mathsf{ST}_\omega(\mathsf{Act})$ which is closed under subtrees we obtain a countable transition system of finite synchronisation trees, which by abuse of notation we refer to also as $\mathsf{ST}_\omega(\mathsf{Act})$.

We are now ready to introduce the main concept we will study.

Definition 2.2 ([Par81, Mil80, Mil81]) A relation $R \subseteq \mathrm{Proc} \times \mathrm{Proc}$ is a *partial bisimulation* if, for all $p, q \in \mathrm{Proc}$:

$$
\begin{array}{rcl}
pRq & \Longrightarrow & \forall a \in \mathsf{Act}. \\
& & \bullet \;\; p \xrightarrow{a} p' \;\; \Longrightarrow \;\; \exists q'.\, q \xrightarrow{a} q' \;\&\; p'Rq' \\
& & \bullet \;\; p\downarrow \;\; \Longrightarrow \;\; q\downarrow \;\&\; [q \xrightarrow{a} q' \;\Rightarrow\; \exists p'.\, p \xrightarrow{a} p' \;\&\; p'Rq'].
\end{array}
$$

We write

$$p \lesssim^B q \equiv \exists R.\, R \text{ is a partial bisimulation and } pRq.$$

For an alternative description of $\lesssim^B$, let $\mathsf{Rel}(\mathrm{Proc})$ be the set of all binary relations over Proc; this is a complete lattice under set inclusion. Now define

$$F : \mathsf{Rel}(\mathrm{Proc}) \rightarrow \mathsf{Rel}(\mathrm{Proc})$$

$$
\begin{array}{rcl}
F(R) & = & \{(p,q) \mid \forall a \in \mathsf{Act}. \\
& & \bullet\, p \xrightarrow{a} p' \;\Rightarrow\; \exists q'.\, q \xrightarrow{a} q' \;\&\; p'Rq' \\
& & \bullet\, p\downarrow \;\Rightarrow\; q\downarrow \;\&\; [q \xrightarrow{a} q' \;\Rightarrow\; \exists p'.\, p \xrightarrow{a} p' \;\&\; p'Rq']\}.
\end{array}
$$

Clearly, R is a partial bisimulation iff $R \subseteq F(R)$, i.e. R is a *pre-fixed point of* F. Since F is monotone, by Tarski's Theorem it has a maximal fixpoint, given by $\bigcup\{R \mid R \subseteq F(R)\}$, i.e. $\lesssim^B$. Thus $\lesssim^B$ is itself a partial bisimulation, and evidently the largest one. Moreover, it is reflexive and transitive; the corresponding equivalence is denoted $\sim^B$.

We can also describe $\lesssim^B$ more explicitly, in terms of iterations of F. We define relations $\lesssim_\alpha$, $(\alpha \in \mathsf{Ord})$ (the class of ordinals), by the following ordinal recursion:

- $p \lesssim_0 q$ always (i.e. $\lesssim_0 = \mathrm{Proc} \times \mathrm{Proc}$, the top element in the lattice $\mathsf{Rel}(\mathrm{Proc})$).

- $p \lesssim_{\alpha+1} q$ iff

$$
\begin{array}{l}
\forall a \in \mathsf{Act}. \\
\bullet\, p \xrightarrow{a} p' \;\; \Longrightarrow \;\; \exists q'.\, q \xrightarrow{a} q' \;\&\; p' \lesssim_\alpha q' \\
\bullet\, p\downarrow \;\; \Longrightarrow \;\; q\downarrow \;\&\; [q \xrightarrow{a} q' \;\Rightarrow\; \exists p'.\, p \xrightarrow{a} p' \;\&\; p' \lesssim_\alpha q'].
\end{array}
$$

  (i.e. $\lesssim_{\alpha+1} = F(\lesssim_\alpha)$).

- For limit λ, $p \lesssim_\lambda q$ iff $\forall \alpha < \lambda.\, p \lesssim_\alpha q$ (i.e. $\lesssim_\lambda = \bigcap_{\alpha<\lambda} \lesssim_\alpha$).

This sequence of relations is decreasing, and bounded below by $\precsim^B$; i.e. for all α

$$\precsim_\alpha \supseteq \precsim_{\alpha+1} \supseteq \precsim^B .$$

For any (small) transition system the sequence is eventually stationary; for some λ, for all $\alpha > \lambda$, $\precsim_\alpha = \precsim_\lambda$. The least ordinal λ for which this holds is called the *closure ordinal* [Mos74]; and we have $\precsim_\lambda = \precsim^B$. Note that each $\precsim_\alpha$ is relexive and transitive.

The relations $\precsim^B$ and $\sim^B$ have been defined in the context of a given transition system. However, we frequently want to use them to compare processes from different transition systems. This is easily accomplished by forming the disjoint union of the two systems, and then using $\precsim^B$ as defined above. In the sequel, we will do this without further comment.

We now introduce a program logic due to Hennessy and Milner [HM85]. The idea is to obtain a characterisation of $\precsim^B$ in terms of a suitable notion of *property* of process; $p \precsim^B q$ iff every property satisfied by p is satisfied by q.

Definition 2.3 Given a set of actions Act, the language $\mathrm{HML}_\infty(\mathsf{Act})$ (we henceforth elide the parameter Act) is defined by the following inductive clauses:

$$\frac{a \in \mathsf{Act},\ \phi \in \mathrm{HML}_\infty}{[a]\phi, \langle a \rangle \phi \in \mathrm{HML}_\infty}$$

$$\frac{\phi_i \in \mathrm{HML}_\infty \ (i \in I)}{\bigwedge_{i \in I} \phi_i, \bigvee_{i \in I} \phi_i \in \mathrm{HML}_\infty}$$

In particular, we write:

$$t \equiv \bigwedge_{i \in \varnothing} \phi_i, \quad f \equiv \bigvee_{i \in \varnothing} \phi_i.$$

We use the subscript ∞ to indicate the presence of infinite conjunctions and disjunctions. We write HML_ω for the sublanguage obtained by restricting the formation rules to finite conjunctions and disjunctions.

We now define a satisfaction relation $\models \subseteq \mathrm{Proc} \times \mathrm{HML}_\infty$.

$$\begin{array}{lcl}
p \models \bigwedge_{i \in I} \phi_i & \equiv & \forall i \in I.\, p \models \phi_i \\
p \models \bigvee_{i \in I} \phi_i & \equiv & \exists i \in I.\, p \models \phi_i \\
p \models \langle a \rangle \phi & \equiv & \exists q.\, p \xrightarrow{a} q \ \&\ q \models \phi \\
p \models [a]\phi & \equiv & p\!\downarrow \ \&\ [\forall q.\, p \xrightarrow{a} q \implies q \models \phi].
\end{array}$$

We write

$$\mathrm{HML}_\infty(p) \equiv \{\phi \in \mathrm{HML}_\infty : p \models \phi\}$$

plus obvious variations on this notation.

We define two useful assignments of ordinals to formulas in HML_∞, the *modal depth*:

$$\begin{array}{lclcl}
\mathsf{md}(\bigwedge_{i \in I} \phi_i) & \equiv & \mathsf{md}(\bigvee_{i \in I} \phi_i) & \equiv & \sup\{\mathsf{md}(\phi_i) : i \in I\} \\
\mathsf{md}([a]\phi) & \equiv & \mathsf{md}(\langle a \rangle \phi) & \equiv & \mathsf{md}(\phi) + 1
\end{array}$$

and the *height*:

$$\begin{array}{lclcl}\mathsf{ht}(\bigwedge_{i\in I}\phi_i) & \equiv & \mathsf{ht}(\bigvee_{i\in I}\phi_i) & \equiv & \sup\{\mathsf{ht}(\phi_i) : i\in I\}+1 \\ \mathsf{ht}([a]\phi) & \equiv & \mathsf{ht}(\langle a\rangle\phi) & \equiv & \mathsf{ht}(\phi)+1.\end{array}$$

We define $\mathsf{sort}(\phi)$ to be the set of action symbols which occur in ϕ.

Now given a set $A \subseteq \mathsf{Act}$ and an ordinal λ, we define a sublanguage of HML_∞:

$$\mathrm{HML}_\infty^{(A,\lambda)} = \{\phi \in \mathrm{HML}_\infty : \mathsf{sort}(\phi) \subseteq A \;\&\; \mathsf{md}(\phi) \leq \lambda\}.$$

We are now ready to prove a generalised and strengthened version of the Modal Characterisation Theorem [Mil81, Mil85, HM85].

Theorem 2.4 (Modal Characterisation Theorem) *Suppose that* $A \subseteq \mathsf{Act}$ *satisfies*

$$\mathsf{sort}(p) \cup \mathsf{sort}(q) \subseteq A \neq \varnothing;$$

then

$$p \sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\lambda q \iff \mathrm{HML}_\infty^{(A,\lambda)}(p) \subseteq \mathrm{HML}_\infty^{(A,\lambda)}(q).$$

As an immediate consequence we obtain

$$p \sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}^B q \iff \mathrm{HML}_\infty(p) \subseteq \mathrm{HML}_\infty(q).$$

PROOF. The left-to-right implication is proved by induction on λ. The cases for $\lambda = 0$, λ a limit ordinal are trivial. For $\lambda = \alpha + 1$, we argue by induction on $\mathsf{ht}(\phi)$. The cases for $\bigwedge_{i\in I}\phi_i$, $\bigvee_{i\in I}\phi_i$ are trivial. Suppose $p \models \langle a\rangle\phi$. Then for some p', $p \xrightarrow{a} p'$ and $p' \models \phi$. Since $p \sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\lambda q$, for some q', $q \xrightarrow{a} q'$ and $p' \sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\alpha q'$. By the outer induction hypothesis, $q' \models \phi$, hence $q \models \langle a\rangle\phi$, as required. The case for $[a]\phi$ is similar.

For the converse, we argue by induction on λ. Suppose $p \not\sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\lambda q$: we must find $\phi \in \mathrm{HML}_\infty^{(A,\lambda)}(p) \setminus \mathrm{HML}_\infty^{(A,\lambda)}(q)$.

Case 1: $p \xrightarrow{a} p'$ and for all q', $q \xrightarrow{a} q'$ implies $p' \not\sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\alpha q'$ for some $\alpha < \lambda$. By induction hypothesis, for each such q' there is $\phi_{q'} \in \mathrm{HML}_\infty^{(A,\alpha)}(p') \setminus \mathrm{HML}_\infty^{(A,\alpha)}(q')$. Now take

$$\phi = \langle a\rangle \bigwedge\{\phi_{q'} : q \xrightarrow{a} q'\}.$$

Case 2: $p\!\downarrow$ and $q\!\uparrow$. Take $\phi \equiv [a]t$, for any $a \in A$.

Case 3: $p\!\downarrow$, $q\!\downarrow$, $q \xrightarrow{a} q'$, and for all p', $p \xrightarrow{a} p'$ implies $p' \not\sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\alpha q'$ for some $\alpha < \lambda$. Defining $\phi_{p'}$ analogously to Case 1,

$$\phi = [a]\bigvee\{\phi_{p'} : p \xrightarrow{a} p'\}. \quad \blacksquare$$

The reader familiar with infinitary logic will recognise the strong similarity between this result and Karp's Theorem [Bar75]. Similar remarks apply to "Master Formula Theorems" as in [Rou85], *vis a vis* the Scott Isomorphism Theorem [Bar75].

Note that, if A is a finite set and λ a finite ordinal, then (up to logical equivalence) $\mathrm{HML}_\infty^{(A,\lambda)}$ is finite. It follows easily from this observation that each formula in $\mathrm{HML}_\infty^{(A,\lambda)}$ is equivalent to one in $\mathrm{HML}_\omega^{(A,\lambda)}$. Hence as a corollary to the Characterisation Theorem we obtain

Theorem 2.5 *[Abr87c] If the transition system is sort-finite, then*

$$p \sqsubseteq\!\!\!\!\!\raise{-0.6ex}{\sim}_\omega q \iff \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q).$$

Moreover, we have the following result from [HM85]:

Theorem 2.6 *If the transition system is image-finite, then*

$$(i) \quad \underset{\sim}{\sqsubset}_\omega = \underset{\sim}{\sqsubset}^B$$
$$(ii) \quad p \underset{\sim}{\sqsubset}_\omega q \iff \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q).$$

Unfortunately, if unguarded recursion is allowed in any of the standard concurrent calculi (SCCS, CCS, CSP, etc.) they are neither image-finite nor sort-finite (though sort-finiteness may be regained e.g. for CCS by imposing fairly mild restrictions on the relabelling operators). Thus the last two Theorems cannot be applied. To see how weak finitary Hennessy-Milner logic is when the set of actions is infinite, consider the following **Example.**

$$\begin{aligned} p &\equiv a\mathbb{O} + \Omega \\ q &\equiv \sum_{n \in \omega} ab_n\mathbb{O} + \Omega \end{aligned}$$

where we assume $b_m \neq b_n$ for $m \neq n$. Now $p \not\underset{\sim}{\sqsubset}_2 q$, but we have

Proposition 2.7 $\mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q)$.

In order to prove this Proposition we need a lemma.

Lemma 2.8 *Every formula in* $\mathrm{HML}_\omega(\mathbb{O})$ *is satisfied by cofinitely many of the* $b_n\mathbb{O}$.

PROOF. By induction on formulas in $\mathrm{HML}_\omega(\mathbb{O})$. For conjunctions and disjunctions, the intersection and union of finitely many cofinite sets are cofinite. (It is the case for conjunction which necessitates the strength of statement of the Lemma). The case for $\langle b \rangle \phi$ is vacuous. For $[b]\phi$, cofinitely many (in fact, all but at most one) of the $b_n\mathbb{O}$ do not have a b-action, hence satisfy $[b]\phi$. $\blacksquare$

The Proposition can now be proved by induction on formulas in HML_ω. The only non-trivial case is $\langle a \rangle \phi$, which follows from the Lemma.

The deficiency of Hennessy-Milner logic illustrated by this example is disturbing, because processes generated by a finitary calculus (including p and q above) should be adequately modelled by a finitary semantics and logic. This suggests that Hennessy-Milner logic is not quite right as it stands.

3 A Domain Equation for Synchronisation Trees

In this section, we shall define a domain of synchronisation trees, and establish some of its basic properties. We shall assume some familiarity with Domain theory; a suitable reference is [Plo81]. Since our definitions will use the Plotkin powerdomain [Plo76], we need to work in a category which is closed under this construction. This means that we cannot use **SDom**, the category of Scott domains. Instead, we will use **SFP** [Plo76]. The only facts about **SFP** which we will need are that it is a category of algebraic domains closed under the following type constructions:

Separated Sum

Let A be a countable set, and $\{D_a\}_{a \in A}$ an A-indexed family of domains. Then $\sum_{a \in A} D_a$ is formed by taking the disjoint union of the D_a and adjoining a bottom element. We shall write elements of the disjoint union as $\langle a, d \rangle$ ($a \in A$, $d \in D_a$). Note that the ordering is defined so that

$$\langle a, d \rangle \sqsubseteq \langle a', d' \rangle \iff a = a' \ \&\ d \sqsubseteq_{D_a} d'.$$

- For each $a \in A$, the function

$$\begin{aligned} D_a &\to \sum_{a \in A} D_a \\ d &\mapsto \langle a, d \rangle \end{aligned}$$

  is continuous.

- Separated sum is functorial; given a family

$$f_a : D_a \to E_a \ \ (a \in A),$$

  the function

$$\sum_{a \in A} f_a : \sum_{a \in A} D_a \to \sum_{a \in A} E_a$$

  is defined by:

$$\begin{aligned} (\textstyle\sum_{a \in A} f_a)\bot &= \bot \\ (\textstyle\sum_{a \in A} f_a)\langle a, d \rangle &= \langle a, f_a d \rangle. \end{aligned}$$

The Plotkin Powerdomain

We write $P[D]$ for the Plotkin powerdomain over D. Although this construction is best *characterised* abstractly, as in [HP79], for purposes of comparison with more concrete operational notions a good representation is invaluable. This is provided in [Plo76, Plo81].

Definition 3.1 For an algebraic domain D the *Lawson topology* on D is generated by the sub-basic sets

$$\uparrow b, \quad D \setminus \uparrow b$$

for finite $b \in D$, where $\uparrow b = \{d \in D : b \sqsubseteq d\}$ (so the Lawson topology refines the Scott topology). We will write the closure operator associated with the Lawson topology as Cl. (NB: in [Plo76], the Lawson topology is called the Cantor topology).

Definition 3.2 For $X \subseteq D$,

$$\begin{array}{lll} (i) & Con(X) & \equiv \{d : \exists d_1, d_2 \in X.\, d_1 \sqsubseteq d \sqsubseteq d_2\} \\ (ii) & X^\star & \equiv Con \circ Cl(X). \end{array}$$

X is said to be

- *Lawson-closed* if $X = Cl\ X$
- *Convex-closed* if $X = Con\ X$
- *Closed* if $X = X^\star$.

Definition 3.3 The *Egli-Milner order*. For $X, Y \subseteq D$:

$$X \sqsubseteq_{EM} Y \equiv \forall x \in X.\, \exists y \in Y.\, x \sqsubseteq y \ \&\ \forall y \in Y.\, \exists x \in X.\, x \sqsubseteq y.$$

The representation of the Plotkin powerdomain can now be defined as follows:

$$P[D] \equiv (\{X \subseteq D : X \neq \varnothing, X = X^\star\}, \sqsubseteq_{EM}).$$

There are also a number of (continuous) operations associated with the Plotkin powerdomain, which we shall describe in terms of our representation of $P[D]$.

- Firstly, P is *functorial*: given $f : D \to E$,

  $$Pf : P[D] \to P[E]$$

  is defined by

  $$Pf(X) \equiv \{f(x) : x \in X\}^\star.$$

- *Singleton:*

  $$\{\!|.|\!\} : D \to P[D]$$

  is defined by

  $$\{\!|d|\!\} \equiv \{d\}^\star = \{d\}.$$

- *Union:*

  $$\uplus : P[D]^2 \to P[D]$$

  is defined by

  $$X \uplus Y \equiv (X \uplus Y)^\star = Con(X \cup Y).$$

- *Big Union:*

  $$\biguplus : P[P[D]] \to P[D]$$

  is defined by

  $$\biguplus(\Theta) \equiv (\bigcup \Theta)^\star = Con(\bigcup \Theta).$$

- *Tensor Product* [HP79]. We will only need the following: given

$$f : D^n \to D$$

the *multilinear extension*

$$f^\dagger \; P[D]^n \to P[D]$$

is defined by

$$f^\dagger(X_1, \ldots, X_n) \;\; \equiv \;\; \{f(x_1, \ldots, x_n) : x_i \in X_i\}^\star.$$

(Note that for $n = 1$, $f^\dagger = Pf$.) This extension has the property

$$f^\dagger(X_1, \ldots, X_i \uplus X_i', \ldots, X_n) = f^\dagger(X_1, \ldots, X_i, \ldots, X_n) \uplus f^\dagger(X_1, \ldots, X_i', \ldots, X_n)$$

for $(1 \leq i \leq n)$.

Adjoining the empty set

To the best of my knowledge, the only significant precursor of the work in this paper is [MM79]. The main reason that something like our present programme could not have been carried through in their framework is that, because of a technical problem, they used the Smyth rather than the Plotkin powerdomain. This rules out any hope of gaining a correspondence with bisimulation. The technical problem is that of adjoining the empty set to the powerdomain to model the convergent process with no actions (NIL in CCS [Mil80], $\mathbb{O}$ in SCCS [Mil83], STOP in CSP [Hoa85], δ in ACP [BK84], etc.). If we add the empty set to our representation of $P[D]$, it is not related to anything except itself under $\sqsubseteq_{EM}$; in category-theoretic terms, the problem is the non-existence of a certain free construction ([Plo81]). Fortunately, we do not need these non-existent solutions. We shall adjoin the empty set to the Plotkin powerdomain in a way which has two advantages:

1. There is no theoretical overhead, since it is definable as a derived operation from standard type constructions.
2. It works, i.e. is exactly suited to our semantic purposes, as the results to follow will show.

For motivation, consider a transition system $(\mathrm{Proc}, \mathsf{Act}, \to, \uparrow)$ and processes $p, r \in \mathrm{Proc}$ such that

$$\begin{array}{ll} (i) & p\uparrow,\ r\downarrow \\ (ii) & p \not\to,\ r \not\to . \end{array}$$

Then it is easy to see that, for all $q \in \mathrm{Proc}$:

$$\begin{array}{llll} (i) & r \precsim^B q & \iff & r \sim^B q \\ (ii) & q \precsim^B r & \iff & q \not\to \\ & & \iff & q \sim^B p \text{ or } q \sim^B r. \end{array}$$

This suggests the following

Definition 3.4 $P^0[D]$, the Plotkin powerdomain with empty set.
Representation of $P^0[D]$:

$$\textbf{Elements} \quad \{X \subseteq D : X = X^*\} = P[D] \cup \{\varnothing\}.$$
$$\textbf{Ordering} \quad X \sqsubseteq Y \;\equiv\; X = \{\perp\} \text{ or } X \sqsubseteq_{EM} Y.$$

Observation 3.5 $P^0[D] \;\cong\; (\mathbf{1})_\perp \oplus P[D]$.

(Here $\mathbf{1}$ is the one-point domain, $(\cdot)_\perp$ is lifting, and $\oplus$ coalesced sum; cf. [Plo81, Abr88a]).

In principle, we could work throughout with 3.5 as the *definition* of $P^0[D]$; in practice, it is much more convenient to work with the representation given by 3.4. This requires that we extend our definitions of the powerdomain operations to work on $P^0[D]$. In fact, all of the definitions following 3.3 still make sense for $P^0[D]$. It is easily checked that $\uplus$, $\biguplus$ and $\{\!|\cdot|\!\}$ are continuous on $P^0[D]$. For $P^0 f$ and $f^\dagger$ a technical point arises, which is not specific to 3.4, but stems from the use of coalesced sum in 3.5. As is well known, coalesced sum is functorial only on the category of *strict* functions. Hence we can only use $P^0 f$ if f is strict, and $f^\dagger$ if f is strict in each argument separately. With these provisos, the extended operations are continuous.

Notation. We use $\emptyset$ to denote the empty set in $P^0[D]$; if I is a finite index set, we write

$$\biguplus_{i \in I} X_i$$

meaning the iterated use of $\uplus$ (which is associative, commutative and idempotent on $P^0[D]$, just as it is on $P[D]$) if $I \neq \varnothing$, and $\emptyset$ otherwise. Also, we write

$$\{\!|d : A|\!\}$$

where $d \in D$ and A is some sentence, meaning $\{\!|d|\!\}$ if A is true, and $\emptyset$ otherwise.

We are now ready for the main definition of the section.

Definition 3.6 Let Act be a *countable* set of actions. Then $\mathcal{D}(\mathsf{Act})$, the domain of synchronisation trees over Act (we henceforth omit the parameter Act), is defined to be the initial solution of the domain equation

$$\mathcal{D} \;\cong\; P^0[\sum_{a \in \mathsf{Act}} \mathcal{D}]. \tag{2}$$

Here the sum $\sum_{a \in \mathsf{Act}} \mathcal{D}$ is the "copower" of Act copies of $\mathcal{D}$. The equation is essentially that of [MM79], minus the value passing and with a different powerdomain.

How can we relate this domain equation to the formalism of [Abr88a]? We can write

$$\mathcal{D} \;\equiv\; \mathsf{rec}\, t.(\mathbf{1})_\perp \oplus P[\sum_{a \in \mathsf{Act}} t]$$

using (3.5) to eliminate P^0. This is not yet a valid type expression because of the sum

$$\sum_{a \in \mathsf{Act}} t \tag{3}$$

Let us take the main case of interest, where Act is countably infinite, say $\mathsf{Act} = \{a_n\}_{n \in \omega}$. Then we can replace (3) by the recursive expression

$$\mathsf{rec}\; u.(t)_\perp \oplus u \tag{4}$$

yielding the overall expression

$$\mathcal{D} \equiv \mathsf{rec}\, t.(\mathbf{1})_\perp \oplus P[\mathsf{rec}\, u.(t)_\perp \oplus u] \tag{5}$$

the intention being that the i'th summand as we unfold (4) corresponds to $a_i \in \mathsf{Act}$.

The reader will by now probably appreciate our efforts to streamline the presentation. Nevertheless, we regard the "closed form" expression (5) as fundamental, and the logic we shall introduce in the next section could be derived mechanically from it in the manner detailed in [Abr87b].

In the remainder of this section, we shall apply some standard domain-theoretic methods to elucidate the structure of $\mathcal{D}$.

Notation. We write $\perp$ for the bottom element of $\sum_{a\in\mathsf{Act}} \mathcal{D}$; $\{\!|\perp|\!\}$ is then the bottom element of $P^0[\sum_{a\in\mathsf{Act}} \mathcal{D}]$.

How can we unpack the structure of $\mathcal{D}$ from the domain equation (2)? This is best done in two parts:

1. A *specified* isomorphism pair

$$\mathcal{D} \underset{\theta}{\overset{\eta}{\rightleftarrows}} P^0[\sum_{a\in\mathsf{Act}} \mathcal{D}].$$

   In fact, we shall elide η and θ, and treat (2) as an identity; this is only a notational convenience, and the reader can put η and θ back without encountering any difficulties.

2. *Initiality.* The categorical framework is clumsy to work with for our purposes. Instead, we will use an "intrinsic" (or in the terminology of [SP82] a "local" or "**O**-notion") formulation.

Definition 3.7 We define a sequence of functions

$$\pi_k : \mathcal{D} \to \mathcal{D}$$

as follows:

$$\begin{array}{rcl} \pi_0 & \equiv & \lambda x \in \mathcal{D}.\{\!|\perp|\!\} \\ \pi_{k+1} & \equiv & P^0 \sum_{a\in\mathsf{Act}} \pi_k. \end{array}$$

Note that $\sum_{a\in\mathsf{Act}}$ always produces a strict function, so this is well-defined.

Now the following proposition is standard ([Plo81, Chapter 5 Theorem 3]):

Proposition 3.8 *$\mathcal{D}$ is the "internal colimit" of the π_k:*

$$\begin{array}{rl} (i) & \textit{Each } \pi_k \textit{ is continuous and } \pi_k \sqsubseteq \pi_{k+1} \\ (ii) & \bigsqcup_k \pi_k = \mathsf{id}_{\mathcal{D}} \\ (iii) & \pi_k \circ \pi_k = \pi_k \\ (iv) & \forall d_1, d_2 \in \mathcal{D}.\, d_1 \sqsubseteq d_2 \iff \forall k.\, \pi_k d_1 \sqsubseteq \pi_k d_2. \end{array}$$

In particular, we will use part (iv) of this Proposition as the cutting edge of initiality.

Next, it will be useful to have an explicit description of the finite elements of $\mathcal{D}$, which, as already noted, is in **SFP**, and hence algebraic.

Definition 3.9 $\mathcal{K}(\mathcal{D}) \subseteq \mathcal{D}$ is defined inductively as follows:

- $\emptyset \in \mathcal{K}(\mathcal{D})$
- $\{\!|\bot|\!\} \in \mathcal{K}(\mathcal{D})$
- $a \in \mathsf{Act}, d \in \mathcal{K}(\mathcal{D}) \;\Rightarrow\; \{\!|\langle a, d\rangle|\!\} \in \mathcal{K}(\mathcal{D})$
- $d_1, d_2 \in \mathcal{K}(\mathcal{D}) \;\Rightarrow\; d_1 \uplus d_2 \in \mathcal{K}(\mathcal{D})$.

The following is again standard:

Proposition 3.10 *$\mathcal{K}(\mathcal{D})$ is exactly the set of finite elements of $\mathcal{D}$.*

Finally, we consider $\mathcal{D}$ as a *transition system* $(\mathcal{D}, \mathsf{Act}, \rightarrow, \uparrow)$ defined by:

$$\begin{array}{lll} \bullet \; d \xrightarrow{a} d' & \equiv & \langle a, d'\rangle \in d \\ \bullet \; d\uparrow & \equiv & \bot \in d. \end{array}$$

Proposition 3.11 *$\mathcal{D}$ is "internally fully abstract", i.e.*

$$\forall d_1, d_2 \in \mathcal{D} \,.\, d_1 \precsim^B d_2 \iff d_1 \sqsubseteq d_2.$$

Proof. We shall prove

$$\begin{array}{ll} (1) & \forall k.\; d_1 \precsim_k d_2 \implies \pi_k d_1 \sqsubseteq \pi_k d_2 \\ (2) & \sqsubseteq \,\subseteq\, \precsim^B. \end{array}$$

Clearly (1) implies $\precsim_\omega \;\subseteq\; \sqsubseteq$ by (3.8)(iv), and since $\precsim^B \;\subseteq\; \precsim_\omega$, we obtain $\precsim^B \;=\; \sqsubseteq$, as required.

(1). By induction on k. The basis is trivial. For the inductive step, assume $d \precsim_{k+1} e$. Now $d = \emptyset$ and $d \precsim_{k+1} e$ implies $e = \emptyset$, while $d = \{\!|\bot|\!\}$ implies $d \sqsubseteq e$, so we may assume $d \neq \emptyset \neq e$, and it suffices to prove $d \sqsubseteq_{EM} e$.

From the definitions we have $\pi_{k+1} d = X^\star$, where

$$X = \{\langle a, \pi_k d'\rangle : \langle a, d'\rangle \in d\} \cup \{\bot : \bot \in d\},$$

and similarly $\pi_{k+1} e = Y^\star$. Now

$$\begin{array}{ll} \bullet & <a, \pi_k d'> \in X \\ \implies & d \xrightarrow{a} d' \\ \implies & \exists e'.\, e \xrightarrow{a} e' \;\&\; d' \precsim_k e' \\ \implies & \exists e'.\, \langle a, e'\rangle \in e \;\&\; \pi_k d' \sqsubseteq \pi_k e' \;\; \text{by induction hypothesis} \\ \implies & \exists \langle a, \pi_k e'\rangle \in Y.\, \langle a, \pi_k d'\rangle \sqsubseteq \langle a, \pi_k e'\rangle. \end{array}$$

Again,

$$\begin{array}{ll}
\bullet & \bot \notin X \\
\Longrightarrow & \bot \notin d \\
\Longrightarrow & \bot \notin e \mathbin{\&} [e \xrightarrow{a} e' \;\Rightarrow\; \exists d'.\, d \xrightarrow{a} d' \mathbin{\&} d' \precsim_k e'] \\
\Longrightarrow & \bot \notin Y \mathbin{\&} \forall \langle a, \pi_k e' \rangle \in Y.\, \exists \langle a, \pi_k d' \rangle \in X.\, \pi_k d' \sqsubseteq \pi_k e'
\end{array}$$

by the induction hypothesis again, and we have shown $X \sqsubseteq_{EM} Y$, which implies $X^\star \sqsubseteq_{EM} Y^\star$, as required.
(2). It suffices to show that $\sqsubseteq$ is a partial bisimulation. This is a simple calculation:

$$\begin{array}{ll}
\bullet & d \sqsubseteq e \\
\Longrightarrow & \forall \langle a, d' \rangle \in d.\, \exists \langle a, e' \rangle \in e.\, d' \sqsubseteq e' \\
& \mathbin{\&} \bot \notin d \;\Rightarrow\; \bot \notin e \mathbin{\&} [\forall \langle a, e' \rangle \in e.\, \exists \langle a, d' \rangle \in d.\, d' \sqsubseteq e'] \\
\Longrightarrow & \forall a \in \mathsf{Act}.\, d \xrightarrow{a} d' \;\Rightarrow\; \exists e'.\, e \xrightarrow{a} e' \mathbin{\&} d' \sqsubseteq e' \\
& \mathbin{\&} d{\downarrow} \;\Rightarrow\; e{\downarrow} \mathbin{\&} [e \xrightarrow{a} e' \;\Rightarrow\; \exists d'.\, d \xrightarrow{a} d' \mathbin{\&} d' \sqsubseteq e']. \quad \blacksquare
\end{array}$$

We finish with some examples to illustrate the richness of $\mathcal{D}$ as a transition system.

Examples

(1). $\mathcal{D}$ is not sort-finite.

$$\begin{array}{rcl}
d_0 & \equiv & \{\!|\langle a_0, \{\!|\bot|\!\}\rangle|\!\} \\
d_1 & \equiv & \{\!|\langle a_0, \{\!|\langle a_1, \{\!|\bot|\!\}\rangle|\!\}\rangle|\!\} \\
& \vdots & \\
\mathsf{sort}(\bigsqcup d_k) & = & \{a_0, a_1, \ldots\}
\end{array}$$

(2). $\mathcal{D}$ is not weakly image-finite.

$$\begin{array}{rcl}
c_k & \equiv & \sum_{i \le k} a^i \mathbb{O} + a^k \Omega \quad (k \in \omega) \\
\bigsqcup c_k & = & \sum_{k \in \omega} a^k \mathbb{O} + a^\omega .
\end{array}$$

4 A Domain Logic for Transition Systems

We now introduce our *domain logic* in the sense of [Abr88a] in an infinitary version $\mathcal{L}_\infty$, with a finitary subset $\mathcal{L}_\omega$. We show how $\mathcal{L}_\infty$ can be interpreted in any transition system, present a proof system, and establish its soundness. We then turn to $\mathcal{L}_\omega$, and prove the main result of the section: $\mathcal{L}_\omega$ is the Stone dual of $\mathcal{D}$. That is, $\mathcal{D}$ is isomorphic to the spectral space of $\mathcal{L}_\omega$, while $\mathcal{L}_\omega$ is isomorphic to the lattice of compact-open subsets of $\mathcal{D}$. This duality will be crucial to our work in the next section.

Definition 4.1 The language $\mathcal{L}_\infty$ has two *sorts*: π (process) and κ (capability). We write $\mathcal{L}_{\infty\pi}$ ($\mathcal{L}_{\infty\kappa}$) for the class of formulae of sort π (κ), which are defined inductively as follows:

- $$\frac{\{\phi_i \in \mathcal{L}_{\infty\sigma}\}_{i\in I}}{\bigvee_{i\in I}\phi_i, \bigwedge_{i\in I}\phi_i \in \mathcal{L}_{\infty\sigma}} \quad (\sigma \in \{\pi,\kappa\})$$
- $$\frac{a \in \mathsf{Act},\quad \phi \in \mathcal{L}_{\infty\pi}}{a(\phi) \in \mathcal{L}_{\infty\kappa}}$$
- $$\frac{\phi \in \mathcal{L}_{\infty\kappa}}{\Box\phi, \Diamond\phi \in \mathcal{L}_{\infty\pi}}.$$

Notation. We write $t \equiv \bigwedge_{i\in\varnothing}\phi_i$, $f \equiv \bigvee_{i\in\varnothing}\phi_i$.

The sublanguage of $\mathcal{L}_\infty$ obtained by the restriction to *finite* conjunctions and disjunctions is denoted $\mathcal{L}_\omega$. *Height*, *modal depth* and *sort* are defined for $\mathcal{L}$ in entirely analogous fashion to HML. For example:

- $\mathsf{md}(\bigwedge_{i\in I}\phi_i) \equiv \mathsf{md}(\bigwedge_{i\in I}\phi_i) \equiv \sup\{\mathsf{md}(\phi_i) : i \in I\}$
- $\mathsf{md}(a(\phi)) \equiv \mathsf{md}(\phi)$
- $\mathsf{md}(\Box\phi) \equiv \mathsf{md}(\Diamond\phi) \equiv \mathsf{md}(\phi) + 1.$

For each $A \subseteq \mathsf{Act}$ and ordinal λ:

$$\mathcal{L}_\infty^{(A,\lambda)} \equiv \{\phi \in \mathcal{L}_\infty : \mathsf{sort}(\phi) \subseteq A \;\&\; \mathsf{md}(\phi) \leq \lambda\}.$$

It should be clear how the form of our language is derived from the type expression

$$\mathsf{rec}\, t.\, P^0[\sum_{a\in\mathsf{Act}} t].$$

The two-sorted structure of $\mathcal{L}$ corresponds to the type constructions P^0 (π) and $\sum_{a\in\mathsf{Act}}$ (κ). The recursion in the type expression is mirrored by the mutual recursion between the two sorts. Note that the Plotkin powerdomain is built from the combination of the *must* modality $\Box$ of the Smyth powerdomain and the *may* modality $\Diamond$ of the Hoare powerdomain (*cf.* [Abr83a, Win83]).

Interpretation of $\mathcal{L}$ in transition systems

Given a transition system $(\mathrm{Proc}, \mathsf{Act}, \rightarrow, \uparrow)$, we define

$$\mathrm{Cap} \equiv \{\bot\} \cup (\mathsf{Act} \times \mathrm{Proc})$$

$$C : \mathrm{Proc} \rightarrow \wp(\mathrm{Cap})$$

$$C(p) = \{\bot : p\uparrow\} \cup \{\langle a, q\rangle : p \xrightarrow{a} q\}.$$

$C(p)$ is the set of *capabilities* of p. We can now define satisfaction relations

$$\models_\pi \subseteq \mathrm{Proc} \times \mathcal{L}_{\infty\pi},$$

$$\models_\kappa \subseteq \mathrm{Cap} \times \mathcal{L}_{\infty\kappa} :$$

For $\sigma \in \{\pi, \kappa\}$:

$$\begin{array}{lcl}
w \models_\sigma \bigwedge_{i\in I} \phi_i & \equiv & \forall i \in I.\, w \models_\sigma \phi_i \\
w \models_\sigma \bigvee_{i\in I} \phi_i & \equiv & \exists i \in I.\, w \models_\sigma \phi_i \\
p \models_\pi \Box\phi & \equiv & p\!\downarrow \,\&\, \forall c \in C(p).\, c \models_\kappa \phi \\
p \models_\pi \Diamond\phi & \equiv & \exists c \in C(p).\, c \models_\kappa \phi \\
c \models_\kappa a(\phi) & \equiv & c = \langle a, q\rangle \,\&\, q \models_\pi \phi.
\end{array}$$

The *assertions* over $\mathcal{L}$ have the form

$$\phi \leq_\sigma \psi, \;\; \phi =_\sigma \psi \quad (\sigma \in \{\pi, \kappa\}, \phi, \psi \in \mathcal{L}_{\infty\sigma}).$$

The satisfaction relation between transition systems $\mathcal{T}$ and assertions is defined by:

$$\begin{array}{lcl}
\mathcal{T} \models \phi \leq_\sigma \psi & \equiv & \forall w \in S_\sigma.\, w \models_\sigma \phi \implies w \models_\sigma \psi \\
\mathcal{T} \models \phi =_\sigma \psi & \equiv & \forall w \in S_\sigma.\, w \models_\sigma \phi \iff w \models_\sigma \psi.
\end{array}$$

$$(\sigma \in \{\pi, \kappa\}, S_\pi = \mathrm{Proc}, S_\kappa = \mathrm{Cap}).$$

This is extended to a class of transition systems $\mathbf{C}$ by:

$$\mathbf{C} \models A \;\;\equiv\;\; \forall \mathcal{T} \in \mathbf{C}.\, \mathcal{T} \models A.$$

If $\mathbf{C}$ is the class of all transition systems, we simply write $\models A$.

A Proof System For $\mathcal{L}_\infty$

We now present a proof system for assertions over $\mathcal{L}_\infty$. Sort subscripts are omitted.

Logical Axioms

$$(\leq -\,\mathrm{ref}) \;\; \phi \leq \phi \qquad (\leq -\,\mathrm{trans}) \;\; \frac{\phi \leq \psi, \;\; \psi \leq \chi}{\phi \leq \chi}$$

$$(= -\, I) \;\; \frac{\phi \leq \psi, \;\; \psi \leq \phi}{\phi = \psi} \qquad (= -\, E) \;\; \frac{\phi = \psi}{\phi \leq \psi, \;\; \psi \leq \phi}$$

$$(\wedge -\, I) \;\; \frac{\{\phi \leq \psi_i\}_{i\in I}}{\phi \leq \bigwedge_{i\in I} \psi_i}$$

$$(\wedge -\, E) \;\; \bigwedge_{i\in I} \phi_i \leq \phi_j \qquad (j \in I)$$

$$(\vee -\, I) \;\; \frac{\{\phi_i \leq \psi\}_{i\in I}}{\bigvee_{i\in I} \phi_i \leq \psi}$$

$$(\vee - E) \quad \phi_j \leq \bigvee_{i \in I} \phi_i \qquad (j \in I)$$

$$(\wedge - \mathrm{dist}) \quad \bigwedge_{i \in I} \bigvee_{j \in J_i} \phi_{i,j} = \bigvee_{f \in \prod_{i \in I} J_i} \bigwedge_{i \in I} \phi_{i,f(i)}$$

$$(\vee - \mathrm{dist}) \quad \bigvee_{i \in I} \bigwedge_{j \in J_i} \phi_{i,j} = \bigwedge_{f \in \prod_{i \in I} J_i} \bigvee_{i \in I} \phi_{i,f(i)}$$

Modal Axioms

$$(a - \leq) \quad \frac{\phi \leq \psi}{a(\phi) \leq a(\psi)}$$

$$(a - \wedge)(i) \quad a(\bigwedge_{i \in I} \phi_i) = \bigwedge_{i \in I} a(\phi_i) \quad (I \neq \varnothing)$$

$$(a - \wedge)(ii) \quad a(\phi) \wedge b(\psi) = f \quad (a \neq b)$$

$$(a - \vee) \quad a(\bigvee_{i \in I} \phi_i) = \bigvee_{i \in I} a(\phi_i)$$

$$(\Box - \leq) \quad \frac{\phi \leq \psi}{\Box\phi \leq \Box\psi}$$

$$(\Box - \wedge) \quad \Box \bigwedge_{i \in I} \phi_i = \bigwedge_{i \in I} \Box\phi_i \quad (I \neq \varnothing)$$

$$(\Diamond - \leq) \quad \frac{\phi \leq \psi}{\Diamond\phi \leq \Diamond\psi}$$

$$(\Diamond - \vee) \quad \Diamond \bigvee_{i \in I} \phi_i = \bigvee_{i \in I} \Diamond\phi_i$$

$$(\Box - \vee) \quad \Box(\phi \vee \psi) \leq \Box\phi \vee \Diamond\psi$$

$$(\Diamond - \wedge) \quad \Box\phi \wedge \Diamond\psi \leq \Diamond(\phi \wedge \psi)$$

The form of our axiomatisation follows the same pattern as that of [Abr88a], of which it is of course a special case. The first group of axioms and rules give the logical structure of entailment, conjunction and disjunction. They give (the Lindenbaum algebra of) $\mathcal{L}_\infty$ the structure of a (large) *completely distributive lattice* [Joh82]. We then articulate the modal structure by showing how the constructors interact with the logical structure. The axioms for the $a(\cdot)$ constructor correspond to those for coalesced sum given in [Abr87b]; the fact that *separated* sum is intended here is reflected by the side-condition on $(a - \wedge)(i)$. The axioms for $\Box$ and $\Diamond$ individually correspond to those presented for the upper and lower powerdomains in [Abr87b]; however, these two modalities interact in the Plotkin powerdomain, resulting in its greater complexity; these interactions are expressed in logical terms by $(\Box - \vee)$ and $(\Diamond - \wedge)$. Our surgery on the ordering to keep a least element while adding the empty set is reflected by the side condition on $(\Box - \wedge)$.

We write $\mathcal{L} \vdash A$ or just $\vdash A$ if an assertion A is derivable from the above rules and axioms. It will be convenient to have equational versions of $(\Box - \vee)$ and $(\Diamond - \wedge)$, which can be obtained as theorems of $\mathcal{L}$:

$$(D1) \quad \vdash \Box(\phi \vee \psi) = \Box\phi \vee (\Box(\phi \vee \psi) \wedge \Diamond\psi)$$

$$(D2) \quad \vdash \Box\phi \wedge \Diamond\psi = \Box\phi \wedge \Diamond(\phi \wedge \psi)$$

We now turn to the question of soundness for our system.

Theorem 4.2 (Soundness of $\mathcal{L}$) $\vdash A \implies \models A$.

PROOF. By a routine induction over proofs. For illustration, we consider $(\Diamond - \wedge)$. Assume $p \models_\pi \Box\phi \wedge \Diamond\psi$. Then $p \models_\pi \Diamond\psi$, and so there must be $c \in C(p)$ such that $c \models_\kappa \psi$. But then $p \models_\pi \Box\phi$ implies that $c \models_\kappa \phi$, and so $p \models_\pi \Diamond(\phi \wedge \psi)$ as required. ∎

We now turn to the finitary logic $\mathcal{L}_\omega$. Henceforth we assume that Act is countable. It is then clear that $\mathcal{L}_\omega$ can be made into a countable set by a suitable choice of canonical representatives of logical equivalence classes.

Recall that $\mathsf{Spec}\ \mathcal{L}_\omega$ is the set of *prime filters* over $\mathcal{L}_{\omega\pi}$, i.e. subsets $x \subseteq \mathcal{L}_{\omega\pi}$ satisfying

- $\phi \in x \ \&\ \vdash \phi \leq \psi \ \Rightarrow\ \psi \in x$
- $t \in x$
- $\phi, \psi \in x \ \Rightarrow\ \phi \wedge \psi \in x$
- $f \notin x$
- $\phi \vee \psi \in x \ \Rightarrow\ \phi \in x$ or $\psi \in x$.

$\mathsf{Spec}\ \mathcal{L}_\omega$ is topologised by taking as basic opens

$$U_\phi \equiv \{x \in \mathsf{Spec}\ \mathcal{L}_\omega : \phi \in x\} \quad (\phi \in \mathcal{L}_{\omega\pi}),$$

or, equivalently in our context, by taking the Scott topology over the specialisation order on $\mathsf{Spec}\ \mathcal{L}_\omega$, which is simply set inclusion.

Our aim is to prove the following fundamental result, which ahows that the logic $\mathcal{L}_\omega$ does indeed correspond exactly to the domain $\mathcal{D}$:

Theorem 4.3 (Stone Duality) *$\mathcal{D}$ and $\mathcal{L}_\omega$ are Stone duals, i.e.*

$$\begin{array}{ll} (i) & \mathcal{D} \cong \mathsf{Spec}\ \mathcal{L}_\omega \\ (ii) & K\Omega(\mathcal{D}) \cong (\mathcal{L}_{\omega\pi}/{=_\pi}, \leq_\pi/{=_\pi}). \end{array}$$

Here $K\Omega(D)$ is the lattice of compact-open subsets of $\mathcal{D}$, while

$$(\mathcal{L}_{\omega\pi}/{=_\pi}, \leq_\pi/{=_\pi})$$

is the *Lindebaum algebra* of $\mathcal{L}_\omega$. Since $\mathcal{D}$ is coherent, (i) and (ii) are indeed equivalent ([Joh82]).

The Stone Duality Theorem is a special case of [Abr88a, Theorem 4.2.4]; however, we shall give the proof here, in order to make the paper more self-contained.

We begin by defining a normal form for $\mathcal{L}_\omega$.

Definition 4.4 (i) ϕ is in *strong disjunctive normal form* (SDNF) if it has the form $\bigvee_{i\in I}\phi_i$, where each ϕ_i is in *coprime normal form* (CNF).
(ii) ϕ is in CNF if it has one of the forms

- $\bigwedge_{i\in I}\Diamond a_i(\phi_i)$, where each ϕ_i is in CNF.
- $\Box\bigvee_{i\in I} a_i(\phi_i) \ \wedge\ \bigwedge_{j\in J}\Diamond b_j(\psi_j)$, where
    1. Each ϕ_i and ψ_j is in CNF.

2. $\forall i \in I. \exists j \in J. \vdash b_j(\psi_j) \leq a_i(\phi_i)$.
3. $\forall j \in J. \exists i \in I. \vdash b_j(\psi_j) \leq a_i(\phi_i)$.

We call (2) and (3) the *convexity conditions* (note the resemblance to the Egli–Milner ordering).

The combinatorics are concentrated in the following

Theorem 4.5 (SDNF) *For every $\phi \in \mathcal{L}_{\omega\pi}$, there is (effectively) a ψ in SDNF such that*

$$\vdash \phi =_\pi \psi.$$

PROOF. By induction on $\mathsf{md}(\phi)$. The idea is to form a sequence of "transformations"

$$\phi \equiv \phi_0 \rightsquigarrow \phi_1 \rightsquigarrow \cdots \rightsquigarrow \phi_n$$

such that

$$(1) \quad \vdash \phi_i = \phi_{i+1} \quad (0 \leq i < n)$$

$$(2) \quad \mathsf{md}(\phi_{i+1}) \leq \mathsf{md}(\phi_i) \quad (0 \leq i < n)$$

$$(3) \quad \phi_n \text{ is in SDNF.}$$

(Condition (2) is needed to keep the induction going.) To keep the notation bearable, we shall omit indices in conjunctions and disjunctions, writing e.g. $\bigvee\{\phi\}$.

Firstly, using the distributive lattice laws we can transform ϕ_0 into

$$\bigvee\{\bigwedge\{\Box \bigwedge\{\bigvee\{a(\phi)\}\}\} \wedge \bigwedge\{\Diamond \bigwedge\{\bigvee\{b(\psi)\}\}\}\} \tag{6}$$

Using $(\Box - \wedge)$ in the outwards direction for each non-empty $\Box$-conjunct in (6), and the distributive law and then $(\Diamond - \vee)$, followed by the distributive law again, in each $\Diamond$-conjunct, we otain

$$\bigvee\{\bigwedge\{\Box \bigvee\{a(\phi)\}\} \wedge \bigwedge\{\Diamond \bigwedge\{b(\psi)\}\}\} \tag{7}$$

Now for each non-empty conjunction

$$\bigwedge\{\Box \bigvee\{a(\phi)\}\}$$

in (7), we can use $(\Box - \wedge)$, the distributive law, and $(a - \wedge)$ (i) or (ii) to get

$$\bigwedge\{\Box \bigvee\{a(\phi)\}\} = \Box \bigwedge\{\bigvee\{a(\phi)\}\} = \Box \bigvee\{\bigwedge\{a(\phi)\}\} = \Box \bigvee\{a(\bigwedge\{\phi\})\}.$$

Similarly, inside each $\Diamond \bigwedge\{b(\psi)\}$ we can use $(b - \wedge)$ (i) or (ii) (with further applications of $(\Diamond - \vee)$ and the distributive laws as in the previous step if $(b - \wedge)(ii)$ is applicable), to obtain

$$\bigvee\{\theta\} \tag{8}$$

where each θ is in one of the forms

$$\bigwedge\{\Diamond b(\psi)\} \tag{9}$$

or

$$\Box \bigvee\{a(\phi)\} \wedge \bigwedge\{\Diamond b(\psi)\} \tag{10}$$

Since we have not increased modal depth in obtaining (8), we can apply the inductive hypothesis to each ϕ and ψ to obtain $\bigvee\{\phi'\}$, $\bigvee\{\psi'\}$ with each ϕ' and ψ' in CNF. Using $(a - \vee)$, $(\Diamond - \vee)$ and the distributive laws, we can thus obtain a formula of the same form as (8), in which each ϕ and ψ in (9) and (10) is in CNF.

At this point, our formula (8) can only fail to be in SDNF because of disjuncts (10) which do not satisfy the convexity conditions

- For each $a(\phi)$, for some $b(\psi)$: $\vdash b(\psi) \leq a(\phi)$.
- For each $b(\psi)$, for some $a(\phi)$: $\vdash b(\psi) \leq a(\phi)$.

Our strategy is to remove any failures of these two conditions, using our derived equations $(D1)$ and $(D2)$ respectively. We begin with the first condition. We argue by induction on (m, n) in the lexicographic ordering on $\omega \times \omega$, where:

- m is the maximum number of $a(\phi)$ occurring in one of the disjuncts (10) of our formula (8) such that there is no $b(\psi)$ with $\vdash b(\psi) \leq a(\phi)$.
- n is the number of disjuncts attaining this maximum.

If $m = 0$, there is nothing to prove. Otherwise, choose such an $a(\phi)$ in one of the maximal disjuncts. We can apply $(D1)$ to

$$\Box(\bigvee\{a'(\phi')\} \ \vee \ a(\phi))$$

to obtain

$$\Box \bigvee\{a'(\phi')\} \quad \vee \quad [\Box(\bigvee\{a'(\phi')\} \vee a(\phi)) \wedge \Diamond a(\phi)] \tag{11}$$

We can then use the distributive law to obtain a new formula of the form (8) to which the inner induction hypothesis can be applied, since the first disjunct in (11) has jettisoned $a(\phi)$, while the second disjunct evidently contains a $\Diamond b(\psi)$ such that $\vdash b(\psi) \leq a(\phi)$, namely $a(\phi)$ itself.

The final stage is to remove failures of the second condition. We argue by induction in the same way as for the previous stage. Suppose we are given a $b(\psi)$ in (10) with no $a(\phi)$ such that $\vdash b(\psi) \leq a(\phi)$. We can use $(D2)$ to obtain

$$\Box \bigvee\{a(\phi)\} \ \wedge \ \Diamond[b(\psi) \wedge \bigvee\{a(\phi)\}] \ \wedge \ \bigwedge\{\Diamond b'(\psi')\} \tag{12}$$

Now we can use the distributive law inside the second main conjunct in (12), followed by $(a - \wedge)$, $(\Diamond - \vee)$, and the distributive law again. In this way, the disjunct (12) of our main formula is replaced by the disjunction of all those formulae

$$\Box \bigvee\{a(\phi)\} \ \wedge \ \Diamond b(\phi' \wedge \psi) \ \wedge \ \bigwedge\{\Diamond b'(\psi')\} \tag{13}$$

for $a'(\phi') \in \{a(\phi)\}$ with $a' = b$. For each such $\phi' \wedge \psi$, we can apply the outer induction hypothesis to obtain $\bigvee\{\theta'\}$ with each θ' in CNF. Applying $(b - \vee)$, $(\Diamond - \vee)$ and the distributive laws as before, we obtain disjuncts of the form

$$\Box \bigvee\{a(\phi)\} \ \wedge \ \Diamond b(\theta') \ \wedge \ \bigwedge\{\Diamond b'(\psi')\} \tag{14}$$

Since

$$\vdash \ \theta' \leq \bigvee\{\theta'\} = \phi' \wedge \psi \leq \phi',$$

we can apply the inner induction hypothesis to (14). This completes the process of transforming ϕ into SDNF. ∎

We shall now prove that formulae in CNF denote coprimes in $K\Omega(\mathcal{D})$.

