Pi-Calculus in Logical Form

M.M. Bonsangue[1⋆] and A. Kurz[2⋆⋆]

[1] LIACS - Leiden University, The Netherlands
[2] Department of Computer Science, University of Leicester, United Kingdom

Abstract. Abramsky's logical formulation of domain theory is extended to encompass the domain theoretic model for pi-calculus processes of Stark and of Fiore, Moggi and Sangiorgi. This is done by defining a logical counterpart of categorical constructions including dynamic name allocation and name exponentiation, and showing that they are dual to standard constructs in functor categories. We show that initial algebras of functors defined in terms of these constructs give rise to a logic that is sound, complete, and characterises bisimilarity. The approach is modular, and we apply it to derive a logical formulation of pi-calculus. The resulting logic is a modal calculus with primitives for input, free output and bound output.

1 Introduction

The π-calculus [10, 14] is a process algebra for expressing processes that interact by exchanging channel names via shared channels. Fiore, Moggi and Sangiorgi [6] and Stark [15] show that π-calculus processes can be considered as the elements of the final coalgebra for a functor T on a suitable category $\mathcal{X}$. The category in question allows processes with local names to depend on the *finite* set of names which can be used for communication, while the functor T is a variation of the one used in [1] to model CCS.

In this paper we build on Abramsky's domain theory in logical form [2], to obtain a logic for π-calculus by considering the Stone-dual category $\mathcal{A}$ of $\mathcal{X}$ and the dual functor L of T

More generally, in any such situation, the initial L-algebra describes a logic for T-processes that characterizes T-bisimilarity. Moreover, by presenting the functor L by operations and equations [3], we obtain a complete calculus for the logic.

Our main result consists in the formulation of two sound and complete proof systems: one for assertions on the universal domain where the π-calculus is interpreted, and another for assigning processes to formulas. The first proof system can be used to reason about syntax-free transition systems (i.e. coalgebras) in the style of more traditional modal logics while the second one is tailored for reasoning about syntactically given π-calculus terms, including recursive ones. In both logics, process specifications

⋆ The research of Dr. Bonsangue has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences
⋆⋆ Partially supported by NWO/British Council and EPSRC EP/C014014/1.

are given in an extension of standard modal logic, extended by primitives for name input and output, taking into account in a uniform way the possibility of communication of a fresh name.

We now give an outline of the reminder of the paper. In Section 2 we recall the language of the π-calculus. Various coalgebraic and algebraic concepts needed to derive the logic are reviewed in Section 3, whereas the general framework for deriving logics for coalgebras is discussed in Section 4. In order to derive a logic for the π-calculus we give in Section 5 a presentation by operations and equations of each functor involved in the denotational model Pi of π-calculus. This way we immediately obtain a logic for the class of all Pi-coalgebras (Section 6), which is sound, complete, and characterises strong late bisimilarity. In Section 7, we use the fully abstract semantics of π-calculus process in terms of the final Pi-coalgebra, and give a logic that can be used to reason in a compositional manner about syntactically given π-calculus terms. The paper ends with a brief comparison with other logics for the π-calculus and outline of future research.

Finally, we would like to acknowledge our great debts to Davide Sangiorgi who provided valuable suggestions throughout the writing of the paper. We are grateful to Marcelo Fiore for suggesting the topic and to Roy Crole and Emilio Tuosto for discussions and valuable comments on previous drafts.

2 The π-Calculus

We let a, b range over names; x over process variables; and P, Q over processes. The syntax of processes is as follows:

$$\alpha ::= a(b) \mid \bar{a}\,b \mid \tau$$
$$P ::= 0 \mid x \mid \alpha.P \mid [a = b]P \mid [a \neq b]P \mid P + P \mid P|P \mid (\nu a)P \mid \mu x.P$$

The process constructs available in the calculus are the inaction process, the process variable, action prefixing, matching, mismatching, choice, parallel composition, restriction and recursion. Actions can be inputs, output, and silent move. Bound output $\bar{a}(b)$ is an extra action needed to describe the output of a private name b along a, and extending the scope of b to the receiver.

Names can be bound by the restriction and input prefix constructs, process variables by the recursion construct. *Free names* (fn) and *free variables* (fv) of a process are defined as expected.

We use a slightly different syntax than that of the original π-calculus as introduced in [10]. Indeed, we allow processes to contain process variables, and hence to be *open*. A closed process is a process not containing free process variables, but it may contain free names. We use process variables for recursion. A standard alternative to recursion is the use of replication. Replication $!P$ can be encoded as the process $\mu x.(P|x)$.

We refer to [14] for a detailed description of the above constructs, and in particular for their associated labelled transition rules and derived notion $\sim$ of strong late bisimilarity between π-calculus processes. Strong late bisimilarity is not a congruence relation, because it need not be preserved by input prefixing; the induced congruence $\sim^c$ is called *strong late congruence*.

3 Coalgebras, Algebras, Stone Duality

Given a locally small category $\mathbf{C}$, its class of objects is denoted by $|\mathbf{C}|$ and the set of arrows from A to B by $\mathbf{C}(A, B)$. The category of sets and functions is denoted by $\mathbf{Set}$. Since π-calculus processes are defined over a finite set of free names available for interactions, we are particularly interested in functor categories $\mathbf{C}^{\mathbf{I}}$, where $\mathbf{I}$ is the category of finite sets i with injective maps $\imath{:}j \to i$. A functor X in $\mathbf{C}^{\mathbf{I}}$ associates to each finite set (of names) an object in $\mathbf{C}$. Further, the action of X on an injection in $\mathbf{I}$ can be thought of as a relabelling operator [6, 15]. We will make use of the fact that, up to equivalence, $(\mathbf{I}, 0, +)$ is the strict monoidal category with initial unit 0 freely generated from one object 1.

Coalgebras Coalgebras are simple mathematical structures for describing dynamical systems like automata and transition systems [13]. Given an endofunctor T on a category $\mathbf{C}$, a *T-coalgebra* is an arrow $\xi{:}X \to TX$ in $\mathbf{C}$. A morphism $f{:}\xi \to \xi'$ between coalgebras is an arrow $f{:}X \to X'$ such that $Tf \circ \xi = \xi' \circ f$. In this paper, the category $\mathbf{C}$ is always a concrete category (like the categories $\mathbf{Spec}$ of spectral topological spaces [9], or $\mathbf{SFP}$ of SFP domains [12]). It makes therefore sense to speak of the elements $x \in X$, or *states*, of an object X of $\mathbf{C}$. We say that two states x, x' of $\xi{:}X \to TX$ and $\xi'{:}X' \to TX'$ are *bisimilar* if there are coalgebra morphisms f, f' with $f(x) = f'(x')$. For example, for the functor $\mathcal{P}_{\mathbf{Set}}(A \times -)$ on $\mathbf{Set}$, where $\mathcal{P}_{\mathbf{Set}}$ is the powerset, coalgebras are transition systems with labels in A and bisimilarity coincides with the standard notion from process algebra [13].

Based on [6, 15], we use coalgebras for modelling π-calculus processes. In particular we consider the following functor Pi on a suitable category $\mathbf{C}^{\mathbf{I}}$

$$\mathcal{P}(\ -\ \ +\ \ N \times (N \to -)\ \ +\ \ N \times (N \times -)\ \ +\ \ N \times \delta -\ ). \tag{1}$$

Intuitively, a coalgebra $\xi{:}X \to Pi(X)$ for this functor is a transition system with $\xi_i(x)$ describing the one-step transition of a process $x \in X(i)$ with free names in i, the components of the coproduct corresponding to silent transition, input, free output, bound output, or termination. Here N is the object of names, $\mathcal{P}$ is a powerdomain for modelling non-determinism, $+$ is the coproduct for selecting different actions, $N \to X$ is an exponential for the input and δX is for modelling dynamic allocation of names.

Algebras In this paper, we think of coalgebras as dynamic systems and of algebras as logics. It is our aim to relate the coalgebraic semantics of the π-calculus [6, 15] with a suitable algebraic semantics.

Given an endofunctor L on a category $\mathbf{A}$, an L-algebra consists of an arrow $\alpha{:}LA \to A$. A morphism $f{:}\alpha \to \alpha'$ between L-algebras is an arrow $f{:}A \to A'$ such that $f \circ \alpha = \alpha' \circ Lf$.

From the point of view of universal algebra, a (Σ, E)-algebra over an S-sorted signature Σ and equations E consists of carrier sets A_s for each sort $s \in S$ together with a collection of operations on the carrier sets respecting the equations in E. The category of S-sorted (Σ, E)-algebras is defined as usual and denoted by $Alg(\Sigma, E)$. We say that a category $\mathbf{A}$, equipped with a forgetful functor $U{:}\mathbf{A} \to \mathbf{Set}^S$ (here S is seen as a discrete category), has a *presentation* if there exists a signature Σ and

equations E such that $\mathbf{A}$ is concretely[3] isomorphic to $Alg(\Sigma, E)$. If, in addition, $\mathbf{A}$ has free algebras then U is *monadic*. For example, the category $\mathbf{DL}$ of distributive lattices is monadic [9].

Each S-sorted signature Σ determines a functor G_Σ:$\mathbf{Set}^S \to \mathbf{Set}^S$, with

$$G_\Sigma(A)(s) = \coprod_{\langle s_1 \ldots s_n \rangle \in S^*} G_{\langle s_1 \ldots s_n \rangle} \times A(s_1) \times \ldots A(s_n) \tag{2}$$

where $G_{\langle s_1 \ldots s_n \rangle}$ is the set of operations of sort $s_1, \ldots s_n \to s$. We have $Alg(\Sigma, \emptyset) \cong Alg(G_\Sigma)$.

Presheaves as many-sorted algebras Given a one-sorted presentation $\mathbf{A} \cong Alg(\Sigma_\mathbf{A}, E_\mathbf{A})$ and a small category $\mathcal{K}$, the presentation $\mathbf{A}^\mathcal{K} \cong Alg(\Sigma_{\mathbf{A}^\mathcal{K}}, E_{\mathbf{A}^\mathcal{K}})$ as a category of $|\mathcal{K}|$-sorted algebras is given as follows.

1. $\Sigma_{\mathbf{A}^\mathcal{K}}$ has an operation op of sort $k, \ldots k \to k$ for each operation op in $\Sigma_\mathbf{A}$ and $k \in |\mathcal{K}|$, and an operation f:$h \to k$ for each arrow f in $\mathcal{K}$.
2. $E_{\mathbf{A}^\mathcal{K}}$ has equations $l = r$ for each equation $l = r$ in $E_\mathbf{A}$, $id(x) = x$ for each identity arrow id:$k \to k$ in $\mathcal{K}$ and equations $h(x) = f(g(x))$ for arrows $h = f \circ g$ in $\mathcal{K}$.

As it will become clear below, our main interest is in the presheaf category $\mathbf{DL}^{\mathbf{I}^{op}}$.

Example 1. $\mathbf{DL}^{\mathbf{I}^{op}}$ is isomorphic to the category of many-sorted algebras given by the following equational theory:

- its sorts are the objects of $\mathbf{I}$. For each sort i we assume pairwise disjoint sets V_i of variables of sort i. We denote $v \in V_i$ by v:i.
- for each sort i there are constants $\top$:i, and $\bot$:i, and two binary operators $\wedge$:$i, i \to i$ and $\vee$:$i, i \to i$. They obey the equational laws of a distributive lattice.
- for each morphism $\imath$:$j \to i$ in $\mathbf{I}$ there is a unary operator $[\imath]$:$i \to j$. Its equational laws are
  $[\imath] \bigvee_{k \in K} v_k = \bigvee_{k \in K} [\imath] v_k$
  $[\imath] \bigwedge_{k \in K} v_k = \bigwedge_{k \in K} [\imath] v_k$
  $[id_i] v = v$,
  $[\imath \circ \jmath] v = [\jmath][\imath] v$ for $\imath$:$j \to i$, $\jmath$:$k \to j$.
  where K is a finite index set and v_k:i, v:i.

The equational laws of the above theory amount to saying that an algebra A is a functor from the opposite category of $\mathbf{I}$ to the category $\mathbf{DL}$ (of distributive lattices). From a logical point of view, elements of a distributive lattice $A(i)$ are (equivalence classes of) formulas, the induced order is the relation of logical entailment between formulas, and a morphism $A(\imath)$:$A(i) \to A(j)$ is a restriction operator on formulas.

The category $\mathbf{DL}^{\mathbf{I}^{op}}$ inherits much of the structure from $\mathbf{DL}$. For example, limits and colimits in $\mathbf{DL}^{\mathbf{I}^{op}}$ can be taken pointwise. Further, because the forgetful functor $\mathbf{DL} \to \mathbf{Set}$ is monadic, $\mathbf{DL}^{\mathbf{I}^{op}} \to \mathbf{Set}^{|\mathbf{I}|}$ is monadic as well.

[3] *Concretely* means that the isomorphism preserves the underlying carriers in $\mathbf{Set}^S$.

Stone duality SFP domains taken with the Scott topology are *spectral spaces* (coherent spaces in [9]). Spectral spaces are those compact sober topological spaces whose compact opens are a basis and closed under finite intersections. The category $\mathbf{Spec}$ of spectral spaces is dual to $\mathbf{DL}$ [9]. In detail, the spectral space $pt(D)$ of a distributive lattice D is the set of prime filters over D taken with the filter topology, while the distributive lattice $\mathcal{KO}(X)$ of a spectral space X is given by the set of compact opens of X ordered by subset inclusion.

$$\mathbf{Spec} \underset{pt}{\overset{\mathcal{KO}}{\rightleftarrows}} \mathbf{DL} \tag{3}$$

On morphisms, both $\mathcal{KO}$ and pt are given by inverse image. The following features of the duality are important:

- for $X \in \mathbf{Spec}$, $x \neq y \in X$ there is $o \in \mathcal{KO}(X)$ separating x and y,
- for $A \in \mathbf{DL}$, $a \not\leq b \in A$ there is $x \in pt(A)$ such that $a \in x$ and $b \notin x$.

The first property will be responsible for expressiveness and the second for completeness in Proposition 1.

The duality between $\mathbf{Spec}$ and $\mathbf{DL}$ restricts to a duality between the categories $\mathbf{SFP}_!$ and $\mathbf{DDL}$ where $\mathbf{SFP}_!$ is the category of SFP domains with strict functions[4] and $\mathbf{DDL}$ is its dual. An explicit description of $\mathbf{DDL}$ can be found in [2]. Furthermore, the duality between $\mathbf{SFP}_!$ and $\mathbf{DDL}$ lifts to a duality between the functor categories $\mathbf{SFP}_!^{\mathbf{I}}$ and $\mathbf{DDL}^{\mathbf{I}^{op}}$, see Diagram (4) below.

4 Logics for Coalgebras

This section gives a brief summary—tailored to the needs of the present paper—of Abramsky's framework of a Domain Theory in Logical Form [2] and extends [3] to the present situation. It is shown how an initial L-algebra gives rise to an adequate logic for T-coalgebras (Proposition 1) and how a presentation of L by operations and equations gives a complete proof calculus for this logic (Theorem 1). The situation we consider is

$$\begin{array}{ccc} Coalg(T) & \overset{\widetilde{\mathcal{KO}}}{\underset{\widetilde{pt}}{\rightleftarrows}} & Alg(L) \\ \downarrow & & \downarrow \\ \mathbf{SFP}_!^{\mathbf{I}} & \overset{\widehat{\mathcal{KO}}}{\underset{\widehat{pt}}{\rightleftarrows}} & \mathbf{DDL}^{\mathbf{I}^{op}} \end{array} \tag{4}$$

[4] The Abramsky powerdomain, which we use to interpret non-determinism, is functorial only for strict functions. An alternative would be to use the ordinary Plotkin powerdomain and require deadlock to be an observable action. This would not change the theory of bisimulation as deadlock processes are anyhow observably distinguishable from the rest of the processes.

with *dual functors* functors $T : \mathbf{SFP}^{\mathbf{I}}_! \to \mathbf{SFP}^{\mathbf{I}}_!$ and $L : \mathbf{DDL}^{\mathbf{I}^{op}} \to \mathbf{DDL}^{\mathbf{I}^{op}}$, ie there is an isomorphism

$$d : L\widehat{\mathcal{KO}} \to \widehat{\mathcal{KO}}\,T. \tag{5}$$

This isomorphism allows us to extend the equivalence between $\mathbf{SFP}^{\mathbf{I}}_!$ and $\mathbf{DDL}^{\mathbf{I}^{op}}$ to an equivalence between $Coalg(T)$ and $Alg(L)$. Therefore, in the same way as $\mathbf{DL}$ provides a logic for $\mathbf{Spec}$, we consider $Alg(L)$ as a logic for $Coalg(T)$:

Definition 1. *The algebra of formulas is the initial L-algebra.*[5] *Given a formula ϕ and a coalgebra (X, ξ), the semantics $[\![\phi]\!]_{(X,\xi)} \subseteq X$ is the image of ϕ under the unique morphism from the algebra of formulas to $\widetilde{\mathcal{KO}}(X, \xi)$.*

We write $Coalg(T) \models (\phi \leq \psi)$ if $[\![\phi]\!]_{(X,\xi)} \subseteq [\![\psi]\!]_{(X,\xi)}$ for all coalgebras.

Proposition 1. *The logic for T-coalgebras given in the previous definition*

1. *respects bisimilarity: formulas are invariant under bisimilarity*
2. *is expressive: any two non-bisimilar states are distinguished by some formula*
3. *is sound and complete: $Coalg(T) \models (\phi \leq \psi)$ iff in the initial algebra $\phi \leq \psi$ holds.*

The proposition is an immediate consequence of Stone duality and does not depend on the special situation considered here. What it doesn't give us yet is a proof calculus.

Presentations yield proof systems A presentation of $Alg(L)$ by operations and equations yields a many-sorted equational logic as a logical calculus for T-coalgebras. Our notion of a presentation by operations and equations is slightly non-standard but convenient as it makes the connection with the logics more straightforward (see [3] for a full discussion).

Definition 2. *Let F be the left adjoint of U:$\mathbf{DL}^{\mathbf{I}^{op}} \to \mathbf{Set}^{|\mathbf{I}|}$, and $\langle \Sigma, E \rangle$ consist of an $|\mathbf{I}|$-sorted signature Σ (inducing a functor G_Σ as in (4)) and of a set of equations $E = (E_{V(i)})_{V(i)\in\omega}$, with $E_V(i) \subseteq (UFG_\Sigma UFV(i))^2$. Here V is a functor (of sorted variables) in $\mathbf{Set}^{|\mathbf{I}|}$ with $V(i)$ ranging over finite cardinals for each sort i in $|\mathbf{I}|$. A functor L:$\mathbf{DL}^{\mathbf{I}^{op}} \to \mathbf{DL}^{\mathbf{I}^{op}}$ is presented by $\langle \Sigma, E \rangle$ if there exists a natural transformation $FG_\Sigma U \to L$ such that each component $FG_\Sigma UA \to LA$ is the joint coequaliser*

$$FE_V \rightrightarrows FG_\Sigma UFV \xrightarrow{FG_\Sigma Uv} FG_\Sigma UA \longrightarrow LA \tag{6}$$

where v ranges over natural transformations (valuations of variables) in $FV \to A$.

Remark 1. 1. The definition captures the idea that the presentation of LA is uniform in A. The format of the equations ensures that the Lindenbaum algebra of the equational logic $(\Sigma_{\mathbf{DL}} + \Sigma_L, E_{\mathbf{DL}} + E_L)$ is isomorphic to the initial L-algebra.

[5] We consider here only formulas without propositional variables. But everything extends smoothly to 'the algebra of formulas over variables X is the free L-algebra over X'.

2. $\mathbf{DL}^{\mathbf{I}^{op}}$ is monadic over $\mathbf{Set}^{|\mathbf{I}|}$ and each presentation presents indeed a functor. This will be used in the next section.
3. The notion extends to binary functors $L{:}(\mathbf{DL}^{\mathbf{I}^{op}})^2 \to \mathbf{DL}^{\mathbf{I}^{op}}$, which is useful to deal with (co)products.
4. To illustrate the format of the equations, consider the equation $\Box(v_0 \wedge v_1) =_i \Box v_0 \wedge \Box v_1$ from the presentation of the powerdomain in the next section. Note that, according to (6), the left-hand $\wedge$ is interpreted in $A(i)$ and the right-hand $\wedge$ in $LA(i)$.
5. Abramsky's logic [2] needs implications for the treatment of coalesced sum and function space. However, we will only use separated coproduct and a restricted form of function space where equations are enough.

As the proof of Proposition 1 only involves the final T-coalgebra and the initial L algebra, we can apply the presentations over $\mathbf{DL}^{\mathbf{I}^{op}}$ to coalgebras over $\mathbf{SFP}^{\mathbf{I}}_{!}$, if we can properly restrict L to the subcategory $\mathbf{DDL}^{\mathbf{I}^{op}}$.

Theorem 1. *Let $\langle \Sigma_L, E_L \rangle$ be a presentation of a functor L on $\mathbf{DL}^{\mathbf{I}^{op}}$ such that: (1) L restricts to the subcategory $\mathbf{DDL}^{\mathbf{I}^{op}}$ of $\mathbf{DL}^{\mathbf{I}^{op}}$, (2) the initial L-algebra lies in $\mathbf{DDL}^{\mathbf{I}^{op}}$, (3) $L{:}\mathbf{DDL}^{\mathbf{I}^{op}} \to \mathbf{DDL}^{\mathbf{I}^{op}}$ is dual to T. Then the equational logic given by $(\Sigma_{\mathbf{DL}} + \Sigma_L, E_{\mathbf{DL}} + E_L)$ is sound and complete for T-coalgebras and characterises T-bisimilarity.*

We can transform the equational logic given by the theorem into a more standard modal logic. Only a brief informal description is given here, examples are presented in Sections 6 and 7. Working with DLs, one cannot represent the order $[\![\phi]\!] \leq [\![\psi]\!]$ in the lattice by an implication $\phi \to \psi$ in the logic. One therefore introduces in the logic a symbol $\leq$, which corresponds to the order in the lattice of formulae.

With the notation from Theorem 1: *Formulas* are formed from operations in $\Sigma_{\mathbf{DL}}$ and Σ_L; *sequents* $\phi \leq \psi$ are pairs consisting of two formulas; each equation $\phi = \psi$ in $E_{\mathbf{DL}}$ or E_L gives rise to *axiom schemes* $\phi \leq \psi$ and $\psi \leq \phi$; the *rules* express that the relation $\leq$ is reflexive and transitive; moreover, we have the congruence rule, which we can write for n-ary monotone operators $\sigma \in \Sigma_{\mathbf{DL}} + \Sigma_L$ as

$$\frac{\phi_i \leq \psi_i \quad 0 \leq i < n}{\sigma(\phi_i) \leq \sigma(\psi_i)}$$

More details can be found in [3].

5 Stone Duality for the π-Calculus

The denotational semantics of π-calculus [6, 15] is given via basic type constructors described by the following meta-language for *functor expression*:

$$H ::= 1 \mid Id \mid N \mid H \times H \mid H + H \mid H_\perp \mid \mathcal{P}H \mid N \to H \mid \delta H\,. \tag{7}$$

In this section, we first review the interpretation T_H of a functor expression H in $\mathbf{SFP}^{\mathbf{I}}_{!}$. Then we are going to give interpretations L_H in $\mathbf{DL}^{\mathbf{I}^{op}}$. The L_H then, restricted to $\mathbf{DDL}^{\mathbf{I}^{op}}$, will be dual to the T_H.

5.1 Domain interpretation of the meta-language

Following [6, 15], we interpret each functor expression H as an endofunctor T_H on $\mathbf{SFP}_!^{\mathbf{I}}$ as follows. The functor T_1 denotes the constant functor mapping to the final object and T_N is the constant functor of *names* defined as the inclusion of $\mathbf{I}$ into $\mathbf{SFP}$ (i.e. $T_N(i)$ is the flat domain $(i)_\perp$). All other functors are defined in terms of a constructor in the category $\mathbf{SFP}_!^{\mathbf{I}}$. For example, $T_{\mathcal{P}H} = \mathcal{P}(T_H)$. The constructors we need are products $\times$, separated sum[6] $+$, lifting $(-)_\perp$, and the powerdomain $\mathcal{P}$ of Abramsky [1]. They are all defined pointwise. Further, δ is for modelling *dynamic allocation* of names, and its interpretation is defined in terms of the monoidal structure $(\mathbf{I}, 0, +)$:

$$\delta X(i) \cong X(i+1)\,. \tag{8}$$

The exponential $N \to H$ models input of names. Due to the structure of $\mathbf{I}$, we have

$$(N \to X)(i) \cong X(i)^i \times X(i+1)\,. \tag{9}$$

The final coalgebra of T_H exists in $\mathbf{SFP}_!^{\mathbf{I}}$. For example, the final coalgebra of the functor Pi given in (1) is the domain theoretic model for strong late bisimilarity used in [6, 15]. In fact, strong late bisimilarity coincides with coalgebraic bisimilarity (p.3).

5.2 Logical interpretation of the meta-language

In this section we interpret functor expressions H as endofunctors on $\mathbf{DL}^{\mathbf{I}^{op}}$. We proceed by presenting each constructor by specific generators and relations (or, operations and equations, see Definition 2). Since terminal object 1, product $\times$, separated sum $+$, lifting $(-)_\perp$, and the Abramsky powerdomain $\mathcal{P}$ are all defined pointwise their presentation is easily derived from Abramsky's description of SFP domains as propositional theories [2]. Of these functors only the Abramsky powerdomain [1] is treated here as an illustrative example. We then give the presentations for names, dynamic allocation, and input. In the following we will omit sort subscripts as in $\Box_i, \Diamond_i$. Further, we assume i, j to range over objects of $\mathbf{I}$ and $\imath$ over arrows $j \to i$ in $\mathbf{I}$.

Abramsky powerdomain $L_{\mathcal{P}}$ is presented by

operations: $\Box, \Diamond : i \to i$ for each i in $\mathbf{I}$

equations:

$$[\imath]\Box v = \Box[\imath]v \qquad [\imath]\Diamond v = \Diamond[\imath]v$$

$$\Box(v_0 \wedge v_1) = \Box v_0 \wedge \Box v_1 \qquad \Diamond(v_0 \vee v_1) = \Diamond v_0 \vee \Diamond v_1$$
$$\Box(v_0 \vee v_1) \leq \Box v_0 \vee \Diamond v_1 \qquad \Box v_0 \wedge \Diamond v_1 \leq \Diamond(v_0 \wedge v_1)$$
$$\Diamond\bot = \bot$$

The difference with the presentation of the Plotkin powerdomain (i.e. the addition of the empty set as a coalesced sum with a lifted one point space) is reflected by the 'missing' relation $\Box\top = \top$. Further, we use $a \leq b$ as a shorthand for the equation $a \wedge b = a$.

Names L_N is presented by

operations: $(s)_\perp : i$ for each $s \subseteq i$

[6] Separated sum is a lifted disjoint union.

equations: $[\iota](s)_\perp = (\iota^{-1}(s))_\perp$

$$(s)_\perp \wedge (s')_\perp = (s \cap s')_\perp \qquad (s)_\perp \vee (s')_\perp = (s \cup s')_\perp \qquad (\perp)_\perp = \perp$$

In the domain interpretation, N maps a finite set of names to the corresponding flat domain. The addition of this extra least element is reflected by $(-)_\perp$ preserving only non-empty meets.

Dynamic allocation L_δ is presented by

operations: $\delta{:}i+1 \to i$
equations: $[\iota]\delta(v) = \delta[\iota+1](v)$

$\delta(-)$ preserves finite joins and finite meets.

Recall (8). The presentation guarantees that the map $A(i+1) \cong (L_\delta A)(i),\ a \mapsto \delta(a)$, is an isomorphism.

Exponential $L_{N\to -}$ is presented by

operations: $n \triangleright (-) : i \to i$ for all i, $n \in i$ and
$() \triangleright (-) : i+1 \to i$ for all i
equations:

$$[\iota](n \triangleright v) = k \triangleright [\iota](v) \quad \text{if } \iota(k) = n$$
$$[\iota](n \triangleright v) = \top \quad \text{if } \iota^{-1}(n) = \emptyset$$
$$[\iota](() \triangleright v) = () \triangleright [\iota+1]v$$

$n \triangleright (-)$ and $() \triangleright (-)$ preserve finite joins and finite meets.

The dual of (9) is an $(i+1)$-fold coproduct, with the coproduct injections corresponding to $n \triangleright (-)$ and $() \triangleright (-)$. The first three equations reflect the non-pointwise nature of the function space.

5.3 Duality of domain and logical interpretation

Each of the presentations above defines a functor L_H on $\mathbf{DL}^{\mathbf{I}^{op}}$, see (7) and Definition 2. That the conditions of Theorem 1 are satisfied is not hard to see in the case of δ and follows from the work of Abramsky [2] for the other constructors. We therefore obtain

Theorem 2. *For every functor expression H, the functor $T_H{:}\mathbf{SFP}^{\mathbf{I}}_! \to \mathbf{SFP}^{\mathbf{I}}_!$ obtained by the domain interpretation and the functor $L_H{:}\mathbf{DDL}^{\mathbf{I}^{op}} \to \mathbf{DDL}^{\mathbf{I}^{op}}$ obtained from the presentations are dual functors (see (5)). In particular, there is an isomorphism η from the final coalgebra of T_H to the dual of the initial algebra of L_H.*

Since functors having a presentation are closed under composition [3], we have that all functors L_H have a presentation. Moreover, this presentation can be obtained in a straightforward way from the presentations of the components, see the following sections for examples. Finally, Theorem 1 shows that the logic obtained from this presentation is sound and complete and characterises bisimilarity.

6 A Logic for Pi-coalgebras

Theorems 1 and 2 together give us a logic for Pi-coalgebras where Pi is the functor described in (1), (8), (9). In this section, we unwind the definitions and give an explicit description in terms of transition systems and modal logic.

A Pi-coalgebra (X, ξ) induces the following transition relation [7]:

$$
\begin{array}{lll}
x \xrightarrow{\tau} x' & \text{iff} & x' \in \xi_i(x),\ x, x' \in X(i) \\
x \xrightarrow{a()} \langle f', x' \rangle & \text{iff} & \langle a, f', x' \rangle \in \xi_i(x), x \in X(i), a \in i, f' : i \to X(i), x' \in X(i+1) \\
x \xrightarrow{\bar{a}\, b} x' & \text{iff} & \langle a, b, x' \rangle \in \xi_i(x),\ x, x' \in X(i),\ a, b \in i \\
x \xrightarrow{\bar{a}()} x' & \text{iff} & \langle a, x' \rangle \in \xi_i(x),\ x \in X(i),\ a \in i,\ x' \in X(i+1)
\end{array}
$$

For $x \in X(i)$, in the case $\xi_i(x) = \bot_{Pi}$ we write $\uparrow x$. The predicate $\uparrow x$ expresses that the state x may diverge. Convergence $\downarrow x$ is defined as not $\uparrow x$ [1]. For Pi-coalgebras, bisimulation can be characterized as ordinary strong late bisimulation (adapted with a divergence predicate).

