Pi-Calculus in Logical Form

M.M. Bonsangue [*]
LIACS - Leiden University
The Netherlands
marcello@liacs.nl

A. Kurz [†]
Department of Computer Science
University of Leicester
United Kingdom
kurz@mcs.le.ac.uk

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 [12, 17] is a process algebra for expressing processes that interact by exchanging channel names via shared channels. Fiore, Moggi and Sangiorgi [7] and Stark [18] 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

$$\begin{tikzcd} \mathcal{X} \arrow[loop left, "T"] \arrow[r, bend left] & \mathcal{A} \arrow[l, bend left] \arrow[loop right, "L"] \end{tikzcd}$$

More generally, in any such situation, the initial L-algebra describes a logic for T-processes that characterises T-bisimilarity. Moreover, by presenting the functor L by operations and equations [5], 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 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 remainder 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 of each type constructor 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 processes in 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.

[*] 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.

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:

$$\begin{array}{lll} \alpha & ::= & a(b) \mid \overline{a}b \mid \tau \\ P & ::= & 0 \mid x \mid \alpha.P \mid [a=b]P \mid [a \neq b]P \mid P+P \mid \\ & & P|P \mid (\nu a)P \mid \mu x.P \end{array}$$

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 $\overline{a}(b)$ 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 [12]. 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 [17] 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 $\iota{:}j \to i$. 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 [7, 18]. 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. In order to describe the action of a functor in $\mathbf{C}^{\mathbf{I}}$ on arrows in $\mathbf{I}$, it is enough to say what it does on bijections $i \cong j$ and on inclusions $i \hookrightarrow i+1$.

Coalgebras Coalgebras are simple mathematical structures for describing dynamical systems like automata and transition systems [16]. 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 [11], or $\mathbf{SFP}$ of SFP domains [15]). 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 [16].

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 [7, 18] 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[1] 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 [11].

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

$$G_\Sigma(A)(s) = \coprod_{\langle s_1 \ldots s_n \rangle \in S^*} G_{\langle s_1 \ldots s_n \rangle} \times A(s_1) \times \ldots \times A(s_n) \tag{1}$$

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 extends the presentation of $\mathbf{A}$ by one unary operator for each arrow in $\mathcal{K}$ and equations for functoriality:

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

1. $\Sigma_{\mathbf{A}^{\mathcal{K}}}$ has an operation op of sort $k, \dots k \to k$ for each operation op in $\Sigma_{\mathbf{A}}$ and $k \in |\mathcal{K}|$, and an operation f of sort $h \to k$ for each arrow $f{:}h \to k$ 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 in the next section, our main interest is in the presheaf category $\mathbf{DL}^{\mathbf{I}^{op}}$.

Example 3.1 $\mathbf{DL}^{\mathbf{I}^{op}}$ *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 $\iota{:}j \to i$ in* $\mathbf{I}$ *there is a unary operator $[\iota]{:}i \to j$. Its equational laws are*

$$[\iota] \bigvee_{k\in K} v_k = \bigvee_{k\in K} [\iota] v_k$$
$$[\iota] \bigwedge_{k\in K} v_k = \bigwedge_{k\in K} [\iota] v_k$$
$$[id_i] v = v,$$
$$[\iota \circ \jmath] v = [\jmath][\iota] v \text{ for } \iota{:}j \to i,\ \jmath{:}k \to j.$$

  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 $A(\iota){:}A(i) \to A(j)$ 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.

Stone duality SFP domains taken with the Scott topology are *spectral spaces* (coherent spaces in [11]). 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 $\mathbf{DL}$ [11, 3]. 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. On morphisms, both $\mathcal{KO}$ and pt are given by inverse image. [2]

$$\begin{tikzcd} \mathbf{Spec} \arrow[r, bend left, "\mathcal{KO}"] & \mathbf{DL} \arrow[l, bend left, "pt"] \end{tikzcd} \qquad (2)$$

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 4.2.

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[3] and $\mathbf{DDL}$ is its dual. An explicit description of $\mathbf{DDL}$ can be found in [2, 3]. 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 (3) 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 [5] to the present situation. It is shown how an initial L-algebra gives rise to an adequate logic for T-coalgebras (Proposition 4.2) and how a presentation of L by operations and equations gives a complete proof calculus for this logic (Theorem 4.5). In this section, we take a rather abstract point of view in order to indicate that our approach is not restricted to the late semantics of π-calculus.