Proposition 4.6 *For all ϕ in CNF there exists $k(\phi) \in \mathcal{K}(\mathcal{D})$ such that:*

$$\forall d \in \mathcal{D}.\ d \models \phi \iff k(\phi) \sqsubseteq d.$$

PROOF. We define $k(\phi)$ (which must clearly be unique) by induction on ϕ:

- $k(\bigwedge_{i\in I} \Diamond a_i(\phi_i)) \equiv \biguplus_{i\in I} \{\!|\langle a_i, k(\phi_i)\rangle|\!\} \uplus \{\!|\bot|\!\}$

- $k(\Box \bigvee_{i\in I} a_i(\phi_i) \wedge \bigwedge_{j\in J} \Diamond b_j(\psi_j)) \equiv$

$$\biguplus_{i\in I} \{\!|\langle a_i, k(\phi_i)\rangle|\!\} \uplus \biguplus_{j\in J} \{\!|\langle b_j, k(\psi_j)\rangle|\!\}.$$

We shall prove the proposition by induction on ϕ. Note that in the statement of the proposition, we are viewing $\mathcal{D}$ as a transition system, according to 3.11. With our convention of eliding the isomorphisms between $\mathcal{D}$ and $P^0[\sum_{a\in\mathsf{Act}} \mathcal{D}]$, we have: $d = C(d)$, $(d \in \mathcal{D})$.

Case 1: $\phi \equiv \bigwedge_{i\in I} \Diamond a_i(\phi_i)$.

$$\begin{array}{ll}
\bullet & d \models \bigwedge_{i\in I} \Diamond a_i(\phi_i) \\
\iff & \forall i \in I.\, \exists \langle a_i, d_i\rangle \in d.\, d_i \models \phi_i \\
\iff & \forall i \in I.\, \exists \langle a_i, d_i\rangle \in d.\, k(\phi_i) \sqsubseteq d_i \ \text{ by induction hypothesis} \\
\iff & k(\phi) \sqsubseteq d.
\end{array}$$

Case 2: $\phi \equiv \Box \bigvee_{i\in I} a_i(\phi_i) \wedge \bigwedge_{j\in J} \Diamond b_j(\psi_j)$. Let $\Phi = \{a_i(\phi_i) : i \in I\} \cup \{b_j(\psi_j) : j \in J\}$.

$$\begin{array}{ll}
\bullet & d \models \phi \\
\iff & \forall \langle a, d'\rangle \in d.\, \exists i \in I.\, a = a_i \,\&\, d' \models \phi_i \\
 & \&\, \bot \notin d \,\&\, \forall j \in J.\, \exists \langle b_j, d_j\rangle \in d.\, d_j \models \psi_j \\
\iff & \forall \langle a, d'\rangle \in d.\, \exists a(\theta) \in \Phi.\, d' \models \theta \\
 & \&\, \bot \notin d \,\&\, \forall a(\theta) \in \Phi.\, \exists \langle a, d'\rangle \in d.\, d \models \theta \\
 & \text{by the convexity conditions and the Soundness Theorem,} \\
\iff & k(\phi) \sqsubseteq d, \text{ by induction hypothesis.} \ \blacksquare
\end{array}$$

Theorem 4.7 (Coprime Completeness) *For all ϕ, ϕ' in CNF:*

$$\mathcal{D} \models \phi \leq \phi' \implies \mathcal{L}_\omega \vdash \phi \leq \phi'.$$

PROOF. By 4.6,

$$\mathcal{D} \models \phi \leq \phi' \iff k(\phi') \sqsubseteq k(\phi).$$

Suppose then that $k(\phi') \sqsubseteq k(\phi)$. We argue by induction on ϕ. There are a number of cases, according to the forms of ϕ and ϕ'. We consider the case

$$\begin{array}{lll}
\phi & \equiv & \Box \bigvee_{i\in I} a_i(\phi_i) \wedge \bigwedge_{j\in J} \Diamond b_j(\psi_j), \\
\phi' & \equiv & \Box \bigvee_{i'\in I'} a_{i'}(\phi_{i'}) \wedge \bigwedge_{j'\in J'} \Diamond b_{j'}(\psi_{j'}).
\end{array}$$

$$\begin{array}{ll}
\bullet & k(\phi') \sqsubseteq k(\phi) \\
\iff & \forall j' \in J'.\, \exists j \in J.\, b_j = b_{j'} \,\&\, k(\psi_{j'}) \sqsubseteq k(\psi_j) \\
& \&\, \forall i \in I.\, \exists i' \in I'.\, a_i = a_{i'} \,\&\, k(\phi_{i'}) \sqsubseteq k(\phi_i), \\
& \text{by the convexity conditions, Soundness, and 4.6} \\
\implies & \forall j' \in J'.\, \exists j \in J.\, \vdash b_j(\psi_j) \leq b_{j'}(\psi_{j'}) \\
& \&\, \forall i \in I.\, \exists i' \in I'.\, \vdash\ a_i(\phi_i) \leq a_{i'}(\phi_{i'}), \\
& \text{by the induction hypothesis,} \\
\implies & \vdash\ \phi \leq \phi'. \ \blacksquare
\end{array}$$

Theorem 4.8 (Completeness) *For all $\phi, \psi \in \mathcal{L}_\omega$:*

$$\mathcal{D} \models \phi \leq \psi \implies \mathcal{L}_\omega \vdash \phi \leq \psi.$$

PROOF. Suppose $\mathcal{D} \models \phi \leq \psi$. By the SDNF Theorem, $\mathcal{L}_\omega \vdash \phi = \bigvee_{i \in I} \phi_i$, $\mathcal{L}_\omega \vdash \psi = \bigvee_{j \in J} \psi_j$, where ϕ_i, ψ_j are in CNF. By the Soundness Theorem and Proposition 4.6,

$$[\![\phi]\!] = [\![\bigvee \phi_i]\!] = \bigcup \uparrow k(\phi_i),$$

and similarly $[\![\psi]\!] = \bigcup \uparrow k(\psi_j)$. Now

$$\begin{array}{rcll}
[\![\phi]\!] \subseteq [\![\psi]\!] & \Rightarrow & \bigcup \uparrow k(\phi_i) \subseteq \bigcup \uparrow k(\psi_j) & \\
& \Rightarrow & \forall i.\, \exists j.\, k(\psi_j) \sqsubseteq k(\phi_i) & \\
& \Rightarrow & \forall i.\, \exists j.\, \mathcal{L}_\omega \vdash \phi_i \leq \psi_j & \text{Theorem 4.7} \\
& \Rightarrow & \mathcal{L}_\omega \vdash \bigvee \phi_i \leq \bigvee \psi_j & \\
& \Rightarrow & \mathcal{L}_\omega \vdash \phi \leq \psi. & \blacksquare
\end{array}$$

We now establish a converse to 4.6.

Theorem 4.9 (Definability) *For all $d \in \mathcal{K}(\mathcal{D})$, for some ϕ in CNF, $k(\phi) = d$.*

PROOF. We define $\phi(d)$ by induction on the construction of d according to 3.9:

$$\begin{array}{lcl}
\phi(\biguplus_{i \in I} \{\!|\langle a_i, d_i\rangle|\!\} \uplus \{\!|\bot|\!\}) & \equiv & \bigwedge_{i \in I} \Diamond a_i(\phi(d_i)) \\
\phi(\biguplus_{i \in I} \{\!|\langle a_i, d_i\rangle|\!\}) & \equiv & \Box \bigvee_{i \in I} a_i(\phi(d_i)) \wedge \bigwedge_{i \in I} \Diamond a_i(\phi(d_i)).
\end{array}$$

Note in particular that $\phi(\emptyset) = \Box f$. It is easily verified that $\phi(d)$ is in CNF and that $k(\phi(d)) = d$. $\blacksquare$

The Duality Theorem is an immediate consequence of Soundness, Completeness and Definability, just as in [Abr88a, T8].

Combining Soundness and Completeness we obtain

Theorem 4.10 (Completeness for $\mathcal{L}_\omega$) *Let $\mathbf{C}$ be any class of transition systems containing $\mathcal{D}$. Then for $\phi, \psi \in \mathcal{L}_\omega$:*

$$\mathbf{C} \models \phi \leq \psi \iff \mathcal{D} \models \phi \leq \psi \iff \mathcal{L}_\omega \vdash \phi \leq \psi.$$

5 Applications of the Domain Logic

We shall now use domain logic to study bisimulation. Our results in this section can be grouped under four main headings:

1. Comparisons with Hennessy-Milner logic
2. Characterisation Theorems
3. Finitary Transition Systems
4. Universal Semantics

Of these, (1) and (2) will confirm the appropriateness of our definitions, while (3) and (4) will represent a distinctive payoff for our approach.

Comparison with Hennessy-Milner logic

We begin with some technicalities on normal forms.

Definition 5.1 We define a class of normal forms $\mathsf{N}\mathcal{L}_\infty \subseteq \mathcal{L}_{\infty\pi}$ inductively as follows:

- $$\frac{\{\phi_i \in \mathsf{N}\mathcal{L}_\infty\}_{i\in I}}{\bigwedge_{i\in I}\phi_i, \bigvee_{i\in I}\phi_i \in \mathsf{N}\mathcal{L}_\infty}$$
- $$\frac{\phi \in \mathsf{N}\mathcal{L}_\infty\ ,\quad a \in \mathsf{Act}}{\Diamond a(\phi) \in \mathsf{N}\mathcal{L}_\infty}$$
- $$\frac{\{\phi_i \in \mathsf{N}\mathcal{L}_\infty\}_{i\in I},\ \ \{a_i \in \mathsf{Act}\}_{i\in I}\ \ \{i \neq j\ \Rightarrow\ a_i \neq a_j\}_{i,j\in I}}{\Box \bigvee_{i\in I} a_i(\phi_i) \in \mathsf{N}\mathcal{L}_\infty}$$

Lemma 5.2 (Normal Forms) *For all $\phi \in \mathcal{L}_{\infty\pi}$, for some $\psi \in \mathsf{N}\mathcal{L}_\infty$:*

$$\mathcal{L}_\infty \vdash \phi = \psi.$$

PROOF. By induction on $\mathsf{md}(\phi)$. We consider the two non-trivial cases.
$\Diamond\phi$: In this case, using the distributive lattice laws there is ϕ' of the form

$$\bigvee_{i\in I}\bigwedge_{j\in J_i} a_{ij}(\phi_{ij})$$

such that $\vdash \phi = \phi'$, and $\mathsf{md}(\phi') \leq \mathsf{md}(\phi)$. By the induction hypothesis, for each ϕ_{ij} there is $\phi'_{ij} \in \mathsf{N}\mathcal{L}_\infty$ such that $\vdash \phi_{ij} = \phi'_{ij}$. Using $(a - \leq)$ and $(\Diamond - \leq)$, we have

$$\vdash \Diamond\phi = \Diamond \bigvee_{i\in I}\bigwedge_{j\in J_i} a_{ij}(\phi_{ij}). \tag{15}$$

Now for each $i \in I$, there are three cases:

1. $J_i = \varnothing$. In this case, $\vdash \Diamond\phi = \Diamond t$, and we can use $(\Diamond - t\ )$ to obtain a normal form.
2. $\exists j_1, j_2 \in J_i,\ a_{j_1} \neq a_{j_2}$. In this case, we can use $(a - \wedge)(ii)$ to delete the i'th disjunct in the RHS of 15.
3. $\{a_{ij} : j \in J_i\} = \{a\}$, for some $a \in \mathsf{Act}$. In this case, we can use $(a - \wedge)(i)$.

In this way, we obtain *either*

$$\vdash \Diamond\phi = t,$$

if case (1) is ever applicable, *or*

$$\vdash \Diamond\phi = \Diamond \bigvee_{i' \in I'} a_{i'}(\psi_{i'}) \quad (\psi_{i'} \in \mathsf{N}\mathcal{L}_\infty).$$

In the latter case, we can apply $(\Diamond - \vee)$ to get a normal form.
$\Box\phi$: Similarly to the previous case, we have

$$\vdash \Box\phi = \Box \bigwedge_{i \in I} \bigvee_{j \in J_i} a_{ij}(\phi_{ij}) \quad (\phi_{ij} \in \mathsf{N}\mathcal{L}_\infty).$$

We can then use $(\Box - \wedge)$ to get

$$\vdash \Box\phi = \bigwedge_{i \in I} \Box \bigvee_{j \in J_i} a_{ij}(\phi_{ij}).$$

Now if we partition each J_i by $\sim_i$, with

$$j \sim_i k \iff a_{ij} = a_{ik} \quad (j, k \in J_i),$$

we have

$$\vdash \Box\phi = \bigwedge_{i \in I} \Box \bigvee_{[j] \in J_i/\sim_i} \left(\bigvee_{k \in [j]} a_{ij}(\phi_{ik}) \right)$$

using the lattice laws; we can then apply $(a - \vee)$ to get a normal form. ∎

Definition 5.3 We define translation functions

$$(\cdot)^* : \mathrm{HML}_\infty \longrightarrow \mathsf{N}\mathcal{L}_\infty \ , \qquad (\cdot)^\dagger : \mathsf{N}\mathcal{L}_\infty \longrightarrow \mathrm{HML}_\infty.$$

$$\begin{array}{lcl}
(\bigwedge_{i \in I} \phi_i)^* & = & \bigwedge_{i \in I} (\phi_i)^* \\
(\bigvee_{i \in I} \phi_i)^* & = & \bigvee_{i \in I} (\phi_i)^* \\
(\langle a \rangle \phi)^* & = & \Diamond a(\phi^*) \\
([a]\phi)^* & = & \Box a((\phi)^*) \vee \bigvee \{ b(t) : b \in \mathsf{Act} \setminus \{a\} \}) \\
(\bigwedge_{i \in I} \phi_i)^\dagger & = & \bigwedge_{i \in I} (\phi_i)^\dagger \\
(\bigvee_{i \in I} \phi_i)^\dagger & = & \bigvee_{i \in I} (\phi_i)^\dagger \\
(\Diamond a(\phi))^\dagger & = & \langle a \rangle (\phi)^\dagger \\
(\Box \bigvee_{i \in I} a_i(\phi_i))^\dagger & = & \bigwedge_{i \in I} [a_i](\phi_i)^\dagger \ \wedge \ \bigwedge \{ [b]f : b \in \mathsf{Act} \setminus \{ a_i : i \in I \} \}
\end{array}$$

The following is easily verified.

Proposition 5.4 *For all $\phi \in \mathrm{HML}_\infty, \psi \in \mathsf{N}\mathcal{L}_\infty$:*

$$\begin{array}{rlcl}
(i) & \mathsf{md}(\phi) & = & \mathsf{md}(\phi^*) \\
(ii) & \mathsf{md}(\psi) & = & \mathsf{md}(\psi^\dagger) \\
(iii) & p \models \phi & \iff & p \models \phi^* \\
(iv) & p \models \psi & \iff & p \models \psi^\dagger.
\end{array}$$

As an immediate consequence of this Proposition together with 5.2, we have

Theorem 5.5 (Comparison Theorem (Infinitary Case)) *For $p, q \in$ Proc in any transition system, $A \subseteq \mathsf{Act}$ and $\lambda \in \mathsf{Ord}$:*

$$\mathcal{L}_\infty^{(A,\lambda)}(p) \subseteq \mathcal{L}_\infty^{(A,\lambda)}(q) \iff \mathrm{HML}_\infty^{(A,\lambda)}(p) \subseteq \mathrm{HML}_\infty^{(A,\lambda)}(q).$$

Thus in the infinitary case, $\mathcal{L}_\infty$ determines the same preorder on processes as HML_∞. However, when Act is infinite this does *not* cut down to a corresponding result for the finitary case, since our translation functions introduce infinite disjunctions in translating $[a]$, and infinite conjunctions in translating $\Box$, even for finite formulas. General considerations on observability [Abr88a] suggest that the introduction of infinite conjunctions is more serious, and indicates a weakness of expressive power in HML_∞ as an "observational logic". This is in keeping with our remarks at the end of Section 2. In fact, our translation functions suggest an appropriate way of extending HML_∞ so as to render it equivalent to $\mathcal{L}_\omega$. This will be the content of a second Comparison Theorem which we will prove later in this section, when we have some additional machinery at our disposal.

Characterisation Theorems

Combining the Comparison Theorem with the Modal Characterisation Theorem 2.4, we have:

Theorem 5.6 (Characterisation Theorem for $\mathcal{L}_\infty$) *With notation as in the previous Theorem,*

$$p \sqsubseteq\!\!\!\sim_\lambda q \iff \mathcal{L}_\infty^{(\mathsf{Act},\lambda)}(p) \subseteq \mathcal{L}_\infty^{(\mathsf{Act},\lambda)}(q)$$

and therefore

$$p \sqsubseteq\!\!\!\sim^B q \iff \mathcal{L}_\infty(p) \subseteq \mathcal{L}_\infty(q).$$

We now turn to the question of finding a Characterisation Theorem for $\mathcal{L}_\omega$. Intuitively, $\mathcal{L}_\omega$ represents finitely observable properties of processes, hence should correspond to the "finitely observable part" of bisimulation. If we accept the finite synchronisation trees ST_ω as a suitable notion of *finite process*, we can use them to determine the *algebraic* part of the bisimulation preorder, in the sense e.g. of [Gue81].

Definition 5.7 The *finitary preorder* $\sqsubseteq\!\!\!\sim^F$ is defined on any transition system by:

$$p \sqsubseteq\!\!\!\sim^F q \equiv \forall t \in \mathsf{ST}_\omega .\, t \sqsubseteq\!\!\!\sim^B p \Rightarrow t \sqsubseteq\!\!\!\sim^B q.$$

Our aim is to prove

Theorem 5.8 (Characterisation Theorem for $\mathcal{L}_\omega$) *With notation as in the previous Theorem,*

$$p \sqsubseteq\!\!\!\sim^F q \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

We will need a few auxiliary results which also have some independent interest.

Definition 5.9 The *height* of a synchronisation tree is defined by:

$$\mathsf{ht}(\sum_{i \in I} a_i t_i\ [+\Omega]) = \sup \{\mathsf{ht}(t_i) : i \in I\} + 1$$

Lemma 5.10 *For any synchronisation tree $T \in \mathsf{ST}_\infty$, $\mathsf{ht}(T) < \lambda$ implies*

$$T \precsim^B p \iff T \precsim_\lambda p.$$

Proof. The left-to-right implication is immediate; the converse is an easy induction on $\mathsf{ht}(T)$. ∎

In particular, we see that for a finite synchronisation tree $t \in \mathsf{ST}_\omega$, $t \precsim^B p \;\Leftrightarrow\; t \precsim_\omega p$. Thus we have the inclusions

$$\precsim^B \subseteq \precsim_\omega \subseteq \precsim^F.$$

In general, these inclusions are strict.

Examples

(1) $\precsim^B \neq \precsim_\omega$.

$$p \equiv a^\omega + \Omega, \quad q \equiv \sum_{k \in \omega} a^k \mathbb{O} \; + \Omega$$

Then $p \precsim_\omega q$, but $p \not\precsim_{\omega+1} q$.

(2) $\precsim_\omega \neq \precsim^F$.

$$\begin{aligned} p &\equiv a(\sum_{n \in \omega} b_n \mathbb{O} + \Omega) + \Omega \\ q &\equiv \sum_{n \in \omega} a(\sum_{m \in \omega \setminus \{n\}} b_n \mathbb{O} + \Omega) + \Omega \end{aligned}$$

Then $p \precsim^F q$, but $p \not\precsim_2 q$.

These examples gain in significance because all the processes involved can be defined in finitary calculi, in particular SCCS, as we shall see in the next section.

Lemma 5.11 (Sort Lemma) *In any transition system, let $p, q \in$ Proc, $\mathsf{sort}(p) \subseteq A \subseteq \mathsf{Act}$, $\lambda \in \mathsf{Ord}$. Then*

$$p \not\precsim_\lambda q \implies \mathcal{L}_\infty^{(A,\lambda)}(p) \not\subseteq \mathcal{L}_\infty^{(A,\lambda)}(q).$$

Proof. By induction on λ. We assume $p \not\precsim_\lambda q$, and must construct $\phi \in \mathcal{L}_\infty^{(A,\lambda)}(p) \setminus \mathcal{L}_\infty^{(A,\lambda)}(q)$. There are three cases.

(1) $p \xrightarrow{a} p'$ and for all q', $q \xrightarrow{a} q'$ implies $p' \not\precsim_\alpha q'$ for some $\alpha < \lambda$. By induction hypothesis, for each such q there is $\phi_{q'} \in \mathcal{L}_\infty^{(A,\alpha)}(p') \setminus \mathcal{L}_\infty^{(A,\alpha)}(q')$. Now define

$$\phi \;\equiv\; \Diamond a (\bigwedge \{\phi_{q'} : q \xrightarrow{a} q'\}).$$

(2) $p\downarrow$ and $q\uparrow$. Let $\phi \equiv \Box \bigvee \{a(t) : \exists p'. \, p \xrightarrow{a} p'\}$.

(3) $p\downarrow$, $q\downarrow$, $q \xrightarrow{a} q'$, and for all p', $p \xrightarrow{a} p'$ implies $p' \not\precsim_\alpha q'$ for some $\alpha < \lambda$. Define $\phi_{p'}$ similarly to case (1). Then define

$$\phi \;\equiv\; \Box (\bigvee \{a(\phi_{p'}) : p \xrightarrow{a} p'\} \;\vee\; \bigvee \{b(t) : b \neq a \;\&\; \exists r. \, p \xrightarrow{a} r\}). \quad ∎$$

Note that this result is stronger than the Modal Characterisation Theorem 2.4 for Hennessy-Milner logic, since we only require $\mathsf{sort}(p) \subseteq A$. This is significant in the light of the example at the end of Section 2.

Proposition 5.12 *For all $t \in \mathsf{ST}_\omega$:*

$$t \lesssim^B p \iff \mathcal{L}_\omega(t) \subseteq \mathcal{L}_\omega(p).$$

Proof. Combining 5.10 and 5.11, we see that

$$t \lesssim^B p \iff \mathcal{L}_\infty^{(A,k)}(t) \subseteq \mathcal{L}_\infty^{(A,k)}(p),$$

where $A = \mathsf{sort}(t)$ and $k = \mathsf{ht}(t)$. Since A and k are both finite, $\mathcal{L}_\infty^{(A,k)}$ is finite up to logical equivalence (i.e. the Lindenbaum algebra is finite). Thus each formula in $\mathcal{L}_\infty^{(A,k)}$ is equivalent to one in $\mathcal{L}_\omega$, and the proposition is proved. ∎

We need one more auxiliary result, which will in fact be a consequence of our work on SCCS in the next section. Firstly, we define a map from coprime normal forms to finite synchronisation trees

$$\mathsf{st} : \mathrm{CNF} \to \mathsf{ST}_\omega$$

as follows:

$$\begin{array}{lcl} \mathsf{st}(\bigwedge_{i\in I} \Diamond a_i(\phi_i)) & \equiv & \sum_{i\in I} a_i \mathsf{st}(\phi_i) + \Omega \\ \mathsf{st}(\Box \bigvee_{i\in I} a_i(\phi_i) \ \wedge\ \bigwedge_{j\in J} \Diamond b_j(\psi_i)) & \equiv & \sum_{i\in I} a_i \mathsf{st}(\phi_i) + \sum_{j\in J} b_j \mathsf{st}(\psi_j). \end{array}$$

Now analogously to 4.6 we have

Proposition 5.13 *For all ϕ in CNF, and $p \in \mathrm{Proc}$ in any transition system:*

$$p \models \phi \iff \mathsf{st}(\phi) \lesssim^B p.$$

The proof is entirely analogous to 4.6.

Proof of 5.8

Firstly, $\mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q)$ implies $p \lesssim^F q$, by 5.12. For the converse, assume $p \lesssim^F q$ and $p \models \phi$, $(\phi \in \mathcal{L}_\omega)$. By the SDNF Theorem 4.5,

$$\begin{array}{lll} \bullet & \vdash \phi = \bigvee_{i\in I} \phi_i \quad (\phi_i \in \mathrm{CNF}) & \\ \Longrightarrow & \exists i \in I.\, p \models \phi_i & \\ \Longrightarrow & \mathsf{st}(\phi_i) \lesssim^B p & 5.13 \\ \Longrightarrow & \mathsf{st}(\phi_i) \lesssim^B q & p \lesssim^F q \\ \Longrightarrow & q \models \phi_i & 5.13 \\ \Longrightarrow & q \models \phi. & \blacksquare \end{array}$$

Finitary Transition Systems

We now embark on our next topic. The various finiteness conditions on transition systems defined in section 2 reflect attempts to capture features of finitary processes. However, none of these conditions seems to capture exactly the right class of systems unless we make some unwelcome assumptions such as that the set of actions is finite. We shall adopt what seems to be a novel approach, of using our program logic to axiomatize a class of systems which we propose as the finitary ones. Our axiomatisation consists of two schemes over $\mathcal{L}_\infty$.

Notation. $\mathsf{Fin}(I)$ is the set of finite subsets of I.

- The axiom scheme of *bounded non-determinacy*:

$$\text{(BN)}\ \Box \bigvee_{i \in I} \phi_i \leq \bigvee_{J \in \mathsf{Fin}(I)} \Box \bigvee_{j \in J} \phi_j \quad (\phi_i \in \mathcal{L}_\omega\ ).$$

- The axiom scheme of *finite approximability*:

$$\text{(FA)}\ \bigwedge_{J \in \mathsf{Fin}(I)} \Diamond \bigwedge_{j \in J} \phi_j \leq \Diamond \bigwedge_{i \in I} \phi_i \quad (\phi_i \in \mathcal{L}_\omega\ ).$$

Note that these axioms are duals. Since the opposite entailments are theorems of $\mathcal{L}_\infty$, we shall in fact use (BN) and (FA) to denote the corresponding *equations*. The axioms could equivalently be formulated as: $\Box$ preserves directed joins, $\Diamond$ preserves filtered meets.

What are the intuitions behind these axioms? (BN) is (thinking of each process as the set of its capabilities and each ϕ_i as an open set) exactly a statement of *compactness*; the link between compactness and the computational notion of bounded non-determinacy is well-known from the literature on powerdomains [Plo81, Smy83].

The axiom of finite approximability is less familiar from either the topological or the computer science literature. It is best understood as a logical (or localic) expression of the idea that only *closed* sets are taken as elements of a finitary powerdomain construction (or, better put, that from the point of view of finite observability we cannot distinguish between a set and its closure). The best way to get a more precise understanding is probably to read the proof of the next Theorem.

The duality between the two axioms is reminiscent of the discussion of finite *breadth* (BN) and finite *length* (FA) limitations of testing in [Abr83b].

Definition 5.14 A transition system is *finitary* if it satisfies (all instances of) (BN) and (FA). The class of finitary transition systems is denoted **FTS**.

As a first step, we shall give a substantive example of a finitary transition system. As we will see, it is actually the best possible example.

Theorem 5.15 *$\mathcal{D}$ is a finitary transition system.*

PROOF. By the Duality Theorem 4.3, we have a map

$$[\![\cdot]\!] : \mathcal{L}_{\omega\pi} \longrightarrow K\Omega(\mathcal{D})$$

$$[\![\phi]\!] \equiv \{d \in \mathcal{D} : d \models \phi\}.$$

Now for $d \in \mathcal{D}$,

$$d \models \Box \bigvee_{i \in I} \phi_i \implies d \models \bigvee_{J \in \mathsf{Fin}(I)} \Box \bigvee_{j \in J} \phi_j$$

is just the statement

$$d \subseteq \bigcup_{i \in I} O_i \implies \exists J \in \mathsf{Fin}(I).\, d \subseteq \bigcup_{j \in J} O_j,$$

where $O_i = [\![\phi_i]\!]$, i.e. that d is compact as a subset of $\sum_{a \in \mathsf{Act}} \mathcal{D}$. Since $d \in \mathcal{D} \cong P^0[\sum_{a \in \mathsf{Act}} \mathcal{D}]$, and elements of the Plotkin powerdomain are Scott-compact subsets of the base domain ([Plo81]), this proves that $\mathcal{D}$ satisfies (BN).

Next we show that $\mathcal{D}$ satisfies (FA). Since there are only countably many logically inequivalent formulae in $\mathcal{L}_\omega$, it suffices to prove the following:

- Given a sequence $\{U_n\}$ of compact-open subsets of $\mathcal{D}$, with $U_n \supseteq U_{n+1}$ $(n \in \omega)$, and an element $d \in \mathcal{D}$ such that $d \cap U_n \neq \varnothing$ $(n \in \omega)$, then $d \cap \bigcap_{n\in\omega} U_n \neq \varnothing$.

(The alternative case for $d \models \Diamond U_n$, namely $\perp \in U_n$ for all n, is trivial.)

Since each U_n is compact-open, it has the form $\uparrow B_n$, where B_n is a finite subset of $\mathcal{K}(\mathcal{D})$. Also, $B_n \sqsubseteq_u B_{n+1}$, where

$$X \sqsubseteq_u Y \;\equiv\; \forall y \in Y.\, \exists x \in X.\, x \sqsubseteq y \quad (X, Y \subseteq \mathcal{D}\ ).$$

Now define

$$C_n \;\equiv\; \{b \in B_n : \exists x \in d.\, b \sqsubseteq x\} \quad (n \in \omega).$$

Since $d \cap U_n \neq \varnothing$, $C_n \neq \varnothing$ for all n. Also, $C_n \sqsubseteq_u C_{n+1}$. Thus by König's Lemma in the form given e.g. in [Niv81], there is a sequence $\{c_n\}$ with $c_n \sqsubseteq c_{n+1}$ and $c_n \in C_n$. Now define

$$e_n \;\equiv\; \{\!|c_n|\!\} \uplus \{\!|\perp|\!\} \quad (n \in \omega).$$

Clearly $e_n \sqsubseteq e_{n+1}$ and $e_n \sqsubseteq d$ for all n, whence $\bigsqcup e_n \sqsubseteq d$. But $\bigsqcup c_n \in \bigsqcup e_n$ (using the description of least upper bounds of chains in the Plotkin powerdomain given in [Plo76, Theorem 8]), and so for some $x \in d$, $\bigsqcup c_n \sqsubseteq x$. Since $\bigsqcup c_n \in U_n$ for all n, $d \cap \bigcap_{n\in\omega} U_n \neq \varnothing$, and the proof is complete. ∎

We now draw some striking consequences from the finitary axioms.

Definition 5.16 A formula $\phi \in \mathcal{L}_\infty$ is in *finitary normal form* if it has the form

$$\bigwedge_{i\in I} \bigvee_{j\in J_i} \phi_{ij} \quad (\phi_{ij} \in \mathcal{L}_\omega).$$

Lemma 5.17 *For each $\phi \in \mathcal{L}_\infty$, for some finitary normal form ψ:*

$$\textit{(BN) + (FA)} \vdash \phi = \psi.$$

Proof. An easy induction on $\mathsf{ht}(\phi)$. ∎

Proposition 5.18 *In any finitary transition system $\mathcal{T}$, for all $p, q \in$ Proc:*

$$\mathcal{L}_\infty(p) \subseteq \mathcal{L}_\infty(q) \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

Proof. The left to right implication is immediate. For the converse, suppose $\mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q)$, and $p \models \phi$, $(\phi \in \mathcal{L}_\infty)$. By 5.17,

$$\text{(BN)} + \text{(FA)} \vdash \phi = \bigwedge_{i\in I} \bigvee_{j\in J_i} \phi_{ij} \quad (\phi_{ij} \in \mathcal{L}_\omega)$$

hence since $\mathcal{T} \models$ (BN) + (FA), $\mathcal{T} \models \phi = \bigwedge_{i\in I} \bigvee_{j\in J_i} \phi_{ij}$, and

$$\begin{array}{ll} \bullet & p \models \bigwedge_{i\in I} \bigvee_{j\in J_i} \phi_{ij} \\ \Longrightarrow & \forall i \in I.\, \exists j \in J_i.\, p \models \phi_{ij} \\ \Longrightarrow & \forall i \in I.\, \exists j \in J_i.\, q \models \phi_{ij} \\ \Longrightarrow & q \models \bigwedge_{i\in I} \bigvee_{j\in J_i} \phi_{ij} \\ \Longrightarrow & q \models \phi. \quad \blacksquare \end{array}$$

Theorem 5.19 (Finitary Characterisation Theorem) *With notation as in the previous Proposition:*

$$p \sqsubseteq^B_{\sim} q \iff p \sqsubseteq_{\omega} q \iff p \sqsubseteq^F_{\sim} q \iff \mathcal{L}_{\omega}(p) \subseteq \mathcal{L}_{\omega}(q).$$

PROOF. Combine Theorems 5.6, 5.8 and 5.18. ∎

In order to continue our study of finitary transition systems, we need to introduce some notions from our final topic of this section.

Universal Semantics

Given any transition system and $p \in \mathrm{Proc}$, it is easy to see that $\mathcal{L}_{\omega}(p) \subseteq \mathcal{L}_{\omega}$ satisfies the axioms of a prime filter; hence we have a map

$$\mathcal{L}_{\omega}(\cdot) : \mathrm{Proc} \longrightarrow \mathsf{Spec}\ \mathcal{L}_{\omega} \ .$$

If we compose this with the isomorphism $\mathsf{Spec}\ \mathcal{L}_{\omega} \cong \mathcal{D}$ from the Duality Theorem 4.3, we get a map

$$[\![\cdot]\!] : \mathrm{Proc} \longrightarrow \mathcal{D}$$

which takes each process to an element of our domain. This map can be regarded as a *syntax-free denotational semantics*; it is *universal* since it is defined on every transition system.

Theorem 5.20 (Universal Semantics) *For any transition system $\mathcal{T}$ with $p, q \in \mathrm{Proc}$:*

$$(i) \quad p \sqsubseteq^F_{\sim} q \iff [\![p]\!] \sqsubseteq [\![q]\!]$$

$$(ii) \quad p \sim^F [\![p]\!].$$

If $\mathcal{T}$ is finitary, then:

$$(iii) \quad p \sqsubseteq^B_{\sim} q \iff [\![p]\!] \sqsubseteq [\![q]\!]$$

$$(iv) \quad p \sim^B [\![p]\!].$$

PROOF. Clearly (i) follows from (ii), and (iii) from (iv). Now $\mathcal{L}_{\omega}(p) = \mathcal{L}_{\omega}([\![p]\!])$; and so (ii) follows from 5.8; while (iv) follows from 5.19. ∎

We can think of 5.20 as a *full abstraction theorem* [Mil75, Plo77, Mil77] for our semantics; it says that every transition system (finitary transition system) can be embedded in $\mathcal{D}$ with as much identification as possible modulo the finitary equivalence (bisimulation).

Since $\mathcal{D}$ can itself be viewed as a transition system, we can tie things up even more neatly. Let **TS** be the category with objects the transition systems, and morphisms $\mathcal{T}_1 \to \mathcal{T}_2$ maps

$$f : \mathrm{Proc}_1 \to \mathrm{Proc}_2$$

for which

$$\mathcal{L}_{\omega}(p) = \mathcal{L}_{\omega}(f(p)) \quad (p \in \mathrm{Proc}_1).$$

It is clear that for such f

$$p \sqsubseteq^F_{\sim} q \iff f(p) \sqsubseteq^F_{\sim} f(q),$$

and if $\mathcal{T}_1$ and $\mathcal{T}_2$ are finitary,

$$p \sqsubseteq^B_{\sim} q \iff f(p) \sqsubseteq^B_{\sim} f(q).$$

Now we have

Theorem 5.21 (Final Algebra Theorem) *$\mathcal{D}$ is final in* **TS***, and also in the subcategory* **FTS** *of finitary transition systems.*

PROOF. All we need to show is that the semantic map $[\![\cdot]\!]$ is the unique morphism from a transition system to $\mathcal{D}$. But for $d_1, d_2 \in \mathcal{D}$,

$$\begin{array}{rcll} \mathcal{L}_\omega(d_1) \subseteq \mathcal{L}_\omega(d_2) & \iff & \forall U \in K\Omega(\mathcal{D}).\, d_1 \in U \;\Rightarrow\; d_2 \in U & \text{by (4.3)} \\ & \iff & d_1 \sqsubseteq d_2 & \text{since } \mathcal{D} \text{ is coherent,} \end{array}$$

which gives uniqueness. ∎

Finitary Transition Systems Resumed

Firstly, some conditions equivalent to finitariness.