Theorem 3. *Two convergent states $x, y \in X(i)$ of a Pi-coalgebra (X, ξ) are bisimilar if and only if there exists a symmetric relation $R \subseteq \coprod_{i \in |I|} X(i) \times X(i)$ with xRy and such that for all $x, y \in \coprod_{i \in |I|} X(i)$*

1. *xRy and $\iota : i \rightarrowtail j$ implies $X(\iota)(x) R X(\iota)(y)$;*
2. *xRy implies*
   - *if $x \xrightarrow{\tau} x'$ then there exists y' such that $y \xrightarrow{\tau} y'$ and $x' R y'$;*
   - *if $x \xrightarrow{a()} \langle f, x' \rangle$ then there exists $\langle g, y' \rangle$ such that $y \xrightarrow{a()} \langle g, y' \rangle$, $x' R y'$, and $f(b) R g(b)$ for all $b \in i$;*
   - *if $x \xrightarrow{\bar{a}\, b} x'$ then there exists y' such that $y \xrightarrow{\bar{a}\, b} y'$ and $x' R y'$;*
   - *if $x \xrightarrow{\bar{a}()} x'$ then there exists y' such that $y \xrightarrow{\bar{a}()} y'$ and $x' R y'$.*

Syntax It is convenient to use a two tiered logic, a tier κ for capabilities, and a tier π for non-deterministic processes. For each $i \in |I|$, the sets of capability and process formulas are defined inductively as follows (K a finite set and $\sigma \in \{\pi, \kappa\}$):

$$
\frac{\{\vdash_\sigma \varphi_k : i\}_{k \in K}}{\vdash_\sigma \bigwedge_{k \in K} \varphi_k : i} \qquad
\frac{\{\vdash_\sigma \varphi_k : i\}_{k \in K}}{\vdash_\sigma \bigvee_{k \in K} \varphi_k : i} \qquad
\frac{\iota : i \to j \quad \vdash_\sigma \varphi : j}{\vdash_\sigma [\iota]\varphi : i}
$$

$$
\frac{\vdash_\kappa \psi : i}{\vdash_\pi \Box\psi : i} \qquad
\frac{\vdash_\kappa \psi : i}{\vdash_\pi \Diamond\psi : i}
$$

$$
\frac{\vdash_\pi \phi : i}{\vdash_\kappa \phi : i} \qquad
\frac{a, b \in i \quad \vdash_\pi \phi : i}{\vdash_\kappa a(b) \triangleright \phi : i} \qquad
\frac{a \in i \quad \vdash_\pi \phi : i+1}{\vdash_\kappa a() \triangleright \phi : i}
$$

$$
\frac{a, b \in i \quad \vdash_\pi \phi : i}{\vdash_\kappa \bar{a}\, b \triangleleft \phi : i} \qquad
\frac{a \in i \quad \vdash_\pi \phi : i+1}{\vdash_\kappa \bar{a}() \triangleleft \delta\phi : i}
$$

We write $\top{:}i$ and $\bot{:}i$ for the empty meet and join of formulas of sort i, respectively. The language is based on the presentations of Section 5.2 as follows.[7] Meets and joins come from the distributive lattices, $\Box$ and $\Diamond$ from the powerdomain, input $ab\triangleright$ and $a()\triangleright$ from the exponential, output $\bar{a}\,b\triangleleft$ from the product $N \times N \times -$, and bound output $\bar{a}()\triangleleft\delta$ from dynamic allocation. The notation $\triangleleft$ is introduced here for readability. There are no connectives for the silent action, as we consider any process formula a 'τ'-capability formula.
For $b \notin i$, and $\vdash_\pi \phi{:}i \cup \{b\}$, we write

$$a(\nu b) \triangleright \phi{:}i \quad \text{and} \quad \bar{a}(\nu b) \triangleleft \phi{:}i$$

for the capability formulas $a() \triangleright [\nu b]\phi{:}i$ and $\bar{a}() \triangleleft [\nu b]\delta\phi{:}i$, respectively, where $\nu b{:}i + 1 \to i \cup \{b\}$ is the isomorphism that renames the new name to b. For example, the formula

$$\Diamond a(\nu b) \triangleright (\Box \bar{b}\, a \triangleleft \top){:}\{a\}$$

specifies a process that may receive a new name b along the channel a, and after this it outputs the name a along the channel newly received.

Semantics Given a Pi-coalgebra (X, ξ), and a state $x \in X(i)$ we define the set $C(x)$ of its capabilities as

$$C(x) = \{\bot{:}i | \uparrow x\} \cup \{(\alpha, x') | x \xrightarrow{\alpha} x'\} \cup \{(a(), f, x') | x \xrightarrow{a()} f, x'\}\,.$$

We will use that capabilities $c \in C(x)$ are obtained from the functor $\mathbf{I} \to \mathbf{SFP}_!$

$$C = X + N \times (N \to X) + N \times (N \times X) + N \times \delta X$$

to denote by $C(\iota)(c)$ the application of a renaming $\iota{:}i \to j$ to a capability c of a state $x \in X(i)$. The semantics of process and capability formulas is obtained for each Pi-coalgebra (X, ξ) by interpreting formulas as elements of the initial algebra for the dual of the functor Pi, and applying the semantics of the logic as given in Definition 1 (eliding the definitions for con/disjunctions):

$$\begin{array}{lll}
x \models_\pi [\iota]\phi{:}i & \text{iff} & X(\iota)(x) \models_\pi \phi{:}j \\
x \models_\pi \Box\psi{:}i & \text{iff} & \downarrow x \text{ and } \forall c \in C(x).\, c \models_\kappa \psi{:}i \\
x \models_\pi \Diamond\psi{:}i & \text{iff} & \exists c \in C(x).\, c \models_\kappa \psi{:}i \\
 & & \\
(\tau, x) \models_\kappa \phi{:}i & \text{iff} & x \models_\pi \phi{:}i \\
(a(), \langle f, x\rangle) \models_\kappa ab \triangleright \phi{:}i & \text{iff} & f(b) \models_\pi \phi{:}i \\
(a(), \langle f, x\rangle) \models_\kappa a() \triangleright \phi{:}i & \text{iff} & x \models_\pi \phi{:}i + 1 \\
(\bar{a}\,b, x) \models_\kappa \bar{a}\,b \triangleleft \phi{:}i & \text{iff} & x \models_\pi \phi{:}i \\
(\bar{a}(), x) \models_\kappa \bar{a}() \triangleleft \delta\phi{:}i & \text{iff} & x \models_\pi \phi{:}i + 1
\end{array}$$

[7] A mechanical transformation of the presentations to logical syntax would introduce a tier for each type constructor involved. Although it is quite reasonable to compress them to just a single tier, we find it slightly more convenient to use two.

Theorem 4. *Two convergent states $x, y \in X(i)$ of a Pi-coalgebra (X, ξ) are bisimilar if and only if $x \models_\pi \phi{:}i$ iff $y \models_\pi \phi{:}i$ for every process formula $\phi{:}i$.*

Proof system Axioms and rules for assertions of the form $\varphi_1 \leq_{\sigma,i} \varphi_2$ (with $\sigma \in \{\pi, \kappa\}$ and φ_1, φ_2 formulas of sort $i \in |I|$) are derived from the equations of the presentations in the previous section as explained at the end of Section 4. We just give few axioms and rules as example:

$$\vdash_\pi \Box(\psi_1 \vee \psi_2) \leq_i \Box\psi_1 \vee \Diamond\psi_2 \qquad \frac{\vdash_\kappa \psi_1 \leq_i \psi_2}{\vdash_\pi \Box\psi_1 \leq_i \Box\psi_2}$$

$$\vdash_k a() \triangleright \bot =_i \bot \qquad \frac{\iota{:}i \to j}{\vdash_k [\iota]\iota(a)(\iota(b)) \triangleright \phi =_i a(b) \triangleright [\iota]\phi}$$

$$\frac{\iota{:}i \to j}{\vdash_k [\iota]\iota(a)() \triangleright \phi =_i a() \triangleright [\iota+1]\phi} \qquad \frac{\iota{:}i \to j \quad \iota^{-1}(a) = \emptyset}{\vdash_k [\iota]a() \triangleright \phi =_i \bot}$$

$$\frac{\iota{:}i \to j \quad \iota^{-1}(a) = \emptyset}{\vdash_k [\iota]a(b) \triangleright \phi =_i \bot} \qquad \frac{\iota{:}i \to j \quad \iota^{-1}(b) = \emptyset}{\vdash_k [\iota]\iota(a)(b) \triangleright \phi =_i a() \triangleright \top}$$

We note that $a() \triangleright \top =_i \top$ and $a(b) \triangleright \top =_i \top$ are not derivable because we used the right hand side as a shorthand notation for the formulas $(\{a\})_\bot \times (* \to \top)$, and $(\{a\})_\bot \times (b \to \top)$, respectively, and $(\top)_\bot$ is simply not equal to $\top$.

Theorem 5. *For every Pi-coalgebra (X, ξ), and process formulas ϕ_1 and ϕ_2, we that $x \models_\pi \phi_1{:}i \Rightarrow x \models_\pi \phi_2{:}i$ if and only if $\vdash_\pi \phi_1 \leq_i \phi_2$, that is the logic for strong late bisimulation is sound and complete.*

7 A Logic for the π-Calculus

Of particular interest is the Pi-coalgebra obtained by the interpretation of the π-calculus in the category $\mathbf{SFP}^{\mathbf{I}}$ that is fully abstract with respect to strong late bisimilarity [6, 15]. In fact, for each set of names i, a π-calculus process P can be assigned to an element of the final coalgebra of the functor Pi by the natural transformation

$$[\![P]\!]\rho{:}1 \to \Omega_{Pi}$$

where ρ is an *environment* mapping variables x to an element of the final coalgebra Ω_{Pi} of the functor Pi. We write $[\![P]\!]\rho$ for $([\![P]\!]\rho)_i$ for $[\![P]\!]\rho$ at stage i. This interpretation is fully abstract in the sense that processes with free names in i are identified iff the processes are strong late bisimilar [15].

Using the machinery developed in the previous section we can now construct a logic for the π-calculus that is sound, complete, and characterizes strong late bisimilarity. The process judgement we use is of the form $P, \Gamma \vdash^i_\sigma \varphi$, where P is a π-calculus process, i is a finite set of names including the free names of P, and $\varphi{:}i$ is either a process (and $\sigma = \pi$) or a capability formula (and $\sigma = \kappa$) as defined for Pi-coalgebras. Further, Γ is

a finite set of *assumptions*. We write them in the form $\mathrm{x}{:}_i\psi$ with ψ a process formula of sort i, and assume that Γ contains at most one of these for each variable at each stage.

Validity of Judgements We define $P, \Gamma \models_\pi \phi{:}i$ to hold if for all environments ρ

$$\rho(\mathrm{x})_j \models_\pi \psi{:}j \text{ for all } \mathrm{x}{:}_j\psi \in \Gamma \text{ implies } (\llbracket P \rrbracket \rho)_i \models_\pi \phi{:}i$$

where P is a π-calculus process and $\phi{:}i$ a process formula of sort i.

The Proof System for the π-calculus includes the proof system for Pi-coalgebras to reason about distributive lattices, non-determinism, and process capabilities. More precisely, the latter is incorporated into the former by the following structural *rule of subsumption*:

$$\frac{\vdash_\pi \psi' \leq_j \psi \quad P, \Gamma, \mathrm{x}{:}_j\psi \vdash^i_\sigma \varphi \quad \vdash_\sigma \varphi \leq_i \varphi'}{P, \Gamma, \mathrm{x}{:}_j\psi' \vdash^i_\sigma \varphi'}$$

The structural rules for conjunction, disjunction, and weakening are (K a finite set):

$$\frac{\{P, \Gamma \vdash^i_\sigma \varphi_k\}_{k\in K}}{P, \Gamma \vdash^i_\sigma \bigwedge_{k\in K} \varphi_k} \qquad \frac{\{P, \Gamma, \mathrm{x}{:}_j\psi_k \vdash^i_\sigma \varphi\}_{k\in K}}{P, \Gamma, \mathrm{x}{:}_j \bigvee_{k\in K} \psi_k \vdash^i_\sigma \varphi} \qquad \frac{P, \Gamma \vdash^i_\sigma \varphi}{P, \Gamma, \mathrm{x}{:}_j\psi \vdash^i_\sigma \varphi} \qquad \mathrm{x}, \mathrm{x}{:}_j\psi \vdash^i_\pi \psi$$

Furthermore, there are rules to do deal with the process constructors. Examples of rules for *action prefix* and *sum of processes* are:

$$\frac{\alpha.P, \Gamma \vdash^i_\kappa \psi}{\alpha.P, \Gamma \vdash^i_\pi \Diamond\psi} \qquad\qquad \frac{\alpha.P, \Gamma \vdash^i_\kappa \psi}{\alpha.P, \Gamma \vdash^i_\pi \Box\psi}$$

$$\frac{P, \Gamma \vdash^i \Diamond\psi}{P + Q, \Gamma \vdash^i \Diamond\psi} \qquad \frac{Q, \Gamma \vdash^i \Diamond\psi}{P + Q, \Gamma \vdash^i \Diamond\psi} \qquad \frac{P, \Gamma \vdash^i \Box\psi \quad Q, \Gamma \vdash^i \Box\psi}{P + Q, \Gamma \vdash^i \Box\psi}$$

Similar rules can be given for the parallel composition of processes as it can be decomposed using the auxiliary operators of synchronisation ($||$) and left merge.

