View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

Electronic Notes in Theoretical Computer Science 33 (2000)
URL: http://www.elsevier.nl/locate/entcs/volume33.html 29 pages

Compositional Constructor Interpretation over Coalgebraic Models for the π–Calculus

Michael Baldamus

University of Karlsruhe
Institute for Computer Design and Fault Tolerance
(Group Prof. Schmid)
baldamus@ira.uka.de

Abstract

The π–calculus and its variants are one of the most important subjects in the field of process algebra. Researchers in the coalgebra community have taken account of that by developing a family of related final coalgebra models for the π–calculus. None of these models has, however, been given with interpretations of the π–calculus constructors as operations on semantic domains. The present paper introduces such interpretations over final coalgebra models for the π–calculus. These models do not exactly belong to the realm of the already existing work. Rather, we emphasise the distinction between a a *ground model* and a *full model*: The ground model is fully abstract with respect to a form of π–calculus ground bisimulation; the full model is built on top of the ground model and is fully abstract with respect to the congruence derived from that ground bisimulation. Also, every semantic object is a 3–tuple with a direct representation of its transformation under renamings. A straightforward adaption of Rutten and Turi's mixed terms technique then yields compositional interpretations of most constructors of the π–calculus on the ground level. These interpretations can be lifted to the full level, again yielding compositionality.

Because input prefixing does not preserve ground bisimilarity, this π–calculus constructor cannot be interpreted compositionally strictly on the ground level. It is therefore given an independent interpretation over the full model.

⋆ The material presented herein was developed while the author was a postdoctoral fellow with the graduate course Specification of Discrete Processes at the Dresden University of Technology. This graduate course is funded by the Deutsche Forschungsgemeinschaft. The author's current position is also funded by the Deutsche Forschungsgemeinschaft, within the project Design and Design Methodology of Embedded Systems.

©2000 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.

1 Introduction

The π–calculus was conceived for the description of concurrent systems with evolving communication topology [22]. Its high degree of expressiveness turned out to be a particularly distinguishing feature, compared to process algebras with static communication topologies (see, for example, work on functions as processes [20]). It is therefore not surprising that significant efforts have been devoted to modelling the π–calculus. Here, we are interested in models that reduce concurrency to nondeterminism; in other words, we are interested in interleaving models. In the case of the π–calculus their kind was lacking for some time. The situation changed by virtue of several works that appeared between 1996 and 1998 [8, 11, 14, 16, 18, 28].

Technically, the π–calculus is about *name passing* via named channels and about the creation of new channels via the creation of new names. A name is *bound* within a (syntactic) process if it is either internal or an input place holder. One of the main difficulties in assigning the π–calculus an interleaving model arises in connection with obtaining both full abstraction and compositional constructor interpretations: It seems as if this combination demands to represent the transformation of processes under all injective replacements of their unbound names by other names (*renamings*, see [8, 11, 14, 28]).

To provide some more detailed background for that, let γ be a k–ary constructor of a CCS–like process algebra, $k \geq 0$, let D be a semantic domain and let $[\![\_]\!]$ be a semantic mapping from the process algebra into D. By an *interpretation* of γ over D, we understand a k–ary function $\gamma_D : D \times \cdots \times D \to D$; by *compositionality* of the interpretation, we understand the property

$$[\![\gamma(P_1, \ldots, P_k)]\!] = \gamma_D([\![P_1]\!], \ldots, [\![P_k]\!]),$$

where P_i ranges over processes, $1 \leq i \leq k$. This means that the interpretation of any arbitrary term is the same as the interpretation of its outermost constructor applