J Autom Reasoning
DOI 10.1007/s10817-011-9224-3

A Presheaf Environment for the Explicit Fusion Calculus

Filippo Bonchi · Maria Grazia Buscemi · Vincenzo Ciancia · Fabio Gadducci

Received: 23 February 2011 / Accepted: 23 February 2011

Abstract Name passing calculi are nowadays one of the preferred formalisms for the specification of concurrent and distributed systems with a dynamically evolving topology. Despite their widespread adoption as a theoretical tool, though, they still face some unresolved semantic issues, since the standard operational, denotational and logical methods often proved inadequate to reason about these formalisms. A domain which has been successfully employed for languages with asymmetric communication, like the π-calculus, are presheaf categories based on (injective) relabellings, such as $Set^{\mathbb{I}}$. Calculi with *symmetric* binding, in the spirit of the *fusion calculus*, give rise to novel research challenges. In this work we examine the *explicit fusion*

This work was carried out during the first author's tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The third author has been supported by the Comunidad de Madrid program ProMeSaS (S-0505/TIC/0407) and by the Netherlands Organization for Scientific Research VICI grant 639.073.501. The fourth author has been partly supported by the Italian Ministry of University and Research project SisteR (PRIN 20088HXMYN).

F. Bonchi
Laboratoire de l'Informatique du Parallélisme, CNRS and ENS Lyon, 46 Allée d'Italie, Lyon 69364, France
e-mail: filippo.bonchi@ens-lyon.fr

M. G. Buscemi
IMT Lucca Institute for Advanced Studies, Piazza S. Ponziano 6, 55100 Lucca, Italy
e-mail: m.buscemi@imtlucca.it

V. Ciancia
Institute for Logic, Language and Computation, University of Amsterdam, P.O. Box 94242, 1090 GE Amsterdam, The Netherlands
e-mail: vincenzoml@gmail.com

F. Gadducci (✉)
Dipartimento di Informatica, University of Pisa, Largo Pontecorvo 3c, 56127 Pisa, Italy
e-mail: gadducci@di.unipi.it

Published online: 17 March 2011

calculus, and propose to model its syntax and semantics using the presheaf category $Set^{\mathbb{E}}$, where $\mathbb{E}$ is the category of equivalence relations and equivalence preserving morphisms.

Keywords Algebras · Coalgebras · Denotational semantics · Nominal calculi · Presheaf categories

1 Introduction

On denotational semantics. Denotational semantics was introduced [31] as a formal way for specifying the meaning of programming languages: to each expression of the language (to each *program*) a *denotation* is assigned, i.e., an object in a mathematical domain. In the original proposal, each program denotes a continuous function on a partially ordered set, mapping an input of the program into the corresponding output. An important tenet of denotational semantics is that it should be *compositional*, i.e, the denotation of a program has to be constructed by the ones of its sub-expressions. This would allow one to reason inductively on the program structure, providing a general methodology for proving properties of programs.

Despite its expressiveness, the approach is less adequate in modelling the semantics of *interactive systems*. Indeed, in this case the non-deterministic behaviour of a program is more important than the function it computes: thus, these systems can not be simply denoted as if they were input-output functions. Many efforts have been devoted to equip concurrent programming languages with a compositional denotational semantics, most often by restricting at first the analysis to simple computational models exhibiting fundamental aspects of concurrent computations.

Denotations for process calculi. Among the proposed formalisms for the specification of interactive systems, *process calculi* proved to be one of the most successful: a set of basic operators defines the syntax of the calculus (formally, the set of operators forms a signature and the expressions of the language are the elements of the initial algebra associated to such a signature) and for each operator there is a set of rules (often in the so-called sos style [28]) describing the (operational) behaviour of a composite expression (a *process*) in terms of the possible interactions among its sub-expressions. The resulting semantics is usually defined by means of a *labelled transition system* (shortly, LTS): an indexed binary relation among a set of states, where each state represents a temporary configuration of a system, and the labelled relation abstractly carries some information on the intended behaviour of the system at each state.

Among the most widely adopted categorical tools for the denotation of process calculi we find *coalgebras* [29], recasting in an abstract setting the notion of transition system. An endo-functor B on *Set* (the category of sets and functions) allows for building the category of B-coalgebras and B-cohomomorphisms, which capture a class of LTSs satisfying suitable conditions and, most importantly, the associated "zig-zag" morphisms (i.e., morphisms that both preserve and reflect transitions). This category has a final object F, i.e., from every LTS there is a unique morphism to F. In this setting, one can easily define the denotation of an LTS as its image through this unique morphism. Similarly, also the syntax may be abstractly captured by an

endo-functor Σ which represents the operators of a given signature: the syntax is obtained as the initial object in the associated category of algebras and their homomorphisms.

Algebraic accounts for names. Recently, the (co)algebraic framework has been shown general enough to handle, by simply modifying either the base category or the associated endo-functors, so-called *nominal* calculi: sophisticated process calculi with complex mechanisms for variable binding, like the π-calculus [25]. More precisely, in order to represent both syntax and semantics, one has to consider endo-functors on categories Set^C of *covariant presheaves* over some C. These are functors from a so-called *index* category C of interfaces and contexts to *Set*. Intuitively, a presheaf maps each object i of C to the set of states having i as interface, and each arrow $c : i \to j$ to a function turning states with interface i into states with interface j. Such a syntax was e.g. tackled as a category of endo-functors Σ over $Set^{\mathbb{F}}$ (see [12] and [19]), for $\mathbb{F}$ the category of finite cardinals (i.e., sets of variables) and functions (i.e., substitutions).

One of the most widely investigated cases is that of the index category $\mathbb{I}$, that is, the full sub-category of $\mathbb{F}$ containing only injective morphisms. Both the early and late semantics of the π-calculus [25] can be characterized by endo-functors on $Set^{\mathbb{I}}$ [15, 16, 32]. Intuitively, any object i of $\mathbb{I}$ is mapped to a set of processes whose free names belong to i, and an arrow is mapped into an injective renaming: the set of names available for a process can always be enlarged (an operation that corresponds to allocate new names) but never contracted (so that two names can never be coalesced).

From substitutions to equivalence classes of names. In the proposals mentioned above, arrows of the index category are basically renamings, and thus names cannot be fused. In order to tackle non injective name substitutions, or more generally for capturing more complex relations among names, either the index category has to be enriched, or more esoteric monads should be taken into account for the syntax functor. For example, the open semantics of the π-calculus [30] can be recast using an endo-functor on $Set^{\mathbb{D}}$, for $\mathbb{D}$ the category of irreflexive relations and relation-preserving morphisms [18, 24]. Roughly, every object is a set of names equipped with an irreflexive relation such that two related names are considered to be distinguished. Morphisms are then functions between sets of names that preserve that irreflexive relation, and as such they cannot coalesce names that are considered distinct.

An alternative model for the open π-calculus (without restriction, hence with no need for distinctions) has been recently proposed in [33]. Here the index category is the standard $\mathbb{F}$, but the behavioural endo-functor (namely, the "free conditional join semi-lattice" monad) is quite novel with respect to those explored in literature. In a similar spirit [23], the category $\mathbb{F}$ is exploited for modelling the *fusion calculus* [27].

Our work follows the first thread of research, and it introduces the category $\mathbb{E}$ of *equivalence relations* in order to properly model the *explicit fusion calculus* [37]. The distinctive feature of such calculus is that name equivalences are themselves processes, thus representing some kind of process storage. These equivalences may influence the operational behaviour of the other processes they run in parallel with, since they allow to use interchangeably the names they equate. Each object of $\mathbb{E}$ is an equivalence relation over a set of names. Analogously to $\mathbb{I}$, morphisms preserve names (i.e., names can not be junked away), but equivalence classes can be enlarged,

thus obtaining a semantic counterpart of name fusions without losing any syntactic name. We prove that $\mathbb{E}$ is adequate for providing syntax and semantics to the explicit fusion calculus by giving suitable, yet ultimately quite standard endo-functors Σ and B on $Set^{\mathbb{E}}$.

Compositionality through saturation. Unfortunately, the ordinary coalgebraic approach does not naively apply to the explicit fusion calculus, because its operational semantics cannot be represented as a coalgebra in $Set^{\mathbb{E}}$. This can be more concretely understood by observing that the operational behaviour of processes is not compositional with respect to the semantic name fusions required by the arrows of $\mathbb{E}$.

Saturated semantics has been introduced in [2, 26] as a systematic approach to make compositional the semantics of interactive systems. In this paper, we show that the saturated semantics of the explicit fusion calculus can be modelled as a coalgebra in $Set^{\mathbb{E}}$ and that the equivalence (resulting from the theory of coalgebras) coincides with the standard inside-outside bisimilarity of [37].

It is noteworthy that the categorical machinery that we use for characterizing saturated semantics is exactly the one of [14, 16]. More explicitly, saturated semantics provides a concrete explanation of the abstract construction presented there.

Moving beyond explicit fusions, $\mathbb{E}$ is a first step towards a better understanding of even more complex calculi with an explicit handling of names, such as CC-PI [8]. Moreover, a slight modification of $\mathbb{E}$ (easily obtained, given its definition as a comma category) results in a category of equivalence relations and distinctions that might be suitable for the open π-calculus [30] and D-fusion [6]. Finally, these formalisms share a symbolic semantics that cannot be naively tackled through coalgebras. Giving a presheaf semantics for them should allow these symbolic semantics to be characterized through *normalized coalgebras* as shown in [3, 4].

Synopsis. In Section 2 we briefly overview the explicit fusion calculus and provide some results on the normal forms for processes. In Section 3 we introduce the category $\mathbb{E}$ of equivalence relations. In Section 4 we show how to use $Set^{\mathbb{E}}$ to characterise the abstract syntax of the explicit fusion calculus as an initial algebra. In Section 5 we study the semantics of the calculus, provide a behavioural endo-functor B in $Set^{|\mathbb{E}|}$, and explain how to recover a semantics in $Set^{\mathbb{E}}$ by (a categorical version of) saturation. Finally, in Section 6 we draw some conclusions, discuss some related work, and provide directions for future research.

Previous works. This work builds on [1], strengthening the presentation of the various sections and adding the novel characterisation result about saturated semantics for the explicit fusion calculus (thus, Section 5 is largely new).

2 Background on the Explicit Fusion Calculus

The *explicit fusion calculus* is a variant of the π-calculus that aims at guaranteeing asynchronous broadcasting of name equivalences to the environment. In order to ease its introduction, this section presents the explicit fusion calculus in the standard π-calculus fashion rather than in the "commitment" style of [37].