Proposition 5.22 *For any transition system $\mathcal{T}$, the following conditions are equivalent:*

(i) $\mathcal{T}$ *is finitary*

(ii) $\forall p \in \mathrm{Proc}.\, p \sim^B [\![p]\!]$

(iii) $\precsim^B = \precsim^F$ *in the combined system $\mathcal{T} + \mathcal{D}$ (disjoint union).*

PROOF. $(i) \implies (ii)$ is 5.20 (iv); $(ii) \implies (iii)$ since $\mathcal{D}$ is finitary.
$(ii) \implies (i)$. Suppose that $\mathcal{T}$ is not finitary, in particular that (BN) fails; i.e. that for some $p \in \mathrm{Proc}$,

$$p \models \Box \bigvee_{i \in I} \phi_i \quad (\phi_i \in \mathcal{L}_\omega)$$

and $\forall J \in \mathsf{Fin}(I).\, p \not\models \bigvee_{j \in J} \phi_j$. Since $\mathcal{L}_\omega(p) = \mathcal{L}_\omega([\![p]\!])$, and each $\bigvee_{j\in J} \phi_j \in \mathcal{L}_\omega$, $[\![p]\!] \not\models \bigvee_{j \in J} \phi_j$ for all $J \in \mathsf{Fin}(I)$; hence since $[\![p]\!] \in \mathcal{D}$ and $\mathcal{D}$ is finitary, $[\![p]\!] \not\models \Box \bigvee_{i \in I} \phi_i$. Thus $\mathcal{L}_\infty([\![p]\!]) \neq \mathcal{L}_\infty(p)$, and so by 5.6 $p \not\sim^B [\![p]\!]$. The case when (FA) fails is similar.
$(iii) \implies (ii)$. Suppose for some p, $p \not\sim^B [\![p]\!]$. Then since $p \sim^F [\![p]\!]$ by 5.20 (ii), $\precsim^B \neq \precsim^F$. ∎

Note that in part (iii) of this Proposition we have "added in" $\mathcal{D}$ to the given transition system $\mathcal{T}$. This is to overcome the problem that there may not be enough processes in $\mathcal{T}$ alone to cause $\precsim^B = \precsim^F$ to fail.

Now we relate some of the finitariness conditions of Section 2 to our axioms.

Proposition 5.23 *(i) Weakly finite branching is equivalent to weakly image finite plus weakly initials finite.*
(ii) Weakly finite branching implies (BN).
(iii) (BN) implies weakly initials finite.
(iv) (BN) + (FA) do not *imply weakly image finite.*

PROOF. (i). Easy.
(ii). Suppose $p \models \Box \bigvee_{i\in I} \phi_i$. $(\bigvee_{i \in I} \phi_i)\!\uparrow \;\Leftrightarrow\; \exists i \in I.\, \phi_i\!\uparrow$, in which case $\vdash \phi_i = t$, and the conclusion is trivial. Otherwise, $p\!\downarrow$, and so $C(p)$ is finite, say

$$C(p) = \{\langle a_1, p_1 \rangle, \ldots, \langle a_n, p_n \rangle\}.$$

Then for each k with $1 \leq k \leq n$, $\langle a_k, p_k \rangle \models \phi_{i_k}$ for some $i_k \in I$, and so $p \models \Box \bigvee_{j \in J} \phi_j$, where $J = \{i_1, \ldots, i_n\}$.
(iii). Assume (BN) and $p\downarrow$. Then $p \models \Box \bigvee_{a \in \mathsf{Act}} a(t\ )$, and so by (BN)

$$p \models \bigvee_{J \in \mathsf{Fin}(\mathsf{Act})} \Box \bigvee_{a \in J} a(t\ ),$$

which says exactly that p has a finite set of initial actions.
(iv). $\sum_{n \in \omega} a^n + a^\omega$ is in $\mathcal{D}$. ∎

All the usual finitary calculi are weakly finite branching, and so satisfy (BN). However, in general these calculi do *not* satisfy (FA) (analogously to the fact that generating trees over domains do not yield closed sets, although they always yield compact ones; cf. [Plo81]). As a standard counterexample, define

$$\begin{array}{lcl} p & \equiv & \sum_{n \in \omega} a^n \mathbb{O} + \Omega \\ \phi_0 & \equiv & t \\ \phi_{k+1} & \equiv & a(\Diamond \phi_k). \end{array}$$

Then for all $J \in \mathsf{Fin}(\omega)$, $p \models \Diamond \bigwedge_{j \in J} \phi_j$, but $p \not\models \Diamond \bigwedge_{i \in \omega} \phi_i$.

Thus if p can be defined in our calculus, it does not satisfy (FA). Since p *can* be defined in CCS, SCCS (see next section), etc., these calculi are not finitary transition systems according to Definition 5.14. However, we can take the view that if we only take account of *observable* information via the semantics $[\![\cdot]\!]$, we have collapsed the given system into a finitary one which will actually, by Theorems 5.20 and 5.21, be isomorphic to a subsystem (or, topologically, a subspace) of $\mathcal{D}$.

Comparison Theorems Resumed

We now return to the question of finding a suitable correspondence between the finitary parts of HML and $\mathcal{L}$. As confirmation of our claim that HML_ω is unsatisfactory, we have:
Observation. HML_ω does not characterise $\lesssim^F$.

In fact, 2.7 provides a counter-example since, with the notation used there, $p \not\lesssim^F q$ while $\mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q)$.

We can get an idea of how to extend HML_ω by inspection of the translation functions 5.3. Although $(\cdot)^\dagger$ introduces infinitary conjunctions, these are of a special kind, for which a finitary counterpart can be found.

Definition 5.24 HML^+ is the extension of HML_ω with additional atomic fomulae of the form

$$\mathsf{init}(A) \quad (A \in \mathsf{Fin}(\mathsf{Act})).$$

The definition of the satisfaction relation is extended by

$$p \models \mathsf{init}(A) \equiv p\downarrow \ \& \ \{a \in \mathsf{Act} : \exists q.\, p \xrightarrow{a} q\} \subseteq A.$$

We can now modify the translation function $(\cdot)^\dagger$ as follows:

$$(\Box \bigvee_{i \in I} a_i(\phi_i))^\dagger \ \equiv \ \bigwedge_{i \in I} [a_i](\phi_i)^\dagger \ \wedge \ \mathsf{init}(\{a_i : i \in I\}).$$

Proposition 5.4 clearly still holds with this modification, and $(\cdot)^{\dagger}$ now cuts down to a function

$$\mathsf{N}\mathcal{L}_{\omega} \longrightarrow \mathrm{HML}^{+}.$$

There is still a mismatch in the other direction, since $(\cdot)^{*}$ introduces infinite disjunctions. To overcome this, we have to make the assumption that the transition system satisfies (BN)—a mild one, as 5.23 and the ensuing discussion shows.

Let $\mathcal{L}_{\bigvee\infty}$ be the sublanguage of $\mathcal{L}_{\infty}$ obtained by the restriction to finite *conjunctions* (but with infinite disjunctions still allowed).

Proposition 5.25 *In any transition system satisfying (BN), for all $p, q \in \mathrm{Proc}$:*

$$\mathcal{L}_{\bigvee\infty}(p) \subseteq \mathcal{L}_{\bigvee\infty}(q) \iff \mathcal{L}_{\omega}(p) \subseteq \mathcal{L}_{\omega}(q).$$

PROOF. Just like 5.18. ∎

Clearly, $(\cdot)^{*}$, extended by the clause

$$(\mathsf{init}(A))^{*} \equiv \Box \bigvee \{a(t \ : a \in A\}$$

cuts down to a function

$$\mathrm{HML}^{+} \longrightarrow \mathsf{N}\mathcal{L}_{\bigvee\infty}.$$

We thus arrive at our

Theorem 5.26 (Comparison Theorem (Finitary Case)) *With notation as in the previous Proposition:*

$$\mathrm{HML}^{+}(p) \subseteq \mathrm{HML}^{+}(q) \iff \mathcal{L}_{\omega}(p) \subseteq \mathcal{L}_{\omega}(q).$$

6 Full Abstraction for SCCS

So far, we have worked with abstract transition systems, in a syntax-free fashion. This degree of abstraction carries a price; we lose compositionality. Indeed, we need syntax to *define* compositionality. Accordingly, in this Section we turn to a particular transition system specified by an algebraic syntax, namely Milner's SCCS [Mil83]. We equip our domain $\mathcal{D}$ with a continuous algebraic structure corresponding to the signature of SCCS. Our main result is that the resulting denotational semantics for SCCS is *fully abstract* [Mil75, Plo77] with respect to bisimulation for finite terms, and with respect to the finitary preorder for recursive terms. As a by-product we will show that $\mathcal{D}$ is isomorphic to Hennessy's term model [Hen81], and hence obtain a complete axiomatisation of its equational theory as an immediate consequence of Hennessy's results.

Our choice of SCCS is for illustrative purposes, because it is simple and yet expressive. Similar accounts could be given for CCS [Mil80], MEIJE [AB84], ACP [BK84], etc. Note, however, that our semantics is fully abstract with respect to the *strong* congruence in Milner's terminology [Mil83], where all actions are observable. A corresponding treatment of *observation equivalence* [HM85], where unobservable actions are factored out, has been obtained very recently by the author, and will be described elsewhere.

We begin by recalling some basic definitions on SCCS from [Mil83, Hen81]. We assume familiarity with basic notions of universal algebra; see e.g. [GTW78, EM85].

We fix a set of actions Act, which we assume comes equipped with an *abelian monoid* structure comprising

- an associative, commutative binary operation which we denote by juxtaposition, e.g. ab
- a unit 1.

The (one-sorted) signature Σ of SCCS is then defined as follows:

Definition 6.1 $\Sigma = \{\Sigma_n\}_{n\in\omega}$, where Σ_n is the set of operation symbols of arity n in Σ.

$$\begin{aligned}
\Sigma_0 &\equiv \{\mathbb{O}, \Omega\} \\
\Sigma_1 &\equiv \{a_ : a \in \mathsf{Act}\} \cup \{_\restriction A : A \subseteq \mathsf{Act}\} \\
&\quad \cup \{_[S] : S \text{ is a monoid endomorphism on } \mathsf{Act}\} \\
\Sigma_2 &\equiv \{+, \times\} \\
\Sigma_n &\equiv \varnothing, \quad n > 2.
\end{aligned}$$

Thus our version of SCCS only has *finite* sums (in contrast with [Mil83]), and has a constant for the undefined process as in [Hen81].

We define the subsignature $\Sigma' \subseteq \Sigma$ to be obtained by omitting the *restriction* operators $\_\restriction A$, the *relabelling* operators $\_[S]$, and the *synchronous product* operator $\times$, leaving only the *nullary sum* $\mathbb{O}$, the *binary sum* $+$, *prefixing* $a\_$, and the *undefined* process Ω.

We take the *finite processes* of SCCS to be the terms over the signature Σ, i.e. the elements of the term algebra T_Σ. Evidently, we can take the elements of $T_{\Sigma'}$ as notations for the finite synchronisation trees ST_ω.

Definition 6.2 (Operational Semantics) We make T_Σ into a transition system by defining the transition relation and divergence predicate in a syntax-directed way, as the *least* relations satisfying the following axioms and rules:

$$(D\Omega)\ \ \Omega\uparrow$$

$$(D+L)\ \ \frac{t_1\uparrow}{(t_1+t_2)\uparrow} \qquad (D+R)\ \ \frac{t_2\uparrow}{(t_1+t_2)\uparrow}$$

$$(D\restriction)\ \ \frac{t\uparrow}{(t\restriction A)\uparrow} \qquad (DS)\ \ \frac{t\uparrow}{t[S]\uparrow}$$

$$(D\times L)\ \ \frac{t_1\uparrow}{t_1\times t_2\uparrow} \qquad (D\times R)\ \ \frac{t_2\uparrow}{t_1\times t_2\uparrow}$$

$$(Ta)\ \ at \xrightarrow{a} t$$

$$(T+L)\ \ \frac{t_1 \xrightarrow{a} t_1'}{t_1+t_2 \xrightarrow{a} t_1'} \qquad (T+R)\ \ \frac{t_2 \xrightarrow{a} t_2'}{t_1+t_2 \xrightarrow{a} t_2'}$$

$$(T\restriction)\ \ \frac{t \xrightarrow{a} t',\ a \in A}{t\restriction A \xrightarrow{a} t'\restriction A} \qquad (TS)\ \ \frac{t \xrightarrow{a} t'}{t[S] \xrightarrow{Sa} t'[S]}$$

$$(T\times)\ \ \frac{t_1 \xrightarrow{a} t_1' \ \ t_2 \xrightarrow{b} t_2'}{t_1\times t_2 \xrightarrow{ab} t_1'\times t_2'}$$

For an illuminating discussion of the conceptual basis for these and related axioms, see [Mil86].

We now have a transition system $(T_\Sigma, \mathsf{Act}, \rightarrow, \uparrow)$ implicitly defined by 6.2. The following proposition gives a more explicit description of this system.

Proposition 6.3 *For all $t, t_1, t_2 \in T_\Sigma$:*

$$\begin{array}{rlcl}
(i)(a) & \mathbb{O}\downarrow & & (b)\ \ \mathbb{O} \stackrel{a}{\not\rightarrow} \\
(ii)(a) & \Omega\uparrow & & (b)\ \ \Omega \stackrel{a}{\not\rightarrow} \\
(iii)(a) & at\downarrow & & \\
(b) & at_1 \xrightarrow{b} t_2 & \iff & b = a \ \&\ t_1 = t_2 \\
(iv)(a) & (t_1+t_2)\uparrow & \iff & t_1\uparrow \text{ or } t_2\uparrow \\
(b) & (t_1+t_2) \xrightarrow{a} t & \iff & t_1 \xrightarrow{a} t \text{ or } t_2 \xrightarrow{a} t \\
(v)(a) & (t\restriction A)\uparrow & \iff & t\uparrow \\
(b) & t_1\restriction A \xrightarrow{a} t_2 & \iff & \exists t.\, t_1 \xrightarrow{a} t \ \&\ t_2 = t\restriction A \ \&\ a \in A \\
(vi)(a) & t[S]\uparrow & \iff & t\uparrow \\
(b) & t_1[S] \xrightarrow{a} t_2 & \iff & \exists b, t.\, t_1 \xrightarrow{b} t \ \&\ t_2 = t[S] \ \&\ a = Sb \\
(vii)(a) & (t_1\times t_2)\uparrow & \iff & t_1\uparrow \text{ or } t_2\uparrow \\
(b) & t_1\times t_2 \xrightarrow{a} t & \iff & \exists t_1', t_2', b_1, b_2.\, t_i \xrightarrow{b_i} t_i'\ (i=1,2) \\
 & & & \&\ t = t_1'\times t_2' \ \&\ a = b_1 b_2.
\end{array}$$

PROOF. By induction on the length of proofs of $t\uparrow$ and $t_1 \xrightarrow{a} t_2$. ∎

Now given any Σ-algebra $\mathcal{A}$, by initiality of T_Σ there is a unique Σ-homomorphism

$$[\![\cdot]\!]^{\mathcal{A}} : T_\Sigma \longrightarrow \mathcal{A},$$

which is just another notation for a compositional denotational semantics as in [MS76], [Sto77], [Gor79]. Thus to form a denotational semantics $[\![\cdot]\!]^{\mathcal{D}}$ based on our domain $\mathcal{D}$, it suffices to define each operation in Σ as a function of the appropriate arity over $\mathcal{D}$. We shall in fact define the operations so that they are *continuous* over $\mathcal{D}$.

Definition 6.4 We specify a Σ-structure on $\mathcal{D}$:

$$\begin{array}{rrcl}
(i) & \mathbb{O}^{\mathcal{D}} & \equiv & \emptyset \\
(ii) & \Omega^{\mathcal{D}} & \equiv & \{\!|\perp|\!\} \\
(iii) & a_^{\mathcal{D}} & \equiv & \lambda d \in \mathcal{D}.\{\!|\langle a, d\rangle|\!\} \\
(iv) & +^{\mathcal{D}} & \equiv & \uplus
\end{array}$$

Restriction:

$$(v)\ \ (\_\lceil A)^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D} \to \mathcal{D}].\ \biguplus \circ P^0(g_A\Phi)$$

where

$$g_A : [\mathcal{D} \to \mathcal{D}] \to [\sum_{a \in \mathsf{Act}} \mathcal{D} \to \mathcal{D}]$$

is defined by

$$\begin{array}{rcl}
g_A\Phi\perp & = & \{\!|\perp|\!\} \\
g_A\Phi\langle a, d\rangle & = & \begin{cases} \{\!|\langle a, \Phi d\rangle|\!\} & \text{if } a \in A \\ \emptyset & \text{otherwise} \end{cases}
\end{array}$$

(i.e.

$$g_A\Phi = \coprod_{a \in A} \lambda d \in \mathcal{D}.\{\!|\langle a, \Phi d\rangle|\!\} \amalg \coprod_{a \in \mathsf{Act}\setminus A} \lambda d \in \mathcal{D}.\emptyset,$$

where $\coprod$ is "source tupling" [WBT85]).

Relabelling:

$$(vi)\ \ (\_[S])^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D} \to \mathcal{D}].\ P^0(g_S\Phi)$$

where

$$g_S : [\mathcal{D} \to \mathcal{D}] \to [\sum_{a \in \mathsf{Act}} \mathcal{D} \to \sum_{a \in \mathsf{Act}} \mathcal{D}]$$

is defined by

$$\begin{array}{rcl}
g_S\Phi\perp & = & \perp \\
g_S\Phi\langle a, d\rangle & = & \langle Sa, \Phi d\rangle
\end{array}$$

Product:

$$(vii)\ \ \times^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D}^2 \to \mathcal{D}].\,(f\Phi)^\dagger$$

where

$$f : [\mathcal{D}^2 \to \mathcal{D}] \to [(\sum_{a \in \mathsf{Act}} \mathcal{D})^2 \to \sum_{a \in \mathsf{Act}} \mathcal{D}]$$

is defined by

$$\begin{aligned} f\Phi(x,\bot) = f\Phi(\bot,x) &= \bot \\ f\Phi(\langle a,d\rangle,\langle b,e\rangle) &= \langle ab,\Phi(d,e)\rangle \end{aligned}$$

The only point which needs to be checked to ensure that this definition yields well-defined continuous functions is that $g_A\Phi$, $g_S\Phi$ and $f\Phi$ are (bi)strict and continuous, which is immediate from the definitions. Note that restriction, relabelling and product are defined recursively, while sum and prefixing are interpreted by the basic operations derived from the domain equation for $\mathcal{D}$. This corresponds to the fact that restriction, relabelling and product can be *eliminated* (for finite terms) in the equational theory of SCCS modulo bisimulation.