The situation we consider is

$$\begin{tikzcd} Coalg(T) \arrow[r, bend left, "\widetilde{\mathcal{KO}}"] \arrow[d] & Alg(L) \arrow[l, bend left, "\widetilde{pt}"] \arrow[d] \\ \mathbf{SFP}_!^{\mathbf{I}} \arrow[r, bend left, "\widehat{\mathcal{KO}}"] & \mathbf{DDL}^{\mathbf{I}^{op}} \arrow[l, bend left, "\widehat{pt}"] \end{tikzcd} \qquad (3)$$

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

$$d : L\widehat{\mathcal{KO}} \to \widehat{\mathcal{KO}}T. \qquad (4)$$

[2]The functors $\mathcal{KO}$ and pt are contravariant, but we find the diagrams more readable if this is not made notationally explicit.

[3]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.

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

Definition 4.1 *The algebra of formulas is the initial L-algebra.*[4] *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 $\mathit{Coalg}(T) \models (\phi \leq \psi)$ if $(\!|\phi|\!)_{(X,\xi)} \subseteq (\!|\psi|\!)_{(X,\xi)}$ for all coalgebras.

Proposition 4.2 *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: $\mathit{Coalg}(T) \models (\phi \leq \psi)$ 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 $\mathit{Alg}(L)$ 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 straightforward (see [5] for a full discussion).

Theorem 4.5 is the basis of our results in the following sections. It states that, in the situation of Diagram (3), given an equational axiomatisation $(\Sigma_{\mathbf{DL}}, E_{\mathbf{DL}})$ of distributive lattices and a presentation $\langle \Sigma_L, E_L\rangle$ of L, one obtains a complete axiomatisation $(\Sigma_{\mathbf{DL}} + \Sigma_L, E_{\mathbf{DL}} + E_L)$ of $\mathit{Alg}(L)$ and hence $\mathit{Coalg}(T)$. We will use this result but for this paper the technical details do not matter so much and the reader might wish to continue directly with Section 5.

Definition 4.3 *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 (1)) 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*

$$\begin{tikzcd} FE_V \arrow[r, shift left] \arrow[r, shift right] & FG_\Sigma UFV \arrow[r, "FG_\Sigma Uv"] & FG_\Sigma UA \arrow[r] & LA \end{tikzcd} \quad (5)$$

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

Remark 4.4
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.*
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 (5), 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 sum and a restricted form of function space where equations are enough.*

As the proof of Proposition 4.2 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 4.5 *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.*

5. Stone Duality for the π-Calculus

Based on [7, 18], we use coalgebras for modelling π-calculus processes. In particular we consider the following functor Pi on $\mathbf{SFP}^{\mathbf{I}}_{!}$

$$\mathcal{P}(- + N \times (N \to -) + N \times (N \times -) + N \times \delta -)\,. \quad (6)$$

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

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.

(6) is an example of a *functor expression* in the meta-language

$$H ::= 1|Id|N|H\times H|H+H|H_\perp|\mathcal{P}H|N\to H|\delta H \quad (7)$$

We will 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, allowing to apply Theorem 4.5.

5.1. Domain interpretation

Each functor expression H is interpreted as an endofunctor T_H on $\mathbf{SFP}_!^{\mathbf{I}}$ as follows. 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[5] $+$, 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 $(\mathbf{I}, 0, +)$:

$$\delta X(i) \quad\cong\quad X(i+1)\,. \qquad (8)$$

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

$$(N \to X)(i) \quad\cong\quad X(i)^i \times X(i+1) \qquad (9)$$

On arrows ι, $(N \to X)(\iota)$ maps (f, x) with $f{:}i \to X(i)$, $x \in X(i+1)$ to one of

$$(\,X(\iota)\circ f\circ \iota^{-1}\,,\,X(\iota+1)(x)\,) \qquad (10)$$
$$(\,f\,,\,x\,,\,X(\iota+1)(x)\,) \qquad (11)$$

depending on whether ι is an isomorphism (10) or the inclusion $i \hookrightarrow i+1$ (11). Informally, δX is determined by X using one new name, while $N \to X$ uses all existing names and a new one.

The final coalgebra of T_H exists in $\mathbf{SFP}_!^{\mathbf{I}}$. In particular, the final coalgebra of the functor Pi (6) is the domain theoretic model for strong late bisimilarity of [7, 18] (strong late bisimilarity coincides with coalgebraic bisimilarity (p.2)).

[5] Separated sum is a lifted disjoint union.

5.2. Logical interpretation

In this section we interpret functor expressions H as functors on $\mathbf{DL}^{\mathbf{I}^{op}}$. We proceed by presenting each type constructor by generators and relations (or, operations and equations, see Definition 4.3). Since terminal object 1, product $\times$, separated sum $+$, lifting $(-)_\perp$, and powerdomain $\mathcal{P}$ are all defined pointwise their presentation is easily derived from Abramsky's description of SFP domains as propositional theories [2]. We treat here Abramsky powerdomain [1] and separated sum. 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 ι 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: $[\iota]\Box v = \Box[\iota]v \qquad [\iota]\Diamond v = \Diamond[\iota]v$

$$\begin{array}{l}\Box(v_0\wedge v_1) = \Box v_0 \wedge \Box v_1\\ \Diamond \bigvee_{k\in K} v_k = \bigvee_{k\in K}\Diamond v_k\\ \Box(v_0\vee v_1) \le \Box v_0 \vee \Diamond v_1\\ \Box v_0 \wedge \Diamond v_1 \le \Diamond(v_0\wedge v_1)\end{array}$$

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 $a \le b$ as a shorthand for the equation $a \wedge b = a$.

Separated Sum L_{-+-} is presented by

operations: $l{:}i \to i$, $r{:}i \to i$

equations: $[\iota]l(v) = l([\iota]v) \qquad [\iota]r(v) = r([\iota]v)$
$l(v_0) \wedge r(v_1) = \bot$
l, r preserve binary meets and finite joins

That $+$ is interpreted as separated sum is reflected in the 'missing' equation $l(\top) \vee r(\top) = \top$.

Names L_N is presented by

operations: $a{:}1 \to i$ for each $a \in i$

equations: $[\iota]a = \bot \quad$ if a not in the image of ι
$[\iota]\iota(a) = a$
$a \wedge a' = \bot \quad$ if $a \neq a'$

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 the 'missing' equation $\bigvee\{a \mid a \in i\} = \top$. The operations a here have arity zero.

Product with names $L_{N\times -}$ is presented by operations: $a - :i \to i$ for each $a \in i$

equations: $[\iota]av = \bot$ if a not in the image of ι
$[\iota]\iota(a)v = a[\iota]v$
$av_0 \wedge a'v_1 = \bot$ if $a \neq a'$
$a-$ preserves binary meets and finite joins

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

equations: for all isomorphisms $\iota{:}j \cong i$
$[\iota](\iota(n) \rhd v) \;=\; n \rhd [\iota](v)$
$[\iota](() \rhd v) \;=\; () \rhd [\iota+1]v$

for all inclusions $\iota{:}i \hookrightarrow i+1$
$[\iota](n \rhd v) \;=\; n \rhd [\iota](v)$ if $n \in i$
$[\iota](n \rhd v) \;=\; () \rhd v$ if $n \notin i$
$[\iota](() \rhd v) \;=\; () \rhd [\iota+1]v$

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

The dual of (9) is an $(i+1)$-fold coproduct, with the coproduct injections corresponding to $n \rhd (-)$ and $() \rhd (-)$. The equations reflect the non-pointwise nature of the function space: The first two equations describe (10), the next three ones (11).

5.3. Duality of the two interpretations

Each of the presentations above defines a functor L_H on $\mathbf{DL}^{\mathbf{I}^{op}}$, see (7) and Definition 4.3. That the conditions of Theorem 4.5 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 5.1 *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 (4)). 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 [5], 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 next section for an example. Finally, Theorem 4.5 shows that the logic obtained from this presentation is sound and complete and characterises bisimilarity.

6. A Logic for Pi-coalgebras

Theorems 4.5 and 5.1 give us a logic for Pi-coalgebras (6). In this section, we unwind the definitions and give an explicit description in terms of transition systems and modal logic. As corollaries we obtain Theorem 6.4 and 6.5.

The functor Pi (6) can be decomposed into $Pi = \mathcal{P}H$ where $HX = X + N \times (N \to X) + N \times (N \times X) + N \times \delta X$. Given a state x in $X(i)$ of a coalgebra, a continuation, or capability, in $HX(i)$ is chosen non-deterministically. We use the following notation [8] (note that, as to be expected for a late semantics, the continuation of the input-clause is an abstraction):

$x \xrightarrow{\tau} x'$	iff	$x' \in \xi_i(x),\ x, x' \in X(i)$
$x \xrightarrow{a(b)} \langle f', x'\rangle$	iff	$\langle a, f', x'\rangle \in \xi_i(x), x \in X(i), a \in i$ $b \notin i, f'{:}i \to X(i), x' \in X(i+\{b\})$
$x \xrightarrow{\overline{a}b} x'$	iff	$\langle a, b, x'\rangle \in \xi_i(x),\ x, x' \in X(i),$ $a, b \in i$
$x \xrightarrow{\overline{a}(b)} x'$	iff	$\langle a, x'\rangle \in \xi_i(x),\ x \in X(i),\ a \in i,$ $b \notin i,\ x' \in X(i+\{b\})$

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 characterised as ordinary strong late bisimulation (adapted with a divergence predicate).

Proposition 6.1 (based on [8]) *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 $X(\iota)(x)RX(\iota)(y)$;*
2. *xRy implies*
   - *if $x \xrightarrow{\tau} x'$ then there exists y' such that $y \xrightarrow{\tau} y'$ and $x'Ry'$;*
   - *if $x \xrightarrow{a(b)} \langle f, x'\rangle$ then there exists $\langle g, y'\rangle$ such that $y \xrightarrow{a(b)} \langle g, y'\rangle$, $x'Ry'$, and $f(c)Rg(c)$ for all $c \in i$;*

- *if $x \xrightarrow{\overline{a}b} x'$ then there exists y' such that $y \xrightarrow{\overline{a}b} y'$ and $x'Ry'$;*
- *if $x \xrightarrow{\overline{a}(b)} x'$ then there exists y' such that $y \xrightarrow{\overline{a}(b)} y'$ and $x'Ry'$.*

Syntax Corresponding to the decomposition $Pi = \mathcal{P}H$, we 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 $\sigma \in \{\pi, \kappa\}$):

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

$$\frac{\vdash_\kappa \psi{:}i}{\vdash_\pi \Box\psi{:}i} \quad \frac{\vdash_\kappa \psi{:}i}{\vdash_\pi \Diamond\psi{:}i} \quad \frac{\vdash_\pi \phi{:}i}{\vdash_\kappa \tau.\phi{:}i}$$

$$\frac{a,b\in i \quad \vdash_\pi \phi{:}i}{\vdash_\kappa ab \rhd \phi{:}i} \quad \frac{a\in i \quad b\notin i \quad \vdash_\pi \phi{:}i+\{b\}}{\vdash_\kappa a(\nu b)\rhd \phi{:}i}$$

$$\frac{a,b\in i \quad \vdash_\pi \phi{:}i}{\vdash_\kappa \overline{a}b \lhd \phi{:}i} \quad \frac{a\in i \quad b\notin i \quad \vdash_\pi \phi{:}i+\{b\}}{\vdash_\kappa \overline{a}(\nu b)\lhd \phi{:}i}$$

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. Meets and joins come from the distributive lattices, $\Box$ and $\Diamond$ from the powerdomain, input $ab\rhd$ and $a(\nu b)\rhd$ from $N \times (N \to -)$, output $\overline{a}b\lhd$ from $N \times N \times -$, and bound output $\overline{a}(\nu b) \lhd -$ from $N \times \delta-$ (the notation $\lhd$ is introduced here for readability).

For example, the formula

$$\Diamond a(\nu b) \rhd (\Box \overline{b} a \lhd \top){:}\{a\}$$

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 The semantics of process formulas is obtained by interpreting formulas as elements of the initial algebra for the dual of the functor $Pi = \mathcal{P}H$ as in Definition 4.1. Eliding the clauses for con/disjunctions, we obtain:

$$\begin{array}{lll} x \models_\pi [\iota]\phi{:}i & \text{iff} & X(\iota)(x) \models_\pi \phi{:}j \\ x \models_\pi \Box\psi{:}i & \text{iff} & \downarrow\! x \text{ and } \forall c \in \xi(x).c \models_\kappa \psi{:}i \\ x \models_\pi \Diamond\psi{:}i & \text{iff} & \exists c \in C(x).c \models_\kappa \psi{:}i \end{array}$$

$$\begin{array}{lll} c \models_\kappa [\iota]\phi{:}i & \text{iff} & H(\iota)(c) \models_\kappa \phi{:}j \\ (\tau, x) \models_\kappa \tau.\phi{:}i & \text{iff} & x \models_\pi \phi{:}i \\ (a(b), \langle f, x\rangle) \models_\kappa ac \rhd \phi{:}i & \text{iff} & f(c) \models_\pi \phi{:}i \\ (a(b), \langle f, x\rangle) \models_\kappa a(\nu b) \rhd \phi{:}i & \text{iff} & x \models_\pi \phi{:}i + \{b\} \\ (\overline{a}b, x) \models_\kappa \overline{a}b \lhd \phi{:}i & \text{iff} & x \models_\pi \phi{:}i \\ (\overline{a}(b), x) \models_\kappa \overline{a}(\nu b) \lhd \phi{:}i & \text{iff} & x \models_\pi \phi{:}i + \{b\} \end{array}$$

Example 6.2 *Consider a Pi-coalgebra (X, ξ) that has states $x, y \in X(\{a, b\})$ whose behaviour conforms to the π-calculus expressions (taken from [17]) $x \cong a(c).0 + a(c).\overline{c}a.0$ and $y \cong a(c).0 + a(c).\overline{c}a.0 + a(c).[c = b]\overline{c}a.0$. These two processes are early-bisimilar but not late-bisimilar. They are distinguished by the formula $\Diamond(ab \rhd \Diamond\top \wedge aa \rhd \Box\bot)$ which is—due to the presence of the third summand—satisfied by y but not by x.*

Remark 6.3 *The expressivity of our logic is the same as of the logic of [13] (with the late quantifiers $\langle a(c)\rangle^{\mathrm{L}}$) because both logics characterise late-bisimilarity. The formula $\langle a(c)\rangle^{\mathrm{L}}\phi$ of [13] corresponds to our—assuming ϕ has free variables in $i + \{c\}$—$\Diamond((\bigwedge_{b\in i} ab \rhd \phi_b) \wedge a(\nu c) \rhd \phi)$ where ϕ_b correspond to ϕ with b substituted for c. Conversely, for example, our $\Diamond(ab \rhd \phi)$ corresponds to [13]'s $\langle a(c)\rangle^{\mathrm{L}}[c = b]\phi$, which provides an explanation why the equality predicate $[c = b]\phi$ is needed in [13].*

Our logic is closely related to the modal fragment of the logic in [6]. More precisely, [6]'s formulas $\langle a\rangle(b \to \phi)$, $\langle a\rangle(b \leftarrow \phi)$, $\langle a\rangle(\nu b \to \phi)$, $\langle a\rangle(\nu b \leftarrow \phi)$ correspond, respectively, to our $\Diamond(ab \rhd \phi)$, $\Diamond(\overline{a}b \lhd \phi)$, $\Diamond(a(\nu b) \rhd \phi)$, $\Diamond(\overline{a}(\nu b) \lhd \phi)$.

Theorem 6.4 *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 The presentations of Section 5.2 together with Theorem 4.5 give us a complete equational axiomatisation for *Pi*-coalgebras. Using well-known techniques, it is straightforward to transform this equational logic into a more standard modal logic as follows. Working with distributive lattices (as opposed to boolean algebras), the order in the lattice cannot be represented by an implication in the logic. We therefore use *sequents* $\varphi_0 \leq \varphi_1$ consisting of two formulas. Axioms state that $\leq$ is reflexive and transitive; moreover, there are congruence rules

$$\frac{\vdash_\kappa \psi_1 \leq \psi_2}{\vdash_\pi \nabla\psi_1 \leq \nabla\psi_2} \quad \frac{\vdash_\pi \phi_1 \leq \phi_2}{\vdash_\kappa \Delta\phi_1 \leq \Delta\phi_2} \quad \frac{\vdash_\sigma \varphi_1 \leq \varphi_2}{\vdash_\sigma [\iota]\varphi_1 \leq [\iota]\varphi_2}$$

where σ ranges over $\{\kappa, \pi\}$, ∇ over $\{\Box, \Diamond\}$ and Δ over $\{ab\rhd, a(\nu b)\rhd, \overline{a}b\lhd, \overline{a}(\nu b)\lhd\}$. Then we have those axioms and rules which give the formulas the structure of an $|\mathbf{I}|$-sorted distributive lattice.

Furthermore, each equation $\varphi_1 = \varphi_2$ obtained from the presentations in Section 5.2 gives rise to *axiom schemes* $\vdash_\sigma \varphi_1 \leq \varphi_2$ and $\vdash_\sigma \varphi_2 \leq \varphi_1$. For example the last equation of the presentation of $L_{\mathcal{P}}$ yields the axiom (scheme) $\vdash_\pi \Box\psi_0 \wedge \Diamond\psi_1 \leq \Diamond(\psi_0 \wedge \psi_1)$. From now on we drop the $\vdash_\sigma$ from axioms and rules.

We treat the input capability as an example, the axioms for the other capabilities being derived similarly. For better readability, the logic uses the modalities $ab \rhd -$ and $a(\nu c) \rhd -$, which combine several types of operations: (1) an injection for L_{-+-}, (2) operations $a-$ for $L_{N\times -}$ and (3) $b \rhd -$ and $() \rhd -$, respectively, for $L_{N\to -}$. Since all these operations preserve binary meets and finite joins, we obtain (for K a finite set)

$$\begin{aligned} ab \rhd (\phi_0 \wedge \phi_1) &= ab \rhd \phi_0 \wedge ab \rhd \phi_1 \\ ab \rhd (\textstyle\bigvee_{k\in K} \phi_k) &= \textstyle\bigvee_{k\in K} ab \rhd \phi_k \end{aligned}$$

and similarly for $a(\nu c) \rhd -$. The 1st and 2nd equation of the presentation of $L_{N\times -}$ give rise to, respectively,

$$\frac{\iota{:}i \hookrightarrow i + \{a\}}{[\iota]ab \rhd \phi =_i \bot} \qquad \frac{\iota{:}i \cong j}{[\iota]\iota(a)\iota(b) \rhd \phi =_i ab \rhd [\iota]\phi}$$

where $i + \{a\}$ implies $a \notin i$. The right-hand axiom also includes the 1st equation of $L_{N\to -}$. Using the 2nd equation instead, the analogous axioms for bound input become

$$\frac{\iota{:}i \hookrightarrow i + \{a\}}{[\iota]a(\nu b) \rhd \phi =_i \bot} \qquad \frac{\iota{:}j \cong i}{[\iota]\iota(a)(\nu b) \rhd \phi =_j a(\nu b) \rhd [\iota + \{b\}]\phi}$$

The 3rd and 4th equation of $L_{N\to -}$ give

$$\frac{\iota{:}i \hookrightarrow i + 1 \quad a, b \in i}{[\iota]ab \rhd \phi =_i ab \rhd [\iota]\phi} \qquad \frac{\iota{:}i \hookrightarrow i + \{b\} \quad a \in i}{[\iota]ab \rhd \phi =_i a(\nu b) \rhd \phi}$$

The left-hand one says that extending the set of free names i has no effect on an input capability if both the subject a and the object b of the input are already known. However (see right-hand axiom) extending the set of free names i with a new name b received has object of an input on a is the same as a bound input on a.

Finally, the 2nd equation of L_{-+-} and the 3rd equation of $L_{N\times -}$ give rise to 'disjointness' axioms stating that two different capabilities cannot happen 'simultaneously'. For example, we have

$$\tau.\phi_0 \wedge ab \rhd \phi_1 = \bot \qquad ab \rhd \phi_0 \wedge cd \rhd \phi_1 = \bot \ \ (a \neq c)$$

The fact that we don't have the proviso ($b \neq d$) for the right-hand axiom reflects the lateness of input (eg for output we have $\overline{a}b \lhd \phi_0 \wedge \overline{c}d \lhd \phi_1 = \bot$ if $a \neq c$ or $b \neq d$).

Theorem 6.5 *For every Pi-coalgebra (X, ξ), and process formulas ϕ_1 and ϕ_2, we have 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 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 [7, 18]. 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 $((\!|P|\!)\rho)_i$ 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 [7, 18] (the $((\!|P|\!)\rho)_i$ corresponds to the closed interpretation of [7]).

Using the machinery developed in the previous section we can now construct a logic for the π-calculus that is sound, complete, and characterises strong late bisimilarity. The process judgement we use is of the form $\Gamma \vdash_{i,\sigma} P{::}\varphi$, 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 ($\sigma = \pi$) or a capability formula ($\sigma = \kappa$) as defined for Pi-coalgebras. Further, Γ is a finite set of *assumptions*. We write them in the form $\mathrm{x} \mapsto \phi$ with ϕ a process formula of sort i, and assume that Γ contains at most one of these for each variable.

Validity of Judgements We define $\Gamma \models^i_\sigma P{::}\varphi$ to hold if for all environments ρ

$$(\text{for all } \mathrm{x} \mapsto \phi \in \Gamma\ .\ \rho(\mathrm{x})_i \models_\pi \phi{:}i) \ \ \Rightarrow \ \ ((\!|P|\!)\rho)_i \models_\sigma \varphi{:}i$$

where P is a π-calculus process and $\varphi{:}i$ a process or capability 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*:

$$\frac{\phi_1 \leq_i \phi_0 \quad \Gamma, \mathrm{x} \mapsto \phi_0 \vdash_{i,\sigma} P{::}\varphi_0 \quad \varphi_0 \leq_i \varphi_1}{\Gamma, \mathrm{x} \mapsto \phi_1 \vdash_{i,\sigma} P{::}\varphi_1}$$

The structural rules for conjunction, disjunction, and weakening are standard.

Furthermore, there are rules to do deal with the process constructors. Examples of rules are:

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

$$\frac{\Gamma \vdash_{i,\pi} P{::}\Box\psi \quad \Gamma \vdash_{i,\pi} Q{::}\Box\psi}{\Gamma \vdash_{i,\pi} P + Q{::}\Box\psi}$$

$$\frac{\Gamma \vdash_{i,\pi} P{::}\Diamond\psi}{\Gamma \vdash_{i,\pi} P + Q{::}\Diamond\psi} \qquad \frac{\Gamma \vdash_{i,\pi} Q{::}\Diamond\psi}{\Gamma \vdash_{i,\pi} P + Q{::}\Diamond\psi}\,.$$

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 ($\lfloor\!\lfloor$).