2.1 Syntax

Let $\mathcal{N}$ be a set of *names*, ranged over by $x, y, \ldots$; and let $\Pi = \{\tau\} \uplus \{x\langle y\rangle, \overline{x}\langle y\rangle \mid x, y \in \mathcal{N}\}$ be a set of *prefixes*, ranged over by π. The *explicit fusion processes*, ranged over by $P, Q, \ldots$, are defined by the syntax in Fig. 1a. The τ prefix stands for a silent action, while the complementary output $\overline{x}\langle y\rangle$ and input $x\langle y\rangle$ prefixes are used for communications. Unlike the π-calculus, the input prefix is not a binder, hence input and output operations are fully symmetric. As usual, $\mathbf{0}$ stands for the inert process, $P \mid P$ for the parallel composition, $!P$ for the replication operator and $(x)P$ for the process that makes the name x local in P. An *explicit fusion* $x = y$ is a process that exists concurrently with the rest of the system and that enables to use the names x and y in a completely interchangeable fashion.

$$\pi ::= \tau \mid \overline{x}\langle y\rangle \mid x\langle y\rangle \qquad P ::= \mathbf{0} \mid x = y \mid \pi.P \mid P \mid P \mid (x)P \mid !P$$

(a) syntax

$$P \mid \mathbf{0} \equiv_1 P \qquad P \mid Q \equiv_1 Q \mid P \qquad (P \mid Q) \mid R \equiv_1 P \mid (Q \mid R)$$
$$!P \equiv_1 P \mid !P \qquad (x)(y)P \equiv_1 (y)(x)P \qquad P \mid (x)Q \equiv_1 (x)(P \mid Q) \quad \text{if } x \notin \mathrm{fn}(P)$$

(b) structural congruence, I

$$x = y \mid (P \mid Q) \equiv_2 (x = y \mid P) \mid Q \equiv_2 P \mid (x = y \mid Q) \qquad x = y \mid P \equiv_2 P \mid x = y$$
$$x = y \mid \pi.P \equiv_2 x = y \mid \pi.(x = y \mid P) \qquad x = x \mid P \equiv_2 P$$
$$x = y \mid (z)P \equiv_2 (z)(x = y \mid P) \quad \text{if } z \notin \{x, y\} \qquad x = y \equiv_2 y = x$$
$$x = y \mid !P \equiv_2 !(x = y \mid P) \qquad x = y \mid y = z \equiv_2 x = z \mid y = z$$

(c) structural congruence, II

$$\mathtt{MEq}(\mathbf{0}) = \emptyset \qquad \text{the empty relation}$$
$$\mathtt{MEq}(\pi.P) = \mathsf{Id}_{\mathrm{fn}(\pi.P)} \qquad \text{the identity relation over } \mathrm{fn}(\pi.P)$$
$$\mathtt{MEq}(x = y) = \{(x, y), (y, x), (x, x), (y, y)\} \qquad \text{smallest equivalence including } (x, y)$$
$$\mathtt{MEq}(P \mid Q) = (\mathtt{MEq}(P) \cup \mathtt{MEq}(Q))^* \qquad \text{symmetric and transitive closure of the union}$$
$$\mathtt{MEq}((x)\,P) = \mathtt{MEq}(P) \setminus \{(y, z) \mid x \in \{y, z\}\} \qquad \text{removing name from equivalence classes}$$
$$\mathtt{MEq}(!\,P) = \mathtt{MEq}(P) \qquad \text{removing replication operator}$$
$$\mathtt{Eq}(P) = \mathtt{MEq}(P) \cup \mathsf{Id}_{\mathcal{N}} \qquad \text{adding the identity relation over all names}$$

(d) equivalence relation $\mathtt{Eq}(P)$

$$(\textsc{pref})\ \pi.P \xrightarrow{\pi} P \qquad (\textsc{comm})\ \frac{P \xrightarrow{x\langle y\rangle} P' \quad Q \xrightarrow{\overline{x}\langle w\rangle} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q' \mid y = w} \qquad (\textsc{par})\ \frac{P \xrightarrow{\lambda} P' \quad \mathrm{bn}(\lambda) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\lambda} P' \mid Q}$$

$$(\textsc{res})\ \frac{P \xrightarrow{\lambda} P' \quad x \notin \mathrm{n}(\lambda)}{(x)\,P \xrightarrow{\lambda} (x)\,P'} \qquad (\textsc{open-i})\ \frac{P \xrightarrow{z\langle x\rangle} P' \quad (x, z) \notin \mathtt{Eq}(P)}{(x)\,P \xrightarrow{z(x)} P'}$$

$$(\textsc{fus})\ \frac{P \xrightarrow{\lambda} Q \quad \mathtt{Eq}(P) \vdash \lambda = \lambda'}{P \xrightarrow{\lambda'} Q} \qquad (\textsc{struct})\ \frac{P \equiv P' \xrightarrow{\lambda} Q' \equiv Q}{P \xrightarrow{\lambda} Q}$$

(e) operational semantics (omitting rule (OPEN-O) for output)

Fig. 1 The explicit fusion calculus

Name substitution $P\sigma$ for a renaming σ is defined as the usual notion of substitution in the presence of binding, where bound names are introduced only by the operator $(x)P$. We define the *structural* congruences $\equiv_1$ and $\equiv_2$ as the least congruences over processes closed with respect to α-conversion and satisfying the axioms in Fig. 1b and c, respectively; and we define the structural congruence $\equiv$ as the transitive closure $(\equiv_1 \cup \equiv_2)^*$.[1] With respect to the standard structural congruence of the calculus, we eliminated the *subtraction* axiom, that equates $(\nu x)(x = y)$ and $\mathbf{0}$, thus allowing to remove names from an equivalence relation via restriction: we prefer to add an α-conversion law for our processes, which is apparently the reason underlying the introduction of the subtraction axiom in [37]. We also dropped the axioms for interchanging prefixes whose names are fused, mimicked by the introduction of the additional operational rule (FUS), as shown below. Finally, we weakened the *reflexivity* axiom, equating $x = x$ and $\mathbf{0}$: we consider a process-preserving version $x = x \mid P \equiv_2 P$.

Summing up, the chosen set of axioms for $\equiv_2$ guarantees that any parallel composition of fusions is closed with respect to the composition with any other fusion occurring in the induced equivalence relation, and it also ensures a restricted version of what is called small-step substitution in Fig. 2 of [37]. The equivalence relation $\mathtt{Eq}(P)$ characterising the name equivalences induced by a process P is defined in Fig. 1d. Formally, we first define a minimal equivalence relation $\mathtt{MEq}(P)$ containing the transitive closure of the name equivalences occurring in P and the identities over the free names of P, $\mathrm{fn}(P)$. $\mathtt{Eq}(P)$ is obtained by adding to $\mathtt{MEq}(P)$ the set of identities over every name that is not in $\mathrm{fn}(P)$. Proposition 1 below uses relation $\mathtt{MEq}(P)$ in order to make explicit the effect of $\equiv_2$: it generalizes [37, Lemma 5].

Proposition 1 (decomposition) *Let P, Q be processes such that $P = C[Q]$ for a unary context $C[-]$ that does not bind the names in $\mathrm{fn}(P)$ (i.e., such that the placeholder $-$ does not fall in the scope of a restriction operator (x), for any $x \in \mathrm{fn}(P)$). Moreover, let E_P be the process*

$$\prod_{\{(x,y)\in \mathtt{MEq}(P)\}} x = y$$

where $\prod$ denotes the multiple parallel composition. Then, $P \equiv_2 C[E_P \mid Q]$.

Additionally, we may provide explicit fusion processes with a suitable normal form, which is needed later on for proving the initiality of our presheaf syntax. At first, we define as *pure fusions* those processes containing only the parallel composition of fusions $x = y$; and we denote as *grounded* those processes that are not pure fusions and such that no pure fusion occurs as the argument of a unary operator.

Proposition 2 (normal form) *Let P be a process. Then, either it is a pure fusion, and it is equivalent (according to $\equiv_2$) to the process E_P; or it is equivalent (according to $\equiv_2$) to a process $E_P \mid P'$, such that $\mathtt{MEq}(P') = \mathit{Id}_{\mathrm{fn}(P')}$ and all fusions in P' occur under the scope of a unary operator. Moreover, if P is grounded, so is P'.*

[1] The axioms are slightly redundant, yet $\equiv_2$ is going to be used on its own later in the paper.

This means that all fusions involving free names are brought to the top: since identities $y = y$ are immaterial, any other fusion occurs either under a prefix or it involves at least a bound name x and it occurs below a restriction operator (x). Indeed, for finite processes (i.e., containing no occurrence of the replication operator) a stronger normal form can be easily found. Albeit not pivotal for our work, it is stated below.

Proposition 3 (normal form) *Let P be a finite process. Then, either it is a pure fusion, or it can be uniquely decomposed (according to $\equiv_1$) as a process of the shape*

$$E_P \mid (x_1) \dots (x_m) \left[\prod_{i=1\dots m} x_i = y_i \mid \prod_{j=1\dots n} \pi_j.P_j \right]$$

such that all x_k's are pairwise disjoint and $y_i \in \mathrm{fn}(P) \uplus \{x_1 \dots x_m\}$ for $i = 1 \dots m$. Moreover, if P is grounded, so are all P_j's.

In the rest of the paper, we focus our attention to grounded processes and, unless otherwise specified, a process is always a grounded one. This is far from restrictive, since each process can be turned into an equivalent, grounded one simply inserting any pure fusion into the context $- \mid \mathbf{0}$.

2.2 Operational Semantics

The rules of the operational semantics, as depicted in Fig. 1e, recall the rules of the π-calculus. We assume a set of labels

$$\mathcal{L} = \{\tau, z\langle y\rangle, \overline{z}\langle y\rangle, z(w), \overline{z}(w)\}$$

where z, y are free and w is bound. We let $\lambda, \lambda_1, \dots$ range over $\mathcal{L}$. By $\mathtt{Eq}(P) \vdash \lambda_1 = \lambda_2$ we mean that P contains enough fusions to interchange λ_1 and λ_2. Formally, we have

$$\begin{array}{lll}
\mathtt{Eq}(P) \vdash \tau = \tau & & \\
\mathtt{Eq}(P) \vdash z\langle y\rangle = x\langle w\rangle & \text{if } (z,x),(y,w) \in \mathtt{Eq}(P) & \text{(similarly for output)} \\
\mathtt{Eq}(P) \vdash z(v) = x(v) & \text{if } (z,x) \in \mathtt{Eq}(P) & \text{(similarly for output)}
\end{array}$$

Unlike the π-calculus, the synchronization of two complementary processes $x\langle y\rangle.P$ and $\overline{x}\langle z\rangle.Q$ yields an explicit fusion $y = z$ rather than binding y to z (rule (COM)). According to rule (FUS), a process can undergo any transition up to interchanging names that are fused. For instance, a process $x = y \mid \overline{x}\langle z\rangle.P$ can make both actions $\overline{x}\langle z\rangle$ and $\overline{y}\langle z\rangle$. As another example, consider the process $P = z\langle x\rangle.\overline{x}\langle v\rangle.Q \mid \overline{z}\langle y\rangle.y\langle w\rangle.R$. P can undergo the following sequence of transitions

$$P \xrightarrow{\tau} x = y \mid \overline{x}\langle v\rangle.Q \mid y\langle w\rangle.R \xrightarrow{\tau} x = y \mid v = w \mid Q \mid R$$

The first step is due to rule (COM), while the second one follows from applying rule (FUS) to $x = y \mid \overline{x}\langle v\rangle.Q$

$$x = y \mid \overline{x}\langle v\rangle.Q \xrightarrow{\overline{y}\langle v\rangle} x = y \mid Q$$

and, then, rule (COM) to $x = y \mid \overline{x}\langle v\rangle.Q$ and $y\langle w\rangle.R$.

In [36] several bisimulations were proposed for the explicit fusion calculus: they were all proved to coincide, and to be a congruence. For convenience, as a reference semantics we consider the *inside-outside* bisimulation, similar to π-calculus open one.

Definition 1 (inside-outside bisimulation) Let R be a symmetric relation on fusion processes. We say that R is a bisimulation if whenever $P\ R\ Q$ holds then

1. $\mathtt{Eq}(P) = \mathtt{Eq}(Q)$;
2. If $P \xrightarrow{\lambda} P'$ with $\mathrm{bn}(\lambda) \cap \mathrm{fn}(Q) = \emptyset$ then $Q \xrightarrow{\lambda} Q'$ and $P'\ R\ Q'$;
3. $P \mid x = y\ R\ Q \mid x = y$, for all fusions $x = y$.

We let $\sim^{io}$ denote the largest bisimulation, the *inside-outside bisimilarity*.

According to clause *1*, processes $x = y \mid \mathbf{0}$ and $\mathbf{0}$ are not bisimilar: $\mathtt{Eq}(x = y \mid \mathbf{0})$ and $\mathtt{Eq}(\mathbf{0})$ differ. In fact, clause *1* is necessary to guarantee that inside-outside bisimilarity be a congruence. For instance, $x = y \mid \mathbf{0}$ and $\mathbf{0}$ can make no transition but in the context $\_ \mid \overline{x}\langle v\rangle \mid y\langle u\rangle$ the former process can make a τ-action while the other one is stuck.

On the other side, clause *3* allows to distinguish the following processes (adapted from an example proposed in [7] for the π-calculus)

$$P = !\overline{y}\langle\rangle.x\langle\rangle.\tau.z\langle\rangle \mid !x\langle\rangle.\overline{y}\langle\rangle.\tau.z\langle\rangle \qquad Q = !(w)(\overline{y}\langle\rangle.\overline{w}\langle\rangle \mid x\langle\rangle.w\langle\rangle.z\langle\rangle)$$

When inserted into the context $\_ \mid x = y$, Q can perform an action $\xrightarrow{z\langle\rangle}$ after two steps (synchronizing $\overline{y}\langle\rangle$ with $x\langle\rangle$), while P needs at least three steps to do the same.

3 A Category of Name Equivalences

The paper aims at extending the presheaf approach in order to tackle the explicit fusion calculus. To this end, we define a category of equivalence relations $\mathbb{E}$ that is used to represent (the sets of) processes as a presheaf in $Set^{\mathbb{E}}$. The objects of $\mathbb{E}$ are surjective functions on (finite) sets and the arrows of $\mathbb{E}$ are suitable injective functions.

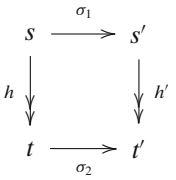
So, let us first define $\mathbb{F}$ as the category of finite sets and functions.

Definition 2 (The category $e\mathbb{E}$ of equivalence relations) The category $e\mathbb{E}$ of equivalence relations has as objects the pairs $\langle s, \sim_s\rangle$ such that $s \in \mathbb{F}$ and $\sim_s \subseteq s \times s$ is an equivalence relation, and as arrows $\sigma : \langle s, \sim_s\rangle \to \langle s', \sim_{s'}\rangle$ those functions in $\mathbb{F}$ from s to s' preserving the equivalence relation.

The category has all limits and colimits, whose first component is computed pointwise. The monomorphisms are precisely those induced by the injective functions in $\mathbb{F}$; while the regular monomorphisms are those that reflect the equivalence relation, i.e., such that $\sigma(a) \sim_{s'} \sigma(b)$ implies $a \sim_s b$ for all $a, b \in s$.

In the following, we are going to consider the sub-category $\mathbb{E}$ of equivalence relations and monomorphisms. However, we are going to adopt a slightly more concrete definition, starting from a sub-category of the comma category $ID_{\mathbb{F}} \downarrow ID_{\mathbb{F}}$.

The category $ce\mathbb{E}$ of concrete equivalence relations has as objects the surjective functions $h : s \twoheadrightarrow t$ in $\mathbb{F}$ and as arrows $\sigma : h \to h'$ those pairs $\langle \sigma_1, \sigma_2 \rangle$ of functions in $\mathbb{F}$, such that the diagram below commutes.



The domain s of the function h specifies a set of names while the codomain represents a set of equivalence classes involving the names of s. Note that the equivalence classes of the names in s, as represented in t and carried along σ_1 to s', may be either left unchanged or further merged in t', but they can not be broken.

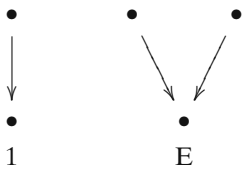
The monomorphisms of $ce\mathbb{E}$ are precisely those arrows $\langle \sigma_1, \sigma_2 \rangle$ such that σ_1 is injective; the regular monomorphisms, those arrows such that σ_2 injective as well. We denote $c\mathbb{E}$ the sub-category of monomorphisms of $ce\mathbb{E}$.

The category $\mathbb{E}$ ($e\mathbb{E}$) is the quotient of $c\mathbb{E}$ (of $ce\mathbb{E}$, respectively) after identifying those arrows $\langle id_s, \sigma_2 \rangle$ such that σ_2 is an isomorphism. Note that the source and the target objects of such arrows must also be identified, hence the objects of the quotient categories are just pairs $\langle s, \sim_s \rangle$. The quotient is well-given: composition is defined since for any two representatives $h : s \twoheadrightarrow t$ and $h' : s \twoheadrightarrow t'$ of the equivalence class there is exactly one arrow shaped $\langle id_s, \sigma_2 \rangle$ between them (due to the surjectivity of h and h').

We are thus going to concretely reason on $\mathbb{E}$ as the quotient category of $c\mathbb{E}$. With an abuse of notation, we will refer implicitly to objects and arrows of $c\mathbb{E}$ as the representatives of their isomorphism classes.

We let $\mathbb{E}(h, h')$ denote the set of arrows in $\mathbb{E}$ with source h and target h'. Each object h of $\mathbb{E}$ defines a functor $\mathbb{E}(h, \_) : \mathbb{E} \to \mathit{Set}$: an object h' of $\mathbb{E}$ is mapped into the set $\mathbb{E}(h, h')$, and an arrow $\sigma : h' \to h''$ of $\mathbb{E}$ into the function $\mathbb{E}(h, \sigma) : \mathbb{E}(h, h') \to \mathbb{E}(h, h'')$, defined by post-composition: for any $\rho \in \mathbb{E}(h, h')$, $\mathbb{E}(h, \sigma)(\rho) = \rho; \sigma$.

Let $\{\star\}$ denote the one element set, graphically represented as $\bullet$, and moreover let 1 and E denote the following two objects of $\mathbb{E}$ (corresponding to $id_{\{\star\}}$ and $[id_{\{\star\}}, id_{\{\star\}}]$, respectively, for the arrow uniquely induced by the coproduct in $\mathbb{F}$)



Now, $\mathbb{E}(1, h)$ is the set of all monomorphisms that map $\star$ to an element of the domain of h: hence, it is isomorphic to it. Similarly, $\mathbb{E}(E, h)$ abstractly represents the set of explicit fusions $x = y$ (for x, y different names) that hold in h. Hereafter, we denote the functor $\mathbb{E}(1, \_)$ by *Names* and the functor $\mathbb{E}(E, \_)$ by *Fus*.

4 Abstract Syntax

In this section we consider the category $Set^{\mathbb{E}}$ of functors from $\mathbb{E}$ to Set (called *presheaves* over $\mathbb{E}^{op}$) and natural transformations. This category can be used for both the syntax and the semantics of the explicit fusion calculus.

With respect to the syntax, objects of the *index* category $\mathbb{E}$ can be actually viewed as *types*, since they denote the equivalence classes defined over the names of a process. More precisely, we may say that the presheaf for the syntax associates to each index $h : s \twoheadrightarrow t$ the set of processes that can be typed with h.

In Section 4.1 we provide an account of the concrete syntax of the explicit fusion calculus as a presheaf, exploiting the previously defined equivalence relation $\mathtt{ME}_{\mathtt{Q}}$. Then, in Section 4.3 we explain how this syntax can be described as an initial algebra in $Set^{\mathbb{E}}$ for a suitable endo-functor, preserving types. For the purpose, in Section 4.2, we introduce a number of basic constructions that are used as a meta-language in $Set^{\mathbb{E}}$, and are distinguishing features of this category.

4.1 Typing the Concrete Syntax

For the definition of the syntactic presheaf, for each arrow $h : s \twoheadrightarrow t$ we introduce the notation Ker_h, representing the equivalence relation induced by h (obviously defined), and the pure fusion E_h, containing all the pairs in an equivalence relation h, viewed as fusions (analogous to the process E_P induced by $\mathtt{ME}_{\mathtt{Q}}(P)$)

$$E_h = \prod_{\{x,y \in s \mid h(x)=h(y)\}} x = y.$$

We can now introduce the functor $Syn : \mathbb{E} \to Set$, defined on objects as

$$Syn(h : s \twoheadrightarrow t) = \{P \mid \mathtt{ME}_{\mathtt{Q}}(P) \cup \mathsf{Id}_s = Ker_h\} / \equiv_2 .$$

The intuition is that if an agent P is in $Syn(h : s \twoheadrightarrow t)$, then its free names are elements of s, and its equivalence classes $\mathtt{ME}_{\mathtt{Q}}(P)$ are exactly those described by h, modulo adding the pairs (x, x) for $x \notin \mathrm{fn}(P)$.

The action of the functor on arrows must injectively relabel processes, whilst it syntactically adds as many fusions as needed (actually, all of them) to put a process in the equivalence class chosen as destination

$$Syn\left(\langle \sigma_1 : s \hookrightarrow s', \sigma_2 : t \to t' \rangle : h \to h'\right)(P) = P\sigma_1 \mid E_{h'}$$

This definition is tailored to the interpretation of the elements at each stage h, in the presheaf for the syntax, as those processes where exactly the fusions of h hold. This is necessary to faithfully reflect the notion of semantic equivalence of the explicit fusion calculus, namely inside-outside bisimilarity: this is going to be made precise in the following sections.

4.2 A Basic Metalanguage in $Set^{\mathbb{E}}$

Besides the usual constructors for the polynomial endo-functors (namely constants, finite sums and products), $Set^{\mathbb{E}}$ actually allows for a few additional constructors that are well-suited for handling fusions.

Bottom operator $F^{\perp}$. For any $s \in Set$, let $s_{\perp}$ be $s + \{\star\}$ and for any $h : s \twoheadrightarrow t$, let $h_{\perp} : s_{\perp} \twoheadrightarrow t_{\perp}$ be the arrow $[h, id_{\{\star\}}]$. The bottom operator $F^{\perp}$ is an endo-functor on Set^{E} defined as $F^{\perp}(h) = F(h_{\perp})$ and $F^{\perp}(\langle\sigma_1, \sigma_2\rangle) = F(\langle(\sigma_1)_{\perp}, (\sigma_2)_{\perp}\rangle)$.

Box operator $F^{\Box}$. The *box operator* $F^{\Box}$ is the endo-functor on $Set^{\mathbb{E}}$ defined on objects and arrows as below. Let $h : s \twoheadrightarrow t$ be an object of $\mathbb{E}$. Then

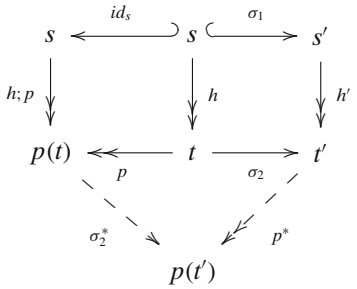
$$F^{\Box}(h) = \sum_{p} F(h; p)$$

where $p : t \twoheadrightarrow p(t)$ is a (identity preserving) morphism from t to a partition $p(t)$ of t. Choosing $p(t)$ as a target of p simply amounts to fix a canonical representative for each isomorphic merging of names, even if its identity is actually immaterial in $\mathbb{E}$.

Let $h : s \twoheadrightarrow t$ and $h' : s' \twoheadrightarrow t'$ be objects of $\mathbb{E}$ and $\sigma = \langle\sigma_1, \sigma_2\rangle : h \to h'$ be an arrow of $\mathbb{E}$. Then

$$F^{\Box}(\sigma) = \sum_{p} F(\langle\sigma_1, \sigma_2^*\rangle)$$

where σ_2^* is uniquely induced by the pushout depicted in the diagram below for each p, noting that $\langle\sigma_1, \sigma_2^*\rangle : h; p \to h'; p^*$.



Roughly, $F^{\Box}$ may add any possible name equivalence to an equivalence relation on s. The box operator is used in Section 4.3 to define the functor for prefixes of the explicit fusion calculus. Indeed, a fusion process with a prefix $\pi.P$ has no equivalence relation on its names, while P may have any equivalence.

Shift operator F^{δ}. The *shift operator* is the functor defined on objects and arrows as follows. Let $h : s \twoheadrightarrow t$ be an object of $\mathbb{E}$. Then

$$F^{\delta}(h) = F^{\perp}(h) + \sum_{e} F([h, e])$$

where $e : \{\star\} \to t$ is a function mapping $\star$ in some element of t and $[h, e] : s + \{\star\} \twoheadrightarrow t$ is the function uniquely induced by the coproduct, mapping all elements of s in $h(s)$ and $\star$ in $e(\star)$.

Let $h : s \twoheadrightarrow t$ and $h' : s' \twoheadrightarrow t'$ be objects of $\mathbb{E}$ and $\sigma = \langle\sigma_1, \sigma_2\rangle : h \to h'$ be an arrow of $\mathbb{E}$. Then

$$F^{\delta}(\sigma) = F^{\perp}(\langle\sigma_1, \sigma_2\rangle) + \sum_{e} F(\langle(\sigma_1)_{\perp}, \sigma_2\rangle)$$

noting that $e; \sigma_2 : \{\star\} \to t'$ and $\langle (\sigma_1)_\perp, \sigma_2 \rangle : [h, e] \to [h', e; \sigma_2]$.

The shift operator is used to define the functor for the restriction operation in the explicit fusion calculus. In fact, F^δ is a variant of the functor for name generation used to model restriction in the π-calculus. The main point here is that objects are now equivalence relations. Hence, when generating a new name x, it is necessary to specify whether x is equivalent to any other name already occurring in the process, or it belongs to its own equivalence class. It is worth to note that the new name belongs at most to one of the old equivalence classes. As an example, consider the process $(x)(x = y \mid x = z \mid P)$. It exists only in those stages where y and z are already fused. Indeed, according to $\equiv_2$ the process would be equivalent to $y = z \mid (x)(x = y \mid P)$.

4.3 The Syntax as an Initial Algebra

An abstract syntax for the explicit fusion calculus is captured by the initial algebra of the endo-functor $\Sigma : Set^{\mathbb{E}} \to Set^{\mathbb{E}}$ that is defined below.

$$
\begin{array}{rcll}
\Sigma F & = & 1 & \text{(inert process)} \\
 & + & F & \text{(replication)} \\
 & + & F \times F & \text{(parallel composition)} \\
 & + & F^{\Box} & \text{(tau prefix)} \\
 & + & Names \times Names \times F^{\Box} & \text{(input prefix)} \\
 & + & Names \times Names \times F^{\Box} & \text{(output prefix)} \\
 & + & F^{\delta} & \text{(restriction)}
\end{array}
$$

We briefly comment on the component functors. For the purpose, we introduce the notation $!_s$ representing the unique morphism of type $s \to \{\star\}$ from s to the final object in *Set*, viewed as an object of $\mathbb{E}$ (noting that $!_{\{\star\}} = id_{\{\star\}}$). Moreover, we say that an object $h : s \twoheadrightarrow t$ of $\mathbb{E}$ is *included* in an object $h_1 : s_1 \twoheadrightarrow t_1$, and write $h \sqsubseteq h_1$, if there exists an arrow $\sigma : h \to h_1$ (hence $Ker_h \subseteq Ker_{h_1}$).

The functor for the inert process just returns a singleton at each stage; whilst the functor for replication just returns an additional copy of the processes at each stage, as if prefixing them with a suitable operator.

The three functors for prefixing are similar, the idea being of "hiding" equivalences of a process P when prefixed as $\pi.P$, similarly to what is done in the definition of $\mathtt{MEq}(P)$. To this purpose, we use the functor $F^{\Box}$: it returns, at stage $h : s \twoheadrightarrow t$, all the processes occurring in $F(h')$ for all h' such that $h \sqsubseteq h' \sqsubseteq !_s$.

The parallel composition of two elements $a \in F(h_1)$ and $b \in F(h_2)$ can only be found at a stage h such that $h_1 \sqsubseteq h$ and $h_2 \sqsubseteq h$. Indeed, being a presheaf, the product $F \times F$ is a functor of type $\mathbb{E} \to Set$ defined as $(F \times F)(h) = F(h) \times F(h)$. Hence, we can not find the product of two elements h_1 and h_2 belonging to *different* stages, but we have to "inject" them into a common stage h, where both can be found. Note that, since arrows can never remove fusions from objects of $\mathbb{E}$, this requirement is equivalent to stating that $\mathtt{MEq}(P \mid Q) = (\mathtt{MEq}(P) \cup \mathtt{MEq}(Q))^*$, carrying on the intuition that the stage of an element always includes its fusions.

Next, we consider F^δ. We can exemplify its meaning using the process $(x)(x = y \mid \mathbf{0})$. This process belongs to $F^\delta(id_{\{y\}})$, since its set of free names is indeed included in $\{y\}$. However, it is obtained from the process $x = y \mid \mathbf{0}$, which does not belong to $F(id_{\{x,y\}})$, but rather to $F(!_{\{x,y\}})$. Hence, to capture the process $(x)(x = y \mid \mathbf{0})$ in the

abstract syntax, $F^{\delta}(id_{\{y\}})$ includes $F(!_{\{y,\star\}})$, or in general, it permits to fuse the fresh name in the process underlying the restriction with any other existing name.

The previous discussion is summed up by the rules in Table 1. The different stages h account for the equivalences. For example, note that $\mathbf{0}$ belongs to $F(id_{\emptyset})$, for $id_{\emptyset}$ the identity of the empty set. It thus belongs also to $F(id_{\{x\}})$ and to $F(!_{\{x,y\}})$, for any name x, y. However, in the latter stage it actually represents the process $x = y \mid \mathbf{0}$.

4.4 Abstract vs Concrete Correspondence

Now we show that *Syn* and the initial algebra T_{Σ} of Σ are naturally isomorphic. The quotient with respect to $\equiv_2$ is aimed at the following proposition, which is the key to define the natural isomorphism.

Proposition 4 *For each $P \in Syn(h : s \to t)$ we have $P \equiv_2 P' \mid E_h$, where*

1. $\mathtt{ME}_{\mathtt{Q}}(P') = Id_{fn(P')}$ *and all fusions in P' occur under the scope of a unary operator;*
2. *if $P' = \pi.P''$ then $P'' \in Syn(h; p)$ for a unique p;*
3. *if $P' = (x)P''$ then $x \notin fn(P)$, and either $P'' \in Syn(h \cup \{x \mapsto *\} : s \cup \{x\} \twoheadrightarrow t + 1)$ or $P'' \in Syn(h')$ for a unique $h' : s \cup \{x\} \twoheadrightarrow t$ such that $(h')_{|_s} = h$.*

The result above is a rephrasing in terms of stages of the normal form provided in Proposition 2. However, it enables us to prove the correspondence between *Syn* and T_{Σ} in a simple and straightforward way.

Theorem 1 *The presheaves Syn and T_{Σ} are naturally isomorphic.*

Proof We use a single induction on the structure of terms in $Syn(h)$ for an arbitrary h, to give the definition of $i_h : Syn(h) \to (T_{\Sigma})_h$, and to prove that it is a natural monomorphism. Surjectivity is immediate from the definition, making i a natural isomorphism. Let $P \in Syn(h)$. We assume that P is in the form $P' \mid E_h$ respecting the equations in Proposition 4.

We now proceed by induction on the structure of P'. We can omit the case in which P' is an explicit fusion, due to Condition 1 in Proposition 4. To denote elements of a coproduct $\sum_{i \in I} X_i$ (including the functor Σ) we use pairs $\langle i \in I, x \in X_i \rangle$.

– If $P' = \mathbf{0}$, we let $i_h(P) = \langle 0, * \in 1_h \rangle$. Naturality of i when applied to P is obvious. By inspection of the other cases, we see that i_h restricted to processes in the form of P is injective, in the sense that there is no $R \neq P$ such that $i_h(R) = i_h(P)$. In the following, we denote with $PNI(P)$ this "partial" naturality and injectivity property, which is considered part of the inductive hypothesis.

Table 1 Inference rules for abstract syntax

$$\mathbf{0} \in F(id_{\emptyset}) \qquad \frac{p \in F(h) \quad h \sqsubseteq h_1}{p \in F(h_1)} \qquad \frac{p \in F(h)}{!p \in F(h)}$$

$$\frac{p \in F^{\square}(h)}{\tau.p \in F(h)} \qquad \frac{p \in F^{\delta}(h)}{(\star)p \in F(h)} \qquad \frac{p \in F(h) \quad q \in F(h)}{p \mid q \in F(h)}$$

$$\frac{a \in Names(h) \quad b \in Names(h) \quad p \in F^{\square}(h)}{\overline{a}\langle b \rangle.p \in F(h)}$$

$$\frac{a \in Names(h) \quad b \in Names(h) \quad p \in F^{\square}(h)}{a\langle b \rangle.p \in F(h)}$$

- If $P' = !P''$, then by Condition 1 we have $P'' \mid E_h \in Syn(h)$, therefore we let $i_h(P) = \langle 1, i_h(P'')\rangle$. $PNI(P)$ is again obtained by inspection of the other cases, and the inductive hypothesis.
- If $P' = P'' \mid P'''$, by definition of $\mathtt{ME}_\mathtt{Q}$, and Condition 1, we have $P'' \mid E_h$, $P''' \mid E_h \in Syn(h)$. We therefore have $i_h(P'') = Q''$, $i_h(P''') = Q'''$, and $PNI(P'')$, $PNI(P''')$. By letting $i_h(P) = \langle 2, \langle Q'', Q'''\rangle\rangle$, by the definition of substitution, and inspection of the other rules, we have $PNI(P)$.
- If $P' = \tau.P''$, by Condition 2, we have $P'' \in Syn(h; p)$, $i_{h;p}(P'') = Q$, therefore we let $i_h(P) = \langle 3, \langle p, Q\rangle\rangle$. By the inductive hypothesis, we have $PNI(P'')$, and it is easy to see that $i_h(P)$ satisfies injectivity. For naturality, we have $Syn(\langle\sigma_1, \sigma_2\rangle : h \to h')(P) = P\sigma_1 \mid E_{h'}$, and on the other hand the colimit in the definition of $\langle\sigma_1, \sigma_2\rangle^{\square}$ puts the resulting process in stage h', and substitutes free names as σ_1.
- The cases for the input and output prefixes are very similar to the one for the τ prefix and are omitted.
- If $P' = (x)P''$, we distinguish two cases, either $P'' \in Syn(h \cup \{x \mapsto *\})$ or $P'' \in Syn(h')$ according to Condition 3. In the first case, we have $P'' \equiv_2 P'''|P_{h\cup\{x\mapsto *\}}$ for some P''', by Proposition 4, and we observe that $P_{h\cup\{x\mapsto *\}} \equiv_2 P_h \mid (x = x)$. We let $i_h(P) = \langle 6, \langle 0, i_{h_\perp}(P''[^*/_x])\rangle\rangle$ where $*$ is the new element added to s in $h_\perp$. Injectivity comes from α-conversion in the definition of $\equiv_2$. Naturality comes from the definition of $F_\perp$, and from the inductive hypothesis. In the second case, we have $P'' \equiv_2 P'''|E_{h'}$ for some P''', and $i_{h'}(P'''|E_{h'}) = Q$. We let $i_h(P) = \langle 6, \langle 1, \langle h', Q[^*/x]\rangle\rangle\rangle$. Again we have PNI by the quotient with respect to α-conversion in Syn and by the inductive hypothesis on P'''. □

The restriction to grounded processes (see Section 2) is essential for the above theorem, since otherwise the process P' (in the proof) might not exist. For instance, grounded process $\tau.(x = y \mid \mathbf{0})$ belongs to T_Σ, while non-grounded $\tau.(x = y)$ does not.

5 Operational and Saturated Semantics For Explicit Fusions

In the (co)algebraic approach, given a syntactic endo-functor Σ and a behavioural endo-functor B, the operational semantics of a formalism can be seen as a B-coalgebra $T_\Sigma \to B(T_\Sigma)$ where T_Σ is the initial algebra of Σ. However, the application of the methodology is hard to be devised for calculi that involve non-injective name mapping.

Consider the explicit fusion calculus. Its operational semantics (as described in Section 2) is not a coalgebra in $Set^{\mathbb{E}}$, or more precisely it does not immediately translate to a natural transformation $\alpha : Syn \to B(Syn)$ for some functor B (where Syn is the syntactic presheaf of Section 4.1, or equivalently the abstract syntax from Section 4.3). To appreciate this, note that the transitions of the calculus are not reflected by the action of the morphisms of $\mathbb{E}$ (i.e., name fusions) in Syn: take $P = \overline{x}\langle z\rangle \mid y\langle w\rangle$ and a morphism σ in $\mathbb{E}$ fusing x and y. Then $Syn(\sigma)(P) = \overline{x}\langle z\rangle \mid y\langle w\rangle \mid x = y \xrightarrow{\tau}$, while $P \not\xrightarrow{\tau}$. For a more detailed description of this problem, the reader is referred to Appendix.

A way out of the *impasse* is provided by *saturated semantics* [2]. This technique was originally introduced for reactive systems [20], but it has been later applied to

many different settings. The idea is almost straightforward: given a LTS $\longrightarrow$ with labels in $\mathcal{L}$, a *saturated transition system* $\longrightarrow_S$ having the same states is defined, taking into account a set of *contexts* $\mathcal{C}$ (in our case, the arrows of $\mathbb{E}$). The saturated LTS is labelled by pairs $\sigma, \lambda \in \mathcal{C} \times \mathcal{L}$, and it contains all the possible transitions of the form $P \xrightarrow{\sigma,\lambda}_S Q$, for each transition $\sigma(P) \xrightarrow{\lambda} Q$ of the original LTS. As for presheaves, the application $\sigma(P)$ is the action of an arrow σ on P. In this setting, the saturated transition system forms a coalgebra in $Set^{\mathbb{E}}$ (i.e., a natural transformation). To see this formally, we render the above construction of saturated semantics as a categorical construction.

We proceed as follows. First we define the operational semantics of the calculus as a B-coalgebra for a proper endo-functor B on $Set^{|\mathbb{E}|}$, where $|\mathbb{E}|$ is the discrete category containing all the objects of $\mathbb{E}$. Then we introduce a pair of adjoint functors $U : Set^{\mathbb{E}} \to Set^{|\mathbb{E}|}$, $R : Set^{|\mathbb{E}|} \to Set^{\mathbb{E}}$ and we model the saturated semantics as a RBU-coalgebra on $Set^{\mathbb{E}}$. Finally, we show that the induced coalgebraic bisimilarity coincides with input-output bisimilarity (Definition 1).

5.1 Operational Semantics in $Set^{|\mathbb{E}|}$

The operational semantics of the calculus can be modelled as a B-coalgebra for a suitable endo-functor $B : Set^{|\mathbb{E}|} \to Set^{|\mathbb{E}|}$. This behavioural endo-functor B describes the typing of the transition system.

$$
\begin{array}{rcll}
BF & = & \mathcal{P}_f(& \\
 & & F^{\square} & \text{(tau action)} \\
 & + & Names \times Names \times F^{\square} & \text{(input action)} \\
 & + & Names \times Names \times F^{\square} & \text{(output action)} \\
 & + & Names \times F^{\perp\square} & \text{(bound input action)} \\
 & + & Names \times F^{\perp\square} & \text{(bound output action)} \\
 & &) &
\end{array}
$$

Note that we use the endo-functors $(-)^{\square}$ and $(-)^{\perp\square}$: they were defined over $Set^{\mathbb{E}}$, even if they obviously extend to endo-functors over $Set^{|\mathbb{E}|}$. We will now comment on the definition of B. First of all, note the definition of input. This is quite different from the standard works on presheaf semantics for π-calculus, such as [16], where an exponential type was used for the input action. In the explicit fusion calculus the input and output prefixes are completely symmetric and thus input is non-binding. For this reason, we can safely tackle it in the same way as output.

Moreover, the destination states for τ actions are in $F^{\square}$. This can be understood by noting that when a process performs a τ transition, then the destination has the same names of the starting state, yet possibly more fusions. As an example, consider the fusion process $\tau.(x = y \mid Q)$. Performing a τ transition, this process arrives into the state $x = y \mid Q$, where x and y are now identified. Analogously for input and output actions. Instead, in the case of bound input and bound output, the destination state has one more name and (possibly) more fusions. Therefore, it belongs to $F^{\perp\square}$.

A B-coalgebra is a pair $\langle F, \beta \rangle$, for F an object of $Set^{|\mathbb{E}|}$ (a functor from $|\mathbb{E}|$ to Set) and $\beta : F \to B(F)$ an arrow of $Set^{|\mathbb{E}|}$ (a natural transformation between F and $B(F)$). In other words, $\beta : F \to B(F)$ is a family of functions $(\beta)_h : F(h) \to B(F)(h)$ for all $h \in \mathbb{E}$. Now, let $\int F$ denote the set of the *elements* of a functor F, namely, the disjoint

union $\sum_{h\in\mathbb{E}} F(h)$. It is easy to note that for any functor F and $h \in \mathbb{E}$ the following inclusions $F^{\perp}(h) \subseteq \int F$, $F^{\Box}(h) \subseteq \int F$ and $F^{\delta}(h) \subseteq \int F$ hold, so by construction every B-coalgebra induces a transition relation whose state space is represented by $\int F$.

More concretely, B-coalgebras are transition systems as described in the following definition, where we denote the elements of a functor F as pairs $h \vdash f$, for $f \in F(h)$ and we write $h \vdash f \in F(h')$ whenever $F(h) \subseteq F(h')$.

Definition 3 (B-coalgebras) A B-coalgebra consists of a presheaf $F : |\mathbb{E}| \to \mathit{Set}$ and a transition relation $\longrightarrow$ such that

1. states are elements of $\int F$;
2. transitions are typed such that
   - if $h \vdash f \xrightarrow{\tau} h' \vdash f'$ then $h' \vdash f' \in F^{\Box}(h)$,
   - if $h \vdash f \xrightarrow{z\langle y\rangle} h' \vdash f'$ then $h' \vdash f' \in F^{\Box}(h)$ and $z, y \in \mathit{Names}(h)$,
   - if $h \vdash f \xrightarrow{z(\star)} h' \vdash f'$ then $h' \vdash f' \in F^{\perp\Box}(h)$ and $z \in \mathit{Names}(h)$,

   where the symmetric rules for output are omitted.

In order to characterize the operational semantics of the explicit fusion calculus we need a further ingredient. Let $\iota : |\mathbb{E}| \to \mathbb{E}$ be the inclusion functor and define $U : \mathit{Set}^{\mathbb{E}} \to \mathit{Set}^{|\mathbb{E}|}$ as the functor mapping each presheaf $F : \mathbb{E} \to \mathit{Set}$ into $\iota; F : |\mathbb{E}| \to \mathit{Set}$. Roughly, U forgets the action of a presheaf F on the morphisms of $\mathbb{E}$.

The B-coalgebra corresponding to the explicit fusion calculus is $\langle U(\mathit{Syn}), \alpha\rangle$, where $\mathit{Syn} : \mathbb{E} \to \mathit{Set}$ is the syntactic presheaf introduced in Section 4.1 and

$$\alpha : U(\mathit{Syn}) \to BU(\mathit{Syn})$$

is the operational semantics shown in Fig. 1e.

Unfortunately, the coalgebraic bisimilarity induced by this characterization (hereafter denoted by $\sim^{B}$) does not coincide with $\sim^{io}$ (Definition 1). Roughly, $\sim^{B}$ can be characterized in a similar way to $\sim^{io}$, yet without clause 3. Thus, as pointed out in Section 2.2, the equivalence is not compositional: in order to get a compositional semantics coinciding with $\sim^{io}$, we move to coalgebras on $\mathit{Set}^{\mathbb{E}}$ by employing saturation.

5.2 Saturated Semantics through a Right Kan Extension

The construction of the saturated semantics crucially depends on the functor $U : \mathit{Set}^{\mathbb{E}} \to \mathit{Set}^{|\mathbb{E}|}$. Since Set is complete and co-complete, U has a left and a right adjoint, namely its left and right Kan extensions (and we refer the reader to [22, Chapter X]). For our purposes, it suffices to introduce the right one.

Concretely, the right adjoint of U, hereafter referred as $R : \mathit{Set}^{|\mathbb{E}|} \to \mathit{Set}^{\mathbb{E}}$, can be described as follows. It maps each $F : |\mathbb{E}| \to \mathit{Set}$ into $RF : \mathbb{E} \to \mathit{Set}$. For each object h of $\mathbb{E}$, $RF(h) = \prod_{\sigma:h\to h_1} F(h_1)$. Each $x \in RF(h)$ is a tuple indexed by the arrows of $\mathbb{E}$ outgoing from h; we will write x_σ to denote the value of the tuple x for the index (arrow) σ. For each arrow $\rho : h \to h'$ of $\mathbb{E}$

$$RF(\rho) : \prod_{\sigma:h\to h_1} F(h_1) \to \prod_{\sigma':h'\to h'_1} F(h'_1)$$

maps each $x \in \prod_{\sigma:h\to h_1} F(h_1)$ into the tuple $RF(\rho)(x) \in \prod_{\sigma':h'\to h'_1} F(h'_1)$ such that

$$RF(\rho)(x)_{\sigma'} = x_{\rho;\sigma'}.$$

The unit of the adjunction $U \dashv R$ is a natural transformation $\eta : Id \to RU$. For each presheaf $F : \mathbb{E} \to Set$, the component $\eta_F : F \to RUF$ is an arrow of $Set^{\mathbb{E}}$, i.e., a natural transformation. For each object h of $\mathbb{E}$, $(\eta_F)_h : F(h) \to RUF(h)$ maps $f \in F(h)$ into a tuple $x \in \prod_{\sigma:h\to h_1} F(h_1)$, such that $x_\sigma = F(\sigma)(f)$.

The B-coalgebra $\langle U(Syn), \alpha\rangle$ corresponding to the operational semantics of the calculus can be lifted along the adjunction, obtaining the RBU-coalgebra

$$Syn \overset{\eta_{Syn}}{\to} RU(Syn) \overset{R(\alpha)}{\to} RBU(Syn)$$

The latter actually coincides with the *saturated semantics* of the explicit fusion calculus. Indeed, for each object h of $\mathbb{E}$, η_{Syn} maps an agent $P \in Syn(h)$ in a tuple $x \in RU(Syn)$ such that $x_\sigma = Syn(\sigma)(P)$ for all fusions $\sigma : h \to h'$; while $R(\alpha)$ assigns to each $Syn(\sigma)(P)$ its transitions defined by the operational semantics α. Summarizing, $\eta_{Syn}; R(\alpha)$ assigns to each process P and to each fusion σ the transitions of the process $Syn(\sigma)(P)$: in turn, this situation corresponds to what happens with the saturated transition system, where $P \xrightarrow{\sigma,\lambda}_S Q$ whenever $\sigma(P) \xrightarrow{\lambda} Q$.

5.3 RBU-Coalgebras and Input-Output Bisimilarity

Until now, we have shown that the saturated semantics of the explicit fusion calculus can be modelled as the RBU-coalgebra $\langle Syn, \eta_{Syn}; R(\alpha)\rangle$. In this section, we will show that the corresponding coalgebraic bisimilarity coincides with $\sim^{io}$. First, we give a more concrete characterization of RBU-coalgebras in terms of transitions systems.

Definition 4 (RBU-transition system) An RBU-transition system consists of a presheaf $F : \mathbb{E} \to Set$ and a transition relation $\longrightarrow$ such that

1. states are elements of $\int F$;
2. transitions are typed such that
   - if $h \vdash f \xrightarrow{\sigma,\tau} h'' \vdash f''$ then $\sigma : h \to h'$ and $h'' \vdash f'' \in F^{\Box}(h')$,
   - if $h \vdash f \xrightarrow{\sigma,z\langle y\rangle} h'' \vdash f''$ then $\sigma : h \to h'$, $h'' \vdash f'' \in F^{\Box}(h')$ and $z, y \in Names(h')$,
   - if $h \vdash f \xrightarrow{\sigma,z(\star)} h'' \vdash f''$ then $\sigma : h \to h', h'' \vdash f'' \in F^{\perp^{\Box}}(h')$ and $z \in Names(h')$,

   where the symmetric rules for output are omitted;
3. transitions are preserved by morphisms $\rho : h \to h'$, that is
   - if $h \vdash f \xrightarrow{\rho;\sigma,\lambda} h'' \vdash f''$ then $h' \vdash F(\rho)(f) \xrightarrow{\sigma,\lambda} h'' \vdash f''$;
4. transitions are reflected by morphisms $\rho : h \to h'$, that is
   - if $h' \vdash F(\rho)(f) \xrightarrow{\sigma,\tau} h'' \vdash f'$ then $h \vdash f \xrightarrow{\rho;\sigma,\tau} h'' \vdash f''$.

Proposition 5 *RBU-transition systems are in one to one correspondence with RBU-coalgebras.*

Proof An RBU-coalgebra consists of a pair $\langle F, \beta \rangle$ where $F : E \to Set$ is a presheaf and $\beta : F \to RBU(F)$ is an arrow of $Set^{\mathbb{E}}$, i.e., a natural transformation. For each object h of $\mathbb{E}$ and for each $f \in F(h)$, the function $\beta_h : F(h) \to RBU(F)(h)$ maps f into an $x \in \prod_{\sigma:h\to h_1} BU(F)(h_1)$, that is a tuple of sets of transitions (typed according to B) indexed by the arrows outgoing from h. Thus, a transition $h \vdash f \xrightarrow{\sigma,\lambda} h'' \vdash f''$ means that $(\lambda, f'') \in \beta_h(f)_\sigma$.

The second condition of RBU-transition systems imposes the correct type: for each $\sigma : h \to h'$, $RBU(F)_\sigma$ is equal to $B(F)(h')$, i.e., the set of transitions available at the stage h'. The third and fourth requirement just impose that the transition structure β is a natural transformation between F and $RBU(F)$, i.e., that for all arrows $\rho : h \to h'$ the following diagram commutes.

$$\begin{array}{ccc} F(h) & \xrightarrow{F(\rho)} & F(h') \\ {\scriptstyle \beta_h}\downarrow & & \downarrow{\scriptstyle \beta_{h'}} \\ RBU(F)(h) & \xrightarrow[RBU(F)(\rho)]{} & RBU(F)(h') \end{array}$$

□

We now move to introduce a suitable notion of RBU-bisimulation.

Definition 5 Let $(F, \longrightarrow)$ and $(G, \longrightarrow)$ be two RBU-transition systems. Let $S \subseteq \bigoplus_{h\in\mathbb{E}} F(h) \times G(h)$ be an $\mathbb{E}$-sorted family of symmetric relations. We say that S is an RBU-bisimulation if whenever $f S_h g$ then

1. if $\sigma : h \to i$, then $F(\sigma)(f) S_i G(\sigma)(g)$,
2. if $h \vdash f \xrightarrow{\sigma,\lambda} h' \vdash f'$ then $h \vdash g \xrightarrow{\sigma,\lambda} h' \vdash g'$ and $f' S_{h'} g'$.