The continuous Σ-algebra defined by 6.4 is denoted $\mathcal{D}_\Sigma$. The following is an easy consequence of 6.4 and 3.10.

Proposition 6.5 *The semantic function*

$$[\![\cdot]\!]^{\mathcal{D}} : T_\Sigma \longrightarrow \mathcal{D}_\Sigma$$

cuts down to surjections

$$T_\Sigma \twoheadrightarrow \mathcal{K}(\mathcal{D}), \quad T_{\Sigma'} \twoheadrightarrow \mathcal{K}(\mathcal{D}).$$

Thus the finite synchronisation trees provide a notation for the finite elements of $\mathcal{D}$.

We now relate our definitions of the SCCS operations on $\mathcal{D}$ to the transition system view of $\mathcal{D}$.

Proposition 6.6 *For all $d, d_1, d_2 \in \mathcal{K}(\mathcal{D})$:*

$$\begin{array}{rlcl} (i)(a) & \mathbb{O}^{\mathcal{D}}\downarrow & & (b)\ \mathbb{O}^{\mathcal{D}} \stackrel{a}{\not\rightarrow} \\ (ii)(a) & \Omega^{\mathcal{D}}\uparrow & & (b)\ \Omega^{\mathcal{D}} \stackrel{a}{\not\rightarrow} \\ (iii)(a) & a^{\mathcal{D}}d\downarrow & & \\ (b) & a^{\mathcal{D}}d_1 \stackrel{b}{\rightarrow} d_2 & \iff & b = a \;\&\; d_1 = d_2 \\ (iv)(a) & (d_1 +^{\mathcal{D}} d_2)\uparrow & \iff & d_1\uparrow \textit{ or } d_2\uparrow \\ (b) & d_1 +^{\mathcal{D}} d_2 \stackrel{a}{\rightarrow} d & \iff & d_1 \stackrel{a}{\rightarrow} d \textit{ or } d_2 \stackrel{a}{\rightarrow} d \end{array}$$

Restriction:

$$\begin{array}{rlcl} (v)(a) & (d\restriction^{\mathcal{D}} A)\uparrow & \iff & d\uparrow \\ (b) & d_1\restriction^{\mathcal{D}} A \stackrel{a}{\rightarrow} d_2 & \iff & \exists e_1, e_2.\, d_1 \stackrel{a}{\rightarrow} e_i,\ (i = 1,2) \\ & & & \&\ e_1\restriction^{\mathcal{D}} A \sqsubseteq d_2 \sqsubseteq e_2\restriction^{\mathcal{D}} A \\ & & & \&\ a \in A \end{array}$$

Relabelling:

$$\begin{array}{rlcl} (vi)(a) & (d[S]^{\mathcal{D}})\uparrow & \iff & d\uparrow \\ (b) & d_1[S]^{\mathcal{D}} \stackrel{a}{\rightarrow} d_2 & \iff & \exists e_1, e_2, b_1, b_2.\, d_1 \stackrel{a}{\rightarrow} e_i,\ (i = 1,2) \\ & & & \&\ e_1[S]^{\mathcal{D}} \sqsubseteq d_2 \sqsubseteq e_2[S]^{\mathcal{D}} \\ & & & \&\ Sb_1 = a = Sb_2 \end{array}$$

Product:

$$
\begin{array}{rlcl}
(vii)(a) & (d_1 \times^{\mathcal{D}} d_2)\uparrow & \iff & d_1\uparrow \textit{ or } d_2\uparrow \\
(b) & d_1 \times^{\mathcal{D}} d_2 \xrightarrow{a} d & \iff & \exists u_i, v_i, b_i, c_i\ (i = 1, 2). \\
 & & & d_1 \xrightarrow{b_i} u_i \,\&\, d_2 \xrightarrow{c_i} v_i\ (i = 1, 2) \\
 & & & \&\ (u_1 \times^{\mathcal{D}} v_1) \sqsubseteq d \sqsubseteq (u_2 \times^{\mathcal{D}} v_2) \\
 & & & \&\ b_i c_i = a\ (i = 1, 2).
\end{array}
$$

PROOF. We give two cases for illustration.
(v). We define

$$
\begin{array}{rcl}
\Theta & \equiv & \{\{\langle a, d' \restriction^{\mathcal{D}} A\rangle\} : \langle a, d'\rangle \in d, a \in A\} \\
 & & \cup \{\varnothing : d = \emptyset \text{ or } \exists \langle a, d'\rangle \in d.\, a \notin A\} \\
 & & \cup \{\{\bot\} : \bot \in d\}.
\end{array}
$$

Now

$$
\begin{array}{rcl}
d \restriction^{\mathcal{D}} A & = & Con(\bigcup \Theta^{\star}) \\
 & = & Con((\bigcup \Theta)^{\star}) \text{ by [Plo76] p. 477} \\
 & = & Con(\bigcup \Theta) \text{ since } d \in \mathcal{K}(\mathcal{D}) \\
 & = & Con(\{\langle a, d' \restriction^{\mathcal{D}} A\rangle : \langle a, d'\rangle \in d \,\&\, a \in A\} \\
 & & \cup \{\bot : \bot \in d\}),
\end{array}
$$

and (v) is readily derived from this description.
(vii). Similarly to (v),

$$
\begin{array}{rcl}
d_1 \times^{\mathcal{D}} d_2 & = & Con(\{\langle b_1 b_2, e_1 \times^{\mathcal{D}} e_2\rangle : \langle b_i, e_i\rangle \in d_i,\ i = 1, 2\} \\
 & & \cup \{\bot : \bot \in d_1 \text{ or } \bot \in d_2\}). \quad \blacksquare
\end{array}
$$

Proposition 6.7 *For all $t \in T_{\Sigma}$, $t \sim^{B} [\![t]\!]^{\mathcal{D}}$.*

PROOF. Firstly, we define a height function on T_{Σ} in the obvious way:

$$\mathsf{ht}(\sigma(t_1, \ldots, t_n)) = \sup \{\mathsf{ht}(t_i) : 1 \leq i \leq n\} + 1.$$

As an easy consequence of 6.3, we have:

$$t \xrightarrow{a} t' \implies \mathsf{ht}(t') < \mathsf{ht}(t).$$

The proposition is proved by induction on $\mathsf{ht}(t)$, and cases on the construction of t. The cases arising from operations in Σ' are immediate in the light of the parallelism between 6.3 and 6.6. We give one of the remaining cases for illustration.
$t \equiv t_1 \restriction^{\mathcal{D}} A$. Firstly,

$$
\begin{array}{rcll}
t\uparrow & \iff & t_1\uparrow & \text{by 6.3(v)} \\
 & \iff & [\![t_1]\!]^{\mathcal{D}}\uparrow & \text{by induction hypothesis} \\
 & \iff & ([\![t_1]\!]^{\mathcal{D}} \restriction^{\mathcal{D}} A)\uparrow & \text{by 6.6(v)} \\
 & \iff & [\![t_1 \restriction A)]\!]^{\mathcal{D}}\uparrow. &
\end{array}
$$

Next,

$$
\begin{array}{lll}
\bullet & t \xrightarrow{a} t' & \\
\Longrightarrow & t_1 \xrightarrow{a} t_1' \;\&\; t' = t_1' \upharpoonright A \;\&\; a \in A & \text{by 6.3(v)} \\
\Longrightarrow & \exists d'. [\![t_1]\!]^{\mathcal{D}} \xrightarrow{a} d' \;\&\; t_1' \precsim^B d' & \text{ind. hyp. on } t_1 \\
\Longrightarrow & t_1' \upharpoonright A \sim^B [\![t_1' \upharpoonright A]\!]^{\mathcal{D}} & \text{ind. hyp. on } t_1' \upharpoonright A \\
& = [\![t_1']\!]^{\mathcal{D}} \upharpoonright^{\mathcal{D}} A & \\
& \precsim^B d' \upharpoonright^{\mathcal{D}} A & \text{by 3.11} \\
& \text{(since } \upharpoonright^{\mathcal{D}} \text{ is monotone)} & \\
\Longrightarrow & \exists u. [\![t]\!]^{\mathcal{D}} \xrightarrow{a} u \;\&\; t' \precsim^B u & \text{by 6.6(v).}
\end{array}
$$

Similarly, we can show

$$t \xrightarrow{a} t' \;\Rightarrow\; \exists u. [\![t]\!]^{\mathcal{D}} \xrightarrow{a} u \;\&\; u \precsim^B t'.$$

Again,

$$
\begin{array}{lll}
\bullet & [\![t]\!]^{\mathcal{D}} \xrightarrow{a} d & \\
\Longrightarrow & \exists d_1, d_2. [\![t_1]\!]^{\mathcal{D}} \xrightarrow{a} d_i,\ i = 1,2 & \\
& \&\; d_1 \upharpoonright^{\mathcal{D}} A \sqsubseteq d \sqsubseteq d_2 \upharpoonright^{\mathcal{D}} A & \\
& \&\; a \in A & \text{by 6.6(v)} \\
\Longrightarrow & \exists t_1', t_2'. t_1 \xrightarrow{a} t_i',\ i = 1,2 & \\
& \&\; t_1' \precsim^B d_1,\ d_2 \precsim^B t_2' & \text{by induction hypothesis} \\
\Longrightarrow & t \xrightarrow{a} t_i' \upharpoonright A,\ i = 1,2 & \\
& \&\; t_1' \upharpoonright A \sim^B [\![t_1' \upharpoonright A]\!]^{\mathcal{D}} & \text{by induction hypothesis} \\
& = [\![t_1']\!]^{\mathcal{D}} \upharpoonright^{\mathcal{D}} A \precsim^B d_1 \upharpoonright^{\mathcal{D}} A \precsim^B d, &
\end{array}
$$

and similarly $d \precsim^B t_2' \upharpoonright A$. Altogether, we have $t \sim^B [\![t]\!]^{\mathcal{D}}$. ∎

As an immediate consequence of this Proposition and 3.11 we have

Theorem 6.8 (Full Abstraction for Finite Terms) *For all $t_1, t_2 \in T_\Sigma$:*

$$t_1 \precsim^B t_2 \iff [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}.$$

As further consequences of 6.8 we have

- $[\![\cdot]\!]^{\mathcal{D}}$ agrees with the syntax-free map $[\![\cdot]\!]$ defined in Section 5. Indeed, $t \sim^B [\![t]\!]^{\mathcal{D}}$ implies $\mathcal{L}_\omega([\![t]\!]^{\mathcal{D}}) = \mathcal{L}_\omega(t) = \mathcal{L}_\omega([\![t]\!])$, which implies $[\![t]\!]^{\mathcal{D}} = [\![t]\!]$.
- T_Σ is a finitary transition system, by 5.22.

Moreover, we can derive two further characterisations of $\mathcal{D}$.

Theorem 6.9 *(i) $\mathcal{K}(\mathcal{D}) \cong (T_{\Sigma'}/{\sim^B}, \precsim^B/{\sim^B})$, and therefore*
(ii) $D \cong \mathsf{Idl}\,(T_{\Sigma'}/{\sim^B}, \precsim^B/{\sim^B})$.

PROOF. Immediate from 6.5 and 6.8. ∎

We recall the notion of *continuous Σ-algebra* [GTW78, Gue81]. This is just a Σ-algebra whose carrier is a cpo, and whose operations are continuous. A homomorphism of such algebras which is continuous on the carriers is a *continuous Σ-homomorphism.* The category of these algebras and homomorphisms is denoted $\mathbf{CAlg}(\Sigma)$.

Definition 6.10 **SCCS-Alg** is the full subcategory of $\mathbf{CAlg}(\Sigma)$ of those algebras $\mathcal{A}$ satisfying

$$\forall t_1, t_2 \in T_\Sigma.\, t_1 \precsim^B t_2 \implies [\![t_1]\!]^{\mathcal{A}} \sqsubseteq [\![t_2]\!]^{\mathcal{A}}.$$

Theorem 6.11 *$\mathcal{D}_\Sigma$ is initial in* **SCCS-Alg**.

PROOF. We begin by recalling a useful fact about continuous algebras ([Gue81] Proposition 3.12). Suppose $\mathcal{A}$ is a continuous algebra whose carrier A is an algebraic domain, such that the finite elements $\mathcal{K}(A)$ form a Σ-subalgebra. Then, given any monotonic Σ-homomorphism

$$f : \mathcal{K}(A) \longrightarrow \mathcal{B}$$

to a continuous Σ-algebra $\mathcal{B}$, there is a unique extension

$$\hat{f} : \mathcal{A} \longrightarrow \mathcal{B}$$

to a continuous Σ-homomorphism on $\mathcal{A}$.

By 6.5, $\mathcal{K}(\mathcal{D})$ is closed under the Σ-operations. Hence it suffices to construct a unique monotone Σ-homomorphism

$$f : \mathcal{K}(\mathcal{D}) \longrightarrow \mathcal{A}$$

to any $\mathcal{A}$ in **SCCS-Alg**. Given $d \in \mathcal{K}(\mathcal{D})$, by 6.5 there is $t \in T_\Sigma$ with $[\![t]\!]^{\mathcal{D}} = d$, and the only possible definition for f giving a Σ-homomorphism is

$$f : d \mapsto [\![t]\!]^{\mathcal{A}}.$$

This establishes uniqueness. For existence,

$$\begin{aligned} [\![t_1]\!]^{\mathcal{D}} = [\![t_2]\!]^{\mathcal{D}} &\iff [\![t_1]\!]^{\mathcal{D}} \sim^B [\![t_2]\!]^{\mathcal{D}} \quad \text{by 3.11} \\ &\iff t_1 \sim^B t_2 \qquad\qquad\quad \text{by 6.8} \\ &\implies [\![t_1]\!]^{\mathcal{A}} = [\![t_2]\!]^{\mathcal{A}} \end{aligned}$$

since $\mathcal{A}$ is in **SCCS-Alg**, and so f is well-defined. Similarly,

$$[\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}} \Rightarrow t_1 \precsim^B t_2 \Rightarrow [\![t_1]\!]^{\mathcal{A}} \sqsubseteq [\![t_2]\!]^{\mathcal{A}},$$

and so f is monotone. ∎

The purely algebraic part of SCCS which we have developed so far only allows the description of *finite* processes. We now extend the calculus with recursion.

Definition 6.12 We fix a set of variables Var, ranged over by x, y, z. The syntax of *recursive terms* REC_Σ, is then defined by

$$t \ ::= \ \sigma(t_1, \ldots, t_n) \ \ (\sigma \in \Sigma_n) \mid x \mid \mathsf{rec}\, x.t$$

In an obvious way, we can take T_Σ as a subset of REC_Σ. Note that $\mathsf{rec}\, x.t$ is a variable-binding construct. The set of *closed recursive terms* is denoted CREC_Σ.

We now extend the definition of the operational semantics to CREC_Σ:

$$(D\mathsf{rec})\ \frac{t[\Omega/x]\uparrow}{\mathsf{rec}\, x.t\uparrow} \qquad (T\mathsf{rec})\ \frac{t[\mathsf{rec}\, x.t/x] \xrightarrow{a} t'}{\mathsf{rec}\, x.t \xrightarrow{a} t'}$$

We thus obtain a transition system $(\mathrm{CREC}_\Sigma, \mathsf{Act}, \rightarrow, \uparrow)$. It is not too hard to see that this system is weakly finite-branching, and therefore by 5.23 satisfies (BN). However, most of the other finiteness conditions on transition systems fail, as the following examples show.

Examples

(1) **Failure of sort-finiteness.** Assume Act is infinite, in particular that $\{a_n\}$ is a sequence of distinct actions, and that S is a relabelling such that

$$Sa_n = a_{n+1} \ \ (n \in \omega).$$

Then

$$\mathsf{rec}\, x.\, a_0\mathbb{O} + x[S]$$

has the behaviour described by the synchronisation tree

$$\sum_{n\in\omega} a_n\mathbb{O} + \Omega.$$

(2) **Failure of (FA), and $\sqsubseteq_\omega \neq \sqsubseteq^B$.** By the example following 5.23, it suffices to show that the synchronisation tree

$$p \equiv \sum_{n\in\omega} a^n\mathbb{O} + \Omega$$

can be defined in SCCS to disprove (FA); while the same example shows that $\sqsubseteq_\omega \neq \sqsubseteq^B$, since

$$p \sim_\omega p + a^\omega, \ \ p \not\sim_{\omega+1} p + a^\omega,$$

and we can define $a^\omega \equiv \mathsf{rec}\, x.\, ax$. But using *unguarded* recursion (cf. [Mil83]), we can define

$$p \equiv (\mathsf{rec}\, x.\, (\Delta a + (\Delta a \times x))) \restriction \{a\}$$

where $\Delta a \equiv \mathsf{rec}\, y.\, a1^\omega + 1y$.

(3) $\sqsubseteq^F \neq \sqsubseteq_\omega$. Again, following the examples after 5.10, it suffices to show that the synchronisation trees

$$\begin{aligned} p &\equiv a(\sum_{n\in\mathbb{N}} b_n\mathbb{O}) + \Omega \\ q &\equiv \sum_{n\in\mathbb{N}} a(\sum_{m\in\mathbb{N}\setminus\{n\}} b_m\mathbb{O} + \Omega) + \Omega \end{aligned}$$

are definable in SCCS. Clearly p is definable in the same way as Example (1). For q, we need some additional assumptions on Act:

- There are $c, \{c_n\} \in \mathsf{Act}$ such that, for $k, m \in \mathbb{N}$:

$$\begin{aligned} c^{(k)} c_m &= b_m \ \ (k \neq m) \\ c^{(m)} c_m &= b_{m+1} \end{aligned}$$

  where $c^{(k)} \equiv \underbrace{c \cdots c}_{k}$, i.e. the product in the monoid Act.