More interesting for us are the rules for *input*

$$\frac{\Gamma \vdash_{i,\pi} P\{c/b\}::\phi \quad b \notin i}{\Gamma \vdash_{i,\kappa} a(b).P::ac \triangleright \phi} \qquad \frac{\Gamma \vdash_{i+\{b\},\pi} P::\phi \quad b \notin i}{\Gamma \vdash_{i,\kappa} a(b).P::a(\nu b) \triangleright \phi}.$$

The first rule is about selecting the right behaviour when receiving an old name $c \in i$, whereas the second rule handles the reception of a new name (i.e. $b \notin i$). In both rules we have $a \neq b$ because $a \in i$ by well-definedness of the formulas, whereas b is assumed not to be in i. The rules for *output* and *silent prefixing* are simpler:

$$\frac{\Gamma \vdash_{i,\pi} P::\phi}{\Gamma \vdash_{i,\kappa} \overline{a}b.P::\overline{a}b \triangleleft \phi} \qquad \frac{\Gamma \vdash_{i,\pi} P::\phi}{\Gamma \vdash_{i,\kappa} \tau.P::\tau.\phi}$$

Process restriction and *synchronisation* are defined by considering all possible actions performed by the processes. We give two exemplary rules.

$$\frac{\Gamma \vdash_{i,\kappa} P::bc \triangleright \phi \quad a \notin \{b,c\}}{\Gamma \vdash_{i,\kappa} (\nu a)P::bc \triangleright \phi}$$

$$\frac{\Gamma \vdash_{i+\{b\},\kappa} P::\overline{a}b \triangleleft \phi \quad b \notin i}{\Gamma \vdash_{i,\kappa} (\nu b)\overline{a}b.P::\overline{a}(\nu b) \triangleleft \phi}$$

We conclude with the rule for *recursive processes*:

$$\frac{\Gamma \vdash_{i,\pi} \mu x.P::\phi_0 \quad \Gamma[x \mapsto \phi_0] \vdash_{i,\pi} P::\phi_1}{\Gamma \vdash_{i,\pi} \mu x.P::\phi_1}.$$

This rules has to be applied finitely many time so to unfold the recursive process $\mu x.P$ as much as necessary.

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

Theorem 7.1 *For every process P with free names in i, finite set of assumptions Γ and process formula $\phi{:}i$*

$$\Gamma \vdash_{i,\pi} P::\phi \quad \textit{if and only if} \quad \Gamma \models_{\pi} P::\phi{:}i\,.$$

This result is proved in the same fashion as for Abramsky's endogenous logic of terms, using the isomorphism η_{Pi} of Theorem 5.1 between the final coalgebra of T_{Pi} and the dual of the initial algebra of L_{Pi}. Indeed, for each environment ρ we have

$$\eta^{-1}(\{[\phi]_= \mid \Gamma \vdash_{i,pi} P::\phi\} = (\!|P|\!)\rho)_i\,.$$

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 7.1 we have a similar soundness and completeness result. Further, strong late bisimilarity coincides with the logical equivalence.

Strong late congruence Strong late bisimilarity is not closed under input prefix [17]: to obtain a congruence, processes have to be bisimilar under *all* name substitutions.

Following Stark [18], for a set of names i, we write N^i for the functor $\mathbf{I} \to \mathbf{SFP}$ obtained as the $|i|$-fold product of N. This way, elements of N^i_j are (not necessarily injective) substitutions $\varsigma{:}i \to j$. For any process P with free names in i and environment ρ mapping each variable x to an element of the final coalgebra Ω_{Pi} of the functor Pi, we define a morphism $[\![P]\!]^i\rho{:}N^i \to \Omega_{Pi}$ by taking

$$([\![P]\!]^i\rho)_j(\varsigma) = ((\!|P\varsigma|\!)\rho)_j,$$

for any set of name $j \in |\mathbf{I}|$ and substitution $\varsigma \in N^i_j$, where $P\varsigma$ is the process P with all free names $a \in i$ substituted by $\varsigma(a)$ (this corresponds to the *open* interpretation of [7]) .

From the logical point of view, for each $i \in |\mathbf{I}|$ we extend the syntax of our logic for the functor Pi with another tier 'c' for lifting processes formulas to morphisms. Its syntax consists of the usual rules for conjunctions, disjunctions and injective renaming plus the following rule

$$\frac{\varsigma \in N^i_j \quad \vdash_{\pi} \phi{:}j}{\vdash_c \varsigma \to \phi{:}i}.$$

These formulas are used in the process judgement for strong late congruence. The latter is of the form $\Gamma \vdash^i_{j,c} P::\theta$, where P is a π-calculus process with free names in i, $\theta{:}i$ is a formula as described above, and Γ is a finite set of assumptions, each of the form $\mathrm{x} \mapsto \phi \in \Gamma$ where ϕ is an *ordinary* process formula of sort j.

The semantics of such a process judgement is obtained by interpreting the extended formulas as a morphisms from N^i_j to the initial algebra for the dual of the functor Pi. In particular, for any π-calculus process P with free names in i, and *extended* process formula $\varsigma \to \phi{:}i$ with $\varsigma{:}i \to j$ a (not necessarily injective) substitution, we say

$$\Gamma \models^i_{j,c} P::\varsigma \to \phi$$

to hold if and only if, for all environments ρ

$$(\text{for all } \mathrm{x} \mapsto \phi' \in \Gamma\ .\ \rho(\mathrm{x})_j \models_{\pi} \phi'{:}j) \;\Rightarrow\; ([\![P]\!]^i\rho)_j(\varsigma) \models_{\pi} \phi{:}j$$

The other cases of conjunctions, disjunctions and injective renaming of extended formulas are treated as expected.

A sound and complete proof system for strong late congruence is obtained by extending the judgment for strong late bisimulation with the following rule:

$$\frac{\Gamma \vdash_{j,\pi} P\varsigma::\phi}{\Gamma \vdash^i_{j,c} P::\varsigma \to \phi}.$$

Further, the following axioms are needed for relating the extended process formulas with the other logical operators:

$$\varsigma \to \bigwedge_{k\in K} \phi_k = \bigwedge_{k\in K}(\varsigma \to \phi_k)$$

$$\varsigma \to \bigvee_{k\in K} \phi_k = \bigvee_{k\in K}(\varsigma \to \phi_k) \qquad \frac{\varsigma_k \neq \varsigma_{k'} \qquad k, k' \in K}{\bigvee_{k\in K}(\varsigma_k \to \phi) = \bot}$$

$$\frac{\imath{:}i \to j \qquad \varsigma \in N^j_k}{[\imath]\varsigma \to \phi = \varsigma \circ \imath \to \phi} \qquad \frac{\varsigma \in N^i_j \qquad \jmath{:}j \to k}{\varsigma \to [\jmath]\phi = \jmath \circ \varsigma \to \phi}$$

For this system, strong late congruence coincides with the logical equivalence.

8. Conclusion

We presented a case study of the Stone duality approach to logics for coalgebras [4]. The originality of this paper is to show that logics for process calculi with name-binding can be obtained from a straightforward reuse of existing ingredients, which are, in our example, the coalgebraic semantics of late-bisimilarity [12, 17] and the domain theory in logical form of [2]. The same method will allow us to treat other process calculi and process equivalences and to build a uniform theory of their logics. Some immediate directions of further work are pointed out below.

Remark 6.3 indicates a comparison with the logics of Milner, Parrow, Walker [13] and Dam [6] on which we briefly comment.

Milner, Parrow, and Walker [13] also gives 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 [6] also studies a compositional proof system for the π-calculus. Our logical connectives are similar, but his logic also includes first-order quantifiers and fixed points. A restricted subset, similar to ours, is equipped with a proof system that is shown to be complete.

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. In our case, only the input constructor $N \to -$ required attention, the presentations of the other constructors being known from the literature.

This inherent modularity will allow us to reuse the work of the present paper in future developments. For example, here we interpreted π-calculus processes 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 interpret 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 [8, 9] and [10], 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] S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic in Computer Science*. OUP, 1994.

[4] M. Bonsangue and A. Kurz. Duality for logics of transition systems. In *FoSSaCS'05*, 2005.

[5] M. Bonsangue and A. Kurz. Presenting functors by operations and equations. In *FoSSaCS'06*, 2006.

[6] M. Dam. Proof systems for pi-calculus logics. In *Logic for Concurrency and Synchronisation*. Kluwer, 2003.

[7] M. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *LICS 96*, 1996.

[8] M. Fiore and D. Turi. Semantics of name and value passing. In *LICS'01*, 2001.

[9] M. Fiore and S. Staton. Comparing operational models of name-passing process calculi. *Inform. and Comput.*, 204, 2006.

[10] M. Hennessy. A fully abstract denotational semantics for the π-calculus. *Theoret. Comput. Sci.*, 278, 2002.

[11] P. Johnstone. *Stone Spaces*. CUP, 1982.

[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Inform. and Comput.*, 100, 1992.

[13] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoret. Comput. Sci.*, 114, 1993.

[14] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Inform. and Comput.*, 120, 1995.

[15] G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computation*, 5, 1976.

[16] J. Rutten. Universal coalgebra: A theory of systems. *Theoret. Comput. Sci.*, 249, 2000.

[17] D. Sangiorgi and D. Walker. *The Pi-calculus: A Theory of Mobile Processes*. CUP, 2001.

[18] I. Stark. A fully-abstract domain model for the π-calculus. In *LICS 96*, 1996.