We can finally state one of the main results of the paper.

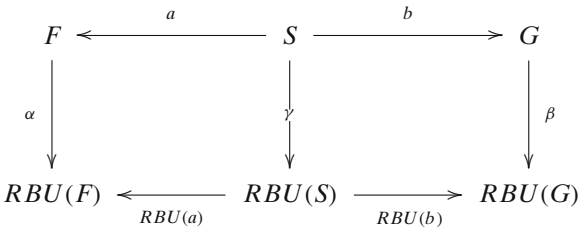
Proposition 6 *RBU-bisimulations are in one to one correspondence with coalgebraic bisimulations for the endo-functor $RBU : Set^{\mathbb{E}} \to Set^{\mathbb{E}}$.*

Proof There exists several definitions of coalgebraic bisimulation: for a detailed comparison, we refer the reader to [33]. In the following we consider the one of "bisimulations as spans of coalgebras", but the proof works also for "bisimulations as cospans".

A presheaf S and two natural transformations $a : S \to F$ and $b : S \to G$ form a bisimulation (S, a, b) between (F, α) and (G, β) if (S, a, b) is a monic span[2] in

[2]The triple (S, a, b) is a monic span (or monic pair) in a category C if for all $h, i \in C$ it holds that $h = i$ whenever $h; a = i; a$ and $h; b = i; b$ do.

$Set^{\mathbb{E}}$ and there is in $Set^{\mathbb{E}}$ an arrow $\gamma : S \to RBU(S)$ making the following diagram commute.



As a start, please note that in any category with binary product, (S, a, b) is a monic span if and only if the induced morphism $\langle a, b\rangle : S \to F \times G$ is a monomorphism. In Definition 5 we require that $S \subseteq \bigoplus_{h\in\mathbb{E}} F(h) \times G(h)$, and this is equivalent to restricting to those $\langle a, b\rangle$ that are injections of S into $F \times G$, instead of just monomorphisms. Therefore, the one to one correspondence holds only up to isomorphism, as it is standard in the theory of coalgebras.

Then, the first requirement of Definition 5 coincides with the fact that S is a presheaf and a, b are natural transformations. Indeed, if $f S_h g$ (i.e., $(f, g) \in S(h)$) and $\sigma : h \to i$, then $S(\sigma)(f, g) \in S(i)$, because S is a functor $\mathbb{E} \to Set$. Notice that $S(\sigma)(f, g) = (F(\sigma)(f), G(\sigma)(g))$, when considering $\langle a, b\rangle$ as the injection of S into $F \times G$. Indeed, since $a : S \to F$ is a natural transformation, then $a_i(S(\sigma)(f, g)) = F(\sigma)a_h(f, g) = F(\sigma)(f)$ (because $\langle a, b\rangle$ is the injection and not simply a mono). Similarly $b_i(S(\sigma)(f, g)) = G(\sigma)b_h(f, g) = G(\sigma)(g)$.

The second requirement of Definition 5 (together with the fact that S is symmetric) coincides with requiring the existence of a γ making the above diagram commute. This is a standard coalgebraic reasoning: the reader is referred to [29, Example 2.1]. □

Condition 2 of Definition 5 can be safely replaced by the following

– if $h \vdash f \xrightarrow{id,\lambda} h' \vdash f'$ then $h \vdash g \xrightarrow{id,\lambda} h' \vdash g'$ and $f' S_{h'} g'$.

Indeed, in RBU-transition systems, the transitions $h \vdash f \xrightarrow{\sigma,\lambda}$ are in one to one correspondence with transitions $h \vdash F(\sigma)(f) \xrightarrow{id,\lambda}$ (by clauses 3 and 4 of Definition 4) that are enforced in the bisimulation game by clause 1 of Definition 5.

Moreover, note that by definition of $\langle Syn, \eta_{Syn}; R(\alpha)\rangle$, for all agents $P \in Syn(h)$ and $P' \in Syn(h')$, $h \vdash P \xrightarrow{id,\lambda} h' \vdash P'$ if and only if $P \xrightarrow{\lambda} P'$.

The correspondence between inside-outside bisimulations and RBU-bisimulations is now evident. Both definitions require that the same equations must hold in the compared processes (this is implicitly expressed in saturated bisimulations by requiring that the relation is indexed over $\mathbb{E}$). Moreover, both definitions require that the relation is closed with respect to name fusions (this is equivalent to requiring that the relation is closed under all arrows of $\mathbb{E}$). And finally, the cases of bound input and bound output (expressed in inside-outside bisimulation by $\mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$) is safely tackled by considering as the bound name a new name for both processes.

6 Conclusions and Further Work

In the paper we introduced the category $\mathbb{E}$ of equivalence relations, and we have shown that it is suitable for providing denotational models of explicit fusions: Section 4.3 shows that the syntax of the explicit fusion calculus coincides with T_Σ (the initial Σ-algebra), while Section 5 guarantees that the semantics of the calculus is an RBU-coalgebra having T_Σ as carrier. The latter result has been obtained adding a further step, after the definition of a more traditional behavioural endo-functor: in order to guarantee the closure of the operational semantics with respect to the morphisms of $\mathbb{E}$ (i.e., that it is a natural transformation in $Set^{\mathbb{E}}$), we exhibited its *saturated* version.

Saturated semantics [2] was formalized in the coalgebraic setting in [3, 5]. Saturation is given there via an "ad hoc" functor, while here we use the adjunction $U \dashv R$ between $Set^{|\mathbb{E}|}$ and $Set^{\mathbb{E}}$, resulting in an elegant and canonical construction. This approach was pursued in [14, 16], yet the link with saturated semantics never shown.

We went to some length for finding the right equivalence for fusions in Section 2, as well as for identifying grounded processes, and we left for future work the proof of the bialgebraic nature of the saturated semantics [35]. Instead of considering grounded processes, an alternative approach would model the algebra of processes quotiented by the structural congruence $\equiv_1$. This proposal falls outside of bialgebraic theory, but it could be recast into frameworks such as those in [11, 21]. Since we aimed at focusing on the introduction of the index category $\mathbb{E}$, we also left this for future investigation.

We would also like to give a coalgebraic characterization of so-called *efficient bisimulation* [36, Definition 9]. There, as it is the case of open bisimulation [30], instead of considering transitions $p \xrightarrow{\sigma,l} p'$ for all possible σ, only some *minimal* σ are taken into account. Note the similarity to the *reactive systems* setting [20]. The exact correspondence with this approach is shown in [4] for the open π-calculus, and it can be extended to deal with explicit fusions. Unfortunately, the definition of efficient bisimulation is asymmetric and thus it seems hard to characterize it through canonical coalgebras. Probably, *normalized coalgebras* [3, 5] could be fruitfully employed.

In the light of finite (and efficient) representations, a typical problem of resource-allocating calculi is the necessity of garbage collecting unused resources in the implementation of the semantics. This is of particular importance in static analysis algorithms such as those for equivalence and model checking. In the case of $Set^{\mathbb{I}}$, the equivalence with the category of *named sets* [13, 17] introduces a garbage-collecting representation of presheaves, which is compositionally extended to functors, and to categories of coalgebras [10]. It would be worth to investigate similar constructions to represent $Set^{\mathbb{E}}$, along the lines of the more general work in [9].

In more general terms, we would like to have a better understanding of the properties of $Set^{\mathbb{E}}$. In particular, we plan to check if alternative characterisations exist, mimicking the correspondence between nominal sets and the Schanuel topos holding for the sub-category of $Set^{\mathbb{I}}$ of pullback-preserving functors. Moreover, should $\mathbb{E}$ be characterized as the result of a free construction, some interesting properties of $Set^{\mathbb{E}}$ would immediately follow from the framework in [34].

As a next step, we would like to address those calculi featuring *distinctions*, such as the D-Fusion [6] and the open π-calculus. A denotational model could be obtained

by considering the category $\mathbb{D}$ of irreflexive graphs [24]. Given the injection $IN_{\mathbb{D}} : \mathbb{D} \to Set$, we should study the comma category $IN_{\mathbb{I}} \downarrow IN_{\mathbb{D}}$, equipping each equivalence class (and each pair of names belonging to them) with an irreflexive relation.

Acknowledgements We are indebted to Sam Staton and to the anonymous referees for important insights that contributed to improve the quality of the paper. We thank Matteo Sammartino for his useful comments on a preliminary version of this paper.

Appendix: A Naive Approach

As a motivation for the saturated semantics introduced in Section 5, in this appendix we show that the operational semantics of the explicit fusion calculus cannot be naively modelled as a coalgebra on $Set^{\mathbb{E}}$. Take as behavioural endo-functor functor $B' : Set^{\mathbb{E}} \to Set^{\mathbb{E}}$ defined as B in Section 5.1 but on $Set^{\mathbb{E}}$ instead of $Set^{|\mathbb{E}|}$.

A B'-coalgebra is a pair $\langle F, \beta \rangle$, for F an object of $Set^{\mathbb{E}}$ (i.e., $F : \mathbb{E} \to Set$ is a functor) and $\beta : F \to B'(F)$ is an arrow of $Set^{\mathbb{E}}$, i.e., a natural transformation between F and $B'(F)$. In other words, $\beta : F \to B'(F)$ is a family of functions $(\beta)_h : F(h) \to B'(F)(h)$ for all $h \in \mathbb{E}$, satisfying suitable naturality requirements.