- There is a relabelling S such that

$$S c_n = c_{n+1} \ \ (n \in \mathbb{N}).$$

(To see that these requirements can be met, let Act be the free abelian monoid over the generators $0, a, b_k, c, c_k$ $(k \in \mathbb{N})$ subject to the relations

$$0x = x0 = 0, \quad c^{(k)} c_m = b_m \ \ (k \neq m), \quad c^{(m)} c_m = b_{m+1}$$

for $k, m \in \mathbb{N}$. Let S be the endomorphism induced by

$$S0 = Sa = Sb_k = Sc = 0, \quad Sc_k = c_{k+1},$$

which is well-defined since S preserves the relations.)

Then we can define

$$\begin{aligned} q &\equiv \mathsf{rec}\, x.\, ar + (1c\mathbb{O} \times x) \\ r &\equiv \mathsf{rec}\, y.\, c_1 \mathbb{O} + x[S], \end{aligned}$$

and calculate:

$$\begin{aligned} r &= \sum_{n \in \mathbb{N}} c_n \mathbb{O} + \Omega, \\ q &= \sum_{n \in \mathbb{N}} (\prod_{i=1}^{n} 1c\mathbb{O} \times ar) + \Omega \\ &= \sum_{n \in \mathbb{N}} a(c^{(n)} \mathbb{O} \times \sum_{m \in \mathbb{N}} c_m \mathbb{O} + \Omega) + \Omega \\ &= \sum_{n \in \mathbb{N}} a(\sum_{m \in \mathbb{N}} (c^{(n)} c_m) \mathbb{O} + \Omega) + \Omega \\ &= \sum_{n \in \mathbb{N}} a(\sum_{m \in \mathbb{N} \setminus \{n\}} b_m \mathbb{O} + \Omega) + \Omega \end{aligned}$$

as required.

By contrast with Example (3), Hennessy claims in [Hen81] Theorem 4.1 that $\precsim^F = \precsim_\omega$ for SCCS. The defect in his argument occurs in the definition of $p^{(n)}$ at the start of section 4 of [Hen81]; there appears to be an implicit assumption that SCCS is sort-finite. Indeed, as an easy consequence of our work in the previous Section, we have

Proposition 6.13 *In any sort-finite transition system satisfying (BN):*

$$\precsim^F = \precsim_\omega .$$

PROOF. Let $p, q \in \mathrm{Proc}$ in such a system.

$$\begin{array}{rcll} p \underset{\sim}{\sqsubset}^F q & \Longrightarrow & \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q) & \\ & \Longrightarrow & \mathcal{L}_{\bigvee\infty}(p) \subseteq \mathcal{L}_{\bigvee\infty}(q) & \text{(BN)} \\ & \Longrightarrow & \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q) & \\ & \Longrightarrow & p \underset{\sim}{\sqsubset}_\omega q & \text{sort-finiteness.} \end{array}$$ ∎

Nevertheless, Hennessy's results on full abstraction are valid when $\underset{\sim}{\sqsubset}_\omega$ is replaced by $\underset{\sim}{\sqsubset}^F$, and we shall make use of them shortly.

Firstly, we need to extend our denotational semantics $[\![\cdot]\!]^{\mathcal{D}}$ to recursive terms. This is done in the standard way; we introduce environments to deal with variables, and interpret recursion by least fixed points.

Definition 6.14 Denotational semantics of recursive terms:

$$[\![\cdot]\!]^{\mathcal{D}} : \mathrm{REC}_\Sigma \longrightarrow \mathsf{Env} \longrightarrow \mathcal{D} \quad (\mathsf{Env} \equiv \mathcal{D}^{\mathsf{Var}})$$

$$\begin{array}{lcl} [\![x]\!]^{\mathcal{D}}\rho & = & \rho x \\ [\![\sigma(t_1, \ldots, t_n)]\!]^{\mathcal{D}}\rho & = & \sigma^{\mathcal{D}}([\![t_1]\!]^{\mathcal{D}}\rho, \ldots, [\![t_n]\!]^{\mathcal{D}}\rho) \\ [\![\mathsf{rec}\, x.\, t]\!]^{\mathcal{D}}\rho & = & \mu d \in \mathcal{D}.\, [\![t]\!]^{\mathcal{D}}\rho[x \mapsto d]. \end{array}$$

We now want to extend our Full Abstraction Theorem to recursive terms. We can use Hennessy's results in [Hen81] to get a cheap proof. In that paper, Hennessy constructs a term model $\mathcal{I}$ with the following properties:

1. $\mathcal{I}$ is an algebraic continuous Σ-algebra all finite elements of which are definable in T_Σ.
2. $\mathcal{I}$ is fully abstract for recursive terms with respect to the finitary preorder; for all $t_1, t_2 \in \mathrm{CREC}_\Sigma$:
$$t_1 \underset{\sim}{\sqsubset}^F t_2 \iff [\![t_1]\!]^{\mathcal{I}} \sqsubseteq [\![t_2]\!]^{\mathcal{I}}.$$

Combining (1) and (2) with Theorem 6.11, we obtain

Theorem 6.15 *$\mathcal{D}_\Sigma$ and $\mathcal{I}$ are isomorphic as continuous Σ-algebras.*

Let $h : \mathcal{D}_\Sigma \to \mathcal{I}$ be the isomorphism given by Theorem 6.15. It is immediate that h preserves denotations of terms in T_Σ:

$$\forall t \in T_\Sigma.\, h([\![t]\!]^{\mathcal{D}}) = [\![t]\!]^{\mathcal{I}}.$$

To extend this to recursive terms we need one further piece of machinery.

Definition 6.16 Let $\simeq$ be the least Σ-congruence over REC_Σ generated by

$$\mathsf{rec}\, x.\, t \simeq t[\mathsf{rec}\, x.\, t/x].$$

Let t_Ω be the term obtained from t by replacing each subexpression of the form $\mathsf{rec}\, x.\, t'$ by Ω. The *syntactic approximants* of t are defined by:

$$SA(t) \equiv \{t'_\Omega : t' \simeq t\}.$$

Note that $SA(t) \subseteq T_\Sigma$ for all $t \in \mathrm{CREC}_\Sigma$.

Now the following is standard (cf. e.g. [GTWW77]):

Lemma 6.17 (Syntactic Approximation) *For all $t \in \mathrm{CREC}_\Sigma$:*

$$[\![t]\!]^{\mathcal{D}} = \bigsqcup\{[\![t']\!]^{\mathcal{D}} : t' \in SA(t)\}.$$

Hennessy proves the corresponding result for $[\![\cdot]\!]^{\mathcal{I}}$ as his Lemma 3.4.

Proposition 6.18 *For all $t \in \mathrm{CREC}_\Sigma$:*

$$h([\![t]\!]^{\mathcal{D}}) = [\![t]\!]^{\mathcal{I}}.$$

PROOF.

$$\begin{aligned} h([\![t]\!]^{\mathcal{D}}) &= h(\bigsqcup\{[\![t']\!]^{\mathcal{D}} : t' \in SA(t)\}) && \text{by 6.17} \\ &= \bigsqcup\{h([\![t']\!]^{\mathcal{D}}) : t' \in SA(t)\} && h \text{ is continuous} \\ &= \bigsqcup\{[\![t']\!]^{\mathcal{I}} : t' \in SA(t)\} && \text{by 6.15} \\ &= [\![t]\!]^{\mathcal{I}}. && \blacksquare \end{aligned}$$

Theorem 6.19 (Full Abstraction for Recursive Terms) *For all $t_1, t_2 \in \mathrm{CREC}_\Sigma$:*

$$t_1 \underset{\sim}{\sqsubset}^F t_2 \iff [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}.$$

PROOF.

$$\begin{aligned} t_1 \underset{\sim}{\sqsubset}^F t_2 &\iff [\![t_1]\!]^{\mathcal{I}} \sqsubseteq [\![t_2]\!]^{\mathcal{I}} \\ &\iff [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}, \end{aligned}$$

by 6.18 and since h is an order-isomorphism. $\blacksquare$

Since $\mathcal{D}$ is algebraic, this result extends to terms with variables in the obvious way. It follows that the axiomatisation of the order and equality relations between terms of SCCS presented in [Hen81] is sound and complete for $\mathcal{D}_\Sigma$.

7 Further Directions

A number of interesting further developments suggest themselves. Firstly, from the results of this paper, we can define a fully abstract denotational semantics for SCCS in our denotational metalanguage, and faithfully interpret Hennessy-Milner logic into our domain logic. Thus we should *automatically* get a compositional proof theory for HML. It would be particularly worthwhile to demonstrate this in detail, as the construction of compositional proof systems for HML by Stirling [Sti87] and Winskel [Win85] is one of the most impressive examples to date of the exercise of *ad hoc* ingenuity in the design of program logics.

Another useful extension of our work would be to equivalences other than bisimulation. The author has recently obtained a fully abstract model for *observation equivalence*; this provides further encouraging evidence for the applicability of denotational methods in concurrency.

The scope of our work in this paper is limited to process calculi which admit only *bounded non-determinism*. Indeed, it is well-known that unbounded non-determinism is incompatible with the standard domain-theoretic framework for denotational semantics (see e.g. [AP86]). It would be interesting to see if our results could be lifted to calculi with unbounded non-determinism, by substituting Plotkin's powerdomain for countable non-determinism ([Plo82]) for the standard Plotkin powerdomain in our domain equation for synchronization trees. An interesting point about this construction is that we lack a good representation for it, and a logical description might help. It would also be interesting to consider probabilistic semantics for e.g. SCCS, by using the probabilistic powerdomain of [JP89]; particularly in the light of recent work on probabilistic testing and bisimulation [LS89]. Finally, it would be interesting to look at the definability of the computable elements in our domain; the results of de Simone [dS85] on the expressive completeness of Meije and SCCS should be relevant here.

Some more general points concerning the two case studies given in the present paper and [Abr88b]. Firstly, the operational models we study—labelled transition systems in the present paper and lambda transition systems in [Abr88b]—are *almost* derived in a systematic way from our domain equations. Namely, a labelled transition system is a map

$$\mathrm{Proc} \longrightarrow \wp((\mathsf{Act} \times \mathrm{Proc}) \cup \{\bot\})$$

i.e. a coalgebra of the functor (on **Set**)

$$X \mapsto \wp((\mathsf{Act} \times X) \cup \{\bot\}).$$

Similarly, an applicative transition system is a coalgebra of the **Set**-functor

$$X \mapsto (X \to X) \cup \{\bot\}.$$

Since $\mathsf{Act} \times \mathcal{D} \cup \{\bot\}$ can be put in natural bijection with $\sum_{a \in \mathsf{Act}} \mathcal{D}$, and $(\mathcal{D} \to \mathcal{D}) \cup \{\bot\}$ with $(\mathcal{D} \to \mathcal{D})_\bot$, we see that our domain equations give rise to essentially the *same* functors, but over domains rather than sets. Moreover, because of the limit-colimit coincidence in Domain theory [SP82], we can take the *initial solution* of a domain equation (with respect to embeddings) as the *final coalgebra* (with respect to projections). Thus our results can in some sense be seen as concerning the interpretation and "best approximation" of **Set**-based structures in topological ones. Clearly some general theory is called for here.

Finally, one of our aims in this paper and [Abr88b] was to place the study of functional languages and concurrency on as similar a footing as possible. Much remains to be done here, although we hope to have made a useful first step.

Acknowledgements

I would like to thank Axel Poigné, Iain Phillips and Glynn Winskel for very useful discussions on the material of this paper, which is a revised version of Chapter 5 of [Abr87a]. The first draft was written during a visit to the University of Nijmegen in March–April 1986. I would like to thank my host Professor Raymond Boute, and also Erik Meijer and Henk Barendregt for a very enjoyable and productive visit. My thanks to Steve Vickers for an improvement to the axiomatization of the domain logic; and to the Journal referee for a number of useful comments and corrections. Finally, my thanks to the U.K. Science and Engineering Research Council for financial support.

References

[AB84] D. Austry and G. Boudol. Algèbres de processus et synchronisations. *Theoretical Computer Science*, 30:91–131, 1984.

[Abr83a] S. Abramsky. Experiments, powerdomains and fully abstract models for applicative multiprogramming. In M. Karpinski, editor, *Foundations of Computation Theory*, pages 1–13, Berlin, 1983. Springer-Verlag. Lecture Notes in Computer Science Vol. 158.

[Abr83b] S. Abramsky. Semantic foundations for applicative multiprogramming. In J. Diaz, editor, *Automata, Languages and programming*, pages 1–14, Berlin, 1983. Springer-verlag. Lecture Notes in Computer Science Vol. 154.

[Abr87a] S. Abramsky. *Domain Theory and the Logic of Observable Properties*. PhD thesis, University of London, October 1987.

[Abr87b] S. Abramsky. Domain theory in logical form. In *Symposium on Logic In Computer Science*, pages 47–53. Computer Society Press of the IEEE, 1987.

[Abr87c] S. Abramsky. Observation equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.

[Abr88a] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 1988. To appear.

[Abr88b] S. Abramsky. The lazy λ-calculus. In D. Turner, editor, *Declarative Programming*. Addison Wesley, 1988. To appear. Also Chapter 6 of [Abr87a].

[AP86] K. Apt and G. Plotkin. Countable nondeterminism and random assignment. *Journal of the Association for Computing Machinery*, 33(4):724–767, 1986.

[Bar75] K. J. Barwise. Back and forth through infinitary logic. In M. Morley, editor, *Studies in Model Theory*. Mathematical Association of America, 1975.

[BK84] J. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[dS85] R. de Simone. Higher level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

[Gor79] M. J. C. Gordon. *The Denotational Description of Programming Languages*. Springer-Verlag, Berlin, 1979.

[GTW78] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. T. Yeh, editor, *Current Trends in Programming Methodology IV: Data Structuring*, pages 80–144. Prentice Hall, 1978.

[GTWW77] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24:68–95, 1977.

[Gue81] I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1981.

[Hen81] M. C. B. Hennessy. A term model for synchronous processes. *Information and Control*, 51(1):58–75, 1981.

[HM85] M. C. B. Hennessy and Robin Milner. Algebraic laws for non-determinism and concurrency. *JACM*, 32:137–161, 85.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985.

[HP79] M. C. B. Hennessy and G. D. Plotkin. Full abstraction for a simple parallel programming language. In J. Beçvar, editor, *Mathematical Foundations of Computer Science*, Berlin, 1979. Springer-Verlag. Lecture Notes in Computer Science Vol. 74.

[Joh82] P. T. Johnstone. *Stone Spaces*, volume 3 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1982.

[JP89] C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the 4th Annual Symposium on Logic in Computer Science*. Computer Society Press of the IEEE, 1989.

[LS89] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Proceedings of the 15th Annual Symposium on the Principles of Programming Languages*. Computer Society Press of the IEEE, 1989.

[Mil75] R. Milner. Processes, a mathematical model of computing agents. In *Logic Colloquium, Bristol 1973*, pages 157–174. North Holland, Amsterdam, 1975.

[Mil77] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[Mil80] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1980.

[Mil81] R. Milner. A modal characterisation of observable machine behaviours. In G. Astesiano and C. Böhm, editors, *CAAP '81*, pages 25–34, Berlin, 1981. Springer-Verlag. Lecture Notes in Computer Science Vol. 112.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil85] R. Milner. Lectures on a calculus for communicating systems. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, pages 197–221, Berlin, 1985. Springer-Verlag. Lecture Notes in Computer Science Vol. 197.

[Mil86] R. Milner. Process constructors and interpretations. In H.-J. Kugler, editor, *Information Processing 86*, pages 507–514. Elsevier Science Publishers B. V. (North Holland), 1986.

[MM79] G. Milne and R. Milner. Concurrent processes and their syntax. *Journal of the ACM*, 26(2):302–321, 1979.

[Mos74] Y. Moschovakis. *Elementary Induction on Abstract Structures.* North Holland, 1974.

[MS76] R. E. Milne and C. Strachey. *A Theory of Programming Language Semantics.* Chapman and Hall, London, 1976.

[Niv81] M. Nivat. Infinite words, infinite trees, infinite computations. In J. W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science III part 2*, volume 109 of *Mathematical Centre Tracts*, pages 3–52. Centrum voor Wiskunde en Informatica, Amsterdam, 1981.

[Par81] D. M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, Berlin, 1981. Springer-Verlag. Lecture Notes in Computer Science Vol. 104.

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

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Plo81] G. D. Plotkin. Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa Notes"). Dept. of Computer Science, Univ. of Edinburgh, 1981.

[Plo82] G. D. Plotkin. A powerdomain for countable non-determinism. In M. Nielsen and E. M. Schmidt, editors, *Automata, Languages and programming*, pages 412–428, Berlin, 1982. EATCS, Springer-Verlag. Lecture Notes in Computer Science Vol. 140.

[Rou85] W. C. Rounds. Applications of topology to semantics of communicating processes. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, pages 360–372, Berlin, 1985. Springer-Verlag. Lecture Notes in Computer Science Vol. 197.

[Smy83] M. B. Smyth. Powerdomains and predicate transformers: a topological view. In J. Diaz, editor, *Automata, Languages and Programming*, pages 662–675, Berlin, 1983. Springer-Verlag. Lecture Notes in Computer Science Vol. 154.

[SP82] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Computing*, 11:761–783, 1982.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

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

[WBT85] E. G. Wagner, S. L. Bloom, and J. W. Thatcher. Why algebraic theories? In M. Nivat and J. C. Reynolds, editors, *Algebraic Semantics*, pages 607–634. Cambridge University press, 1985.

[Win83] G. Winskel. Powerdomains and modality. In M. Karpinski, editor, *Foundations of Computation Theory*, pages 505–514, Berlin, 1983. Springer-Verlag. Lecture Notes in Computer Science Vol. 158.

[Win85] G. Winskel. A complete proof system for SCCS with modal assertions. In S. N. Maheshwari, editor, *Foundations of Software technology and Theoretical Computer Science*, pages 392–410, Berlin, 1985. Springer-Verlag. Lecture Notes in Computer Science Vol. 206.