More interesting for us are the rules for *input*

$$\frac{P\{n/b\}, \Gamma \vdash^i_\pi \phi}{a(b).P, \Gamma \vdash^i_\kappa ab \triangleright \phi} \qquad \frac{P\{*/b\}, \Gamma \vdash^{i+1} \phi \quad b \notin i}{a(b).P \vdash^i_\kappa a(\nu b) \triangleright \phi}$$

The first rule is about selecting the right behaviour when receiving an old name $n \in i$, whereas the second rule handles the reception of a new name (i.e. $b \notin i$). In both rules we assume $a \neq b$. The rules for *output* and *silent prefixing* are simpler:

$$\frac{P, \Gamma \vdash^i_\pi \phi}{\bar{a}b.P \vdash^i_\kappa \bar{a}b \triangleleft \phi} \qquad \frac{P, \Gamma \vdash^i_\pi \phi}{\tau.P, \Gamma \vdash^i_\kappa \phi}$$

Synchronisation and *process restriction* are defined by considering all possible actions performed by the processes. We give two exemplary sets of rules. The first is about the

restriction of prefixed processes $(\nu b)\alpha.P$. The case when α is a free output assume that b is not the channel where the output take place, since otherwise the process deadlocks, logically reflected by the absence of rules for such a case:

$$\frac{P\{*/b\}, \Gamma \vdash_\pi^{i+1} \phi \quad a \neq b}{(\nu b)\bar{a}b.P, \Gamma \vdash_\kappa^i \bar{a}(\nu b) \triangleleft \phi} \qquad \frac{\alpha.(\nu b)P, \Gamma \vdash_\pi^i \phi \quad b \notin names(\alpha)}{(\nu b)\alpha.P, \Gamma \vdash_\pi^i \phi}$$

The second set of rules is about the synchronisation of an input with a free or a bound output, respectively. Here c if free in both P and Q, and different from a.

$$\frac{P\{b/c\}|Q, \Gamma \vdash_\pi^i \phi}{a(b).P||\bar{a}c.Q, \vdash_\kappa^i \phi} \qquad \frac{(\nu c)(P\{c/b\}|Q), \Gamma \vdash_\pi^i \phi}{a(b).P||(\nu c)\bar{a}c.Q, \vdash_\kappa^i \phi}\,.$$

The fact that both derivations are indexed by κ reflects the effect of the silent action that has taken place because of the synchronisation. We conclude with the rule for *recursive processes*:

$$\frac{\mu x.P, \Gamma \vdash^i \phi_1 \quad \mu x.P, \Gamma[x \mapsto \phi{:}i] \vdash^i \phi_2}{\mu x.P, \Gamma \vdash^i \phi_2}\,.$$

This rules has to be applied finitely many time so to unfold the recursive process.

The main result of our paper states the soundness and completeness of the logic for π-calculus processes.

Theorem 6. *For every process P with free names in i, finite set of assumptions Γ and process formula $\phi{:}i$*

$$P, \Gamma \vdash^i \phi \text{ if and only if } P, \Gamma \models_\pi \phi{:}i\,.$$

This result is proved in the same fashion as for Abramsky's endogenous logic of terms, using the isomorphism η_{Pi} of Theorem 2 between the final coalgebra of T_{Pi} and the dual of the initial algebra of L_{Pi}. Indeed, for each environment ρ we have

$$\eta^{-1}(\{[\phi]_= \mid P, \Gamma \vdash^i \phi\} = (\llbracket P \rrbracket \rho)_i\,.$$

In other words, the elements of the final coalgebra of T_{Pi} given by the semantics of process P at each stage i are identified with the formulas we can prove to hold of P in our logic.

As a consequence of Theorem 6 we have a similar soundness and completeness result. Further, strong late bisimilarity coincides with the logical equivalence. To obtain strong late congruence we have to consider formulas with a renaming modality in the outermost position, that is $P \sim^c Q$ if and only if

$$P, \Gamma \vdash^i [\imath]\phi \text{ if and only if } Q, \Gamma \vdash^i [\imath]\phi\,,$$

for all $\imath{:}i \to j$ in $\mathbf{I}$.

8 Conclusion

Using Stone duality we have developed a logic for strong late bisimulation. Milner, Parrow, Walker [11] have also given a logical characterisation of process equivalences, including strong bisimulation. In contrast to our proposal, their logic is infinitary, it does not have logical connectives for name creation, and no proof system is given.

Dam [4] has also studied a compositional proof system for strong late bisimilarity. Our logical connectives are similar to those of Dam, but he also includes quantifiers and fixed points. A restricted subset of Dam's logic, similar to ours, is equipped with a proof system that is shown to be sound and complete under certain restrictions.

In favour of our approach, we would like to point out that our logic directly reflects the denotational semantics and that, therefore, the logic is modular. For example, the proof that the logic is complete and characterises bisimilarity is done independently for each type constructor involved.

This inherent modularity will allow us to reuse the work of the present paper in future developments. For example, here we interpreted π-calculus process on SFP domains because we wanted to treat recursive terms with a finitary logic. However, restricting the π-calculus to either finite terms or guarded recursion will allow us to interprete processes on sets or Stone spaces, respectively, giving rise to process-logics with negation (and infinitary in the former case). Moreover, variations of the functor Pi will allow us to consider different process equivalences, including early bisimulation and testing equivalences. Fully abstract models based on presheaves for these equivalences are studied in [7] and [8], respectively.

References

1. S. Abramsky. A domain equation for bisimulation. *Inform. and Comput.*, 92, 1991.
2. S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51, 1991.
3. M. Bonsangue and A. Kurz. Presenting functors by operations and equations. In *FoSSaCS'06*, 2006.
4. M. Dam. Proof systems for pi-calculus logics. In *Logic for Concurrency and Synchronisation*. Kluwer, 2003.
5. M. Fiore. Mathematical models of computational and combinatorial structures. In *FoSSaCS*, LNCS 3441, 2005.
6. M. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *LICS 96*, 1996.
7. M. Fiore and D. Turi. Semantics of name and value passing. In *LICS'01*, 2001.
8. M. Hennessy. A fully abstract denotational semantics for the π-calculus. Theorethical Computer Science 278:53–89, 2002.
9. P. Johnstone. *Stone Spaces*. CUP, 1982.
10. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Inform. and Comput.*, 100, 1992.
11. R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoret. Comput. Sci.*, 114, 1993.
12. G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computation*, 5, 1976.
13. J. Rutten. Universal coalgebra: A theory of systems. *Theoret. Comput. Sci.*, 249, 2000.
14. D. Sangiorgi and D. Walker. *The Pi-calculus: A Theory of Mobile Processes*. CUP, 2001.
15. I. Stark. A fully-abstract domain model for the π-calculus. In *LICS 96*, 1996.