Definition 6 (B'-transition system) An B'-transition system consists of a presheaf $F : \mathbb{E} \to Set$ and a transition relation $\longrightarrow$ such that

1. states are elements of $\int F$;
2. transitions are typed such that
   - if $h \vdash f \xrightarrow{\tau} h' \vdash f'$ then $h' \vdash f' \in F^{\Box}(h)$,
   - if $h \vdash f \xrightarrow{z\langle y \rangle} h' \vdash f'$ then $h' \vdash f' \in F^{\Box}(h)$ and $z, y \in Names(h)$,
   - if $h \vdash f \xrightarrow{z(\star)} h' \vdash f'$ then $h' \vdash f' \in F^{\perp\Box}(h)$ and $z \in Names(h)$,

   where the symmetric rules for output are omitted;
3. transitions are preserved by morphisms $\sigma = \langle \sigma_1, \sigma_2 \rangle : h \to i$, that is
   - if $h \vdash f \xrightarrow{\tau} h' \vdash f'$ then $i \vdash F(\sigma)(f) \xrightarrow{\tau} i;\, p^* \vdash F(\langle \sigma_1, \sigma_2^* \rangle)(f')$ for p such that $h' = h;\, p$,
   - if $h \vdash f \xrightarrow{z\langle y \rangle} h' \vdash f'$ then $i \vdash F(\sigma)(f) \xrightarrow{\sigma_1(z\langle y \rangle)} i;\, p^* \vdash F(\langle \sigma_1, \sigma_2^* \rangle)(f')$ for p such that $h' = h;\, p$,
   - if $h \vdash f \xrightarrow{z(\star)} h' \vdash f'$ then $i \vdash F(\sigma)(f) \xrightarrow{(\sigma_1)_\perp(z(\star))} i;\, p^* \vdash F(\langle (\sigma_1)_\perp, (\sigma_2)_\perp^* \rangle)(f')$ for p such that $h' = h_\perp;\, p$

   where p, p^*, and σ_2^* are as in the definition of the box operator (see Section 4.2) and where the symmetric rules for output are omitted;
4. transitions are reflected by morphisms $\sigma = \langle \sigma_1, \sigma_2 \rangle : h \to i$, that is
   - if $i \vdash F(\sigma)(f) \xrightarrow{\tau} i' \vdash f'$ then $h \vdash f \xrightarrow{\tau} h'' \vdash f''$ such that $h'' = h;\, p$ and $i' = i;\, p^*$ and $F(\langle \sigma_1, \sigma_2^* \rangle)(f'') = f'$,
   - if $i \vdash F(\sigma)(f) \xrightarrow{z\langle y \rangle} i' \vdash f'$ then $h \vdash f \xrightarrow{u\langle v \rangle} h'' \vdash f''$ such that $h'' = h;\, p$, $i' = i;\, p^*$, $F(\langle \sigma_1, \sigma_2^* \rangle)(f'') = f'$ and $\sigma_1(u\langle v \rangle) = z\langle y \rangle$,
   - if $i \vdash F(\sigma)(f) \xrightarrow{z(\star)} i' \vdash f'$ then $h \vdash f \xrightarrow{u\langle v \rangle} h'' \vdash f''$ such that $h'' = h_\perp;\, p$, $i' = i;\, p^*$, $F(\langle (\sigma_1)_\perp, (\sigma_2)_\perp^* \rangle)(f'') = f'$ and $(\sigma_1)_\perp(u(\star)) = z(\star)$.

where p, p^*, and σ_2^* are as in the definition of the box operator (see Section 4.2) and where the symmetric rules for output are omitted.

The first condition requires that the states are indexed by the object of $\mathbb{E}$. The second condition requires that the transitions have the right type (according to the definition of B'). As for the third and fourth condition, they boil down to require that for arrow $\sigma : h \to i$ (that is, for each renaming of names σ_1 and each enlargement of equivalence classes σ_2), the set of transitions leaving from each state $h \vdash f$ has to be precisely the same as for the set of transitions leaving from $i \vdash f'$, for f' the state $F(\sigma)(f)$ in the stage i.

Proposition 7 *B'-transition systems are in one to one correspondence with B'-coalgebras.*

Proof Consider the requirements of B'-transition system. The first condition just states that we are working with a presheaf. The second condition imposes the correct type. The third and fourth requirement just impose that the transition structure is a natural transformation between F and $B'(F)$, for each presheaf F. □

With this characterization, it is easy to see that the operational semantics of the calculus does not form a B'-coalgebra, because condition 4 of Definition 6 is violated. As an example take $P = \overline{x}\langle z\rangle \mid y\langle w\rangle$ and a morphism σ in $\mathbb{E}$ fusing x and y. Then $Syn(\sigma)(P) = \overline{x}\langle z\rangle \mid y\langle w\rangle \mid x = y \xrightarrow{\tau}$, while $P \not\xrightarrow{\tau}$.

References

1. Bonchi, F., Buscemi, M.G., Ciancia, V., Gadducci, F.: A category of explicit fusions. In: Concurrency, Graphs and Models. Lect. Notes in Comput. Sci., vol. 5065, pp. 544–562. Springer (2008)
2. Bonchi, F., König, B., Montanari, U.: Saturated semantics for reactive systems. In: Procs. of LICS, pp. 69–80. IEEE Computer Society (2006)
3. Bonchi, F., Montanari, U.: Coalgebraic models for reactive systems. In: Procs. of CONCUR. Lect. Notes in Comput. Sci., vol. 4701, pp. 364–380. Springer (2007)
4. Bonchi, F., Montanari, U.: Symbolic semantics revisited. In: Procs. of FOSSACS. Lect. Notes in Comput. Sci., vol. 4962, pp. 395–412. Springer (2008)
5. Bonchi, F., Montanari, U.: Coalgebraic symbolic semantics. In: Procs. of CALCO. Lect. Notes in Comput. Sci., vol. 5728, pp. 173–190. Springer (2009)
6. Boreale, M., Buscemi, M.G., Montanari, U.: D-fusion: A distinctive fusion calculus. In: Procs. of APLAS. Lect. Notes in Comput. Sci., vol. 3302, pp. 296–310. Springer (2004)
7. Boreale, M., Sangiorgi, D.: Some congruence properties for π-calculus bisimilarities. Theor. Comp. Sci. **198**(1–2), 159–176 (1998)
8. Buscemi, M.G., Montanari, U.: Cc-pi: A constraint-based language for specifying service level agreements. In: Procs. of ESOP. Lect. Notes in Comput. Sci., vol. 4421, pp. 18–32. Springer (2007)
9. Ciancia, V., Kurz, A., Montanari, U.: Families of symmetries for the semantics of programming languages. In: Procs. of CMCS. Elect. Notes in Th. Comput. Sci., vol. 264.2, pp. 63–81. Elsevier (2010)
10. Ciancia, V., Montanari, U.: Symmetries, local names and dynamic (de)-allocation of names. Inf. Comput. **208**(12), 1349–1367 (2010)
11. Corradini, A., Heckel, R., Montanari, U.: Compositional SOS and beyond: a coalgebraic view of open systems. Theor. Comp. Sci. **280**(1–2), 163–192 (2002)
12. Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: Procs. of LICS, pp. 193–202. IEEE Computer Society (1999)

13. Fiore, M., Staton, S.: Comparing operational models of name-passing process calculi. Inf. Comput. **204**(4), 524–560 (2006)
14. Fiore, M., Staton, S.: A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In: Procs. of LICS, pp. 49–58. IEEE Computer Society (2006)
15. Fiore, M.P., Moggi, E., Sangiorgi, D.: A fully abstract model for the π-calculus. Inf. Comput. **179**(1), 76–117 (2002)
16. Fiore, M.P., Turi, D.: Semantics of name and value passing. In: Procs. of LICS, pp. 93–104. IEEE Computer Society (2001)
17. Gadducci, F., Miculan, M., Montanari, U.: About permutation algebras, (pre)sheaves and named sets. Higher-Order and Symbolic Computation **19**(2–3), 283–304 (2006)
18. Ghani, N., Yemane, K., Victor, B.: Relationally staged computations in calculi of mobile processes. In: Procs. of CMCS. Elect. Notes in Th. Comput. Sci., vol. 106, pp. 105–120. Elsevier (2004)
19. Hofmann, M.: Semantical analysis of higher-order abstract syntax. In: Procs. of LICS, pp. 204–213. IEEE Computer Society (1999)
20. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: Procs. of CONCUR. Lect. Notes in Comput. Sci., vol. 1877, pp. 243–258. Springer (2000)
21. Lenisa, M., Power, J., Watanabe, H.: Category theory for operational semantics. Theor. Comp. Sci. **327**(1–2), 135–154 (2004)
22. Mac Lane, S.: Categories for the Working Mathematician, 2nd edn. Springer (1998)
23. Miculan, M.: A categorical model of the fusion calculus. In: Procs. of MFPS. Elect. Notes in Th. Comput. Sci., vol. 218, pp. 275–293. Elsevier (2008)
24. Miculan, M., Yemane, K.: A unifying model of variables and names. In: Procs. of FOSSACS. Lect. Notes in Comput. Sci., vol. 3441, pp. 170–186. Springer (2005)
25. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I and II. Inf. Comput. **100**(1), 1–40, 41–77 (1992)
26. Montanari, U., Sassone, V.: Dynamic congruence vs. progressing bisimulation for CCS. Fundam. Inform. **16**(1), 171–199 (1992)
27. Parrow, J., Victor, B.: The fusion calculus: Expressiveness and symmetry in mobile processes. In: Procs. of LICS, pp. 176–185. IEEE Computer Society (1998)
28. Plotkin, G.: A Structural Approach to Operational Semantics. Tech. Rep. DAIMI FN-19, Aarhus University, Computer Science Department (1981)
29. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theor. Comp. Sci. **249**(1), 3–80 (2000)
30. Sangiorgi, D.: A theory of bisimulation for the π-calculus. Acta Inform. **33**(1), 69–97 (1996)
31. Scott, D., Strachey, C.: Toward a mathematical semantics for computer languages. In: Programming Research Group Technical Monograph, vol. PRG-6. Oxford University, Computing Laboratory (1971)
32. Stark, I.: A fully abstract domain model for the π-calculus. In: Procs. of LICS, pp. 36–42. IEEE Computer Society (1996)
33. Staton, S.: Relating coalgebraic notions of bisimulation. In: Procs. of CALCO. Lect. Notes in Comput. Sci., vol. 5728, pp. 191–205. Springer (2009)
34. Tanaka, M., Power, J.: Pseudo-distributive laws and axiomatics for variable binding. Higher-Order and Symbolic Computation **19**(2–3), 305–337 (2006)
35. Turi, D., Plotkin, G.D.: Towards a mathematical operational semantics. In: Procs. of LICS, pp. 280–291. IEEE Computer Society (1997)
36. Wischik, L., Gardner, P.: Strong bisimulation for the explicit fusion calculus. In: Procs. of FOSSACS. Lect. Notes in Comput. Sci., vol. 2987, pp. 484–498. Springer (2004)
37. Wischik, L., Gardner, P.: Explicit fusions. Theor. Comp. Sci. **340**(3), 606–630 (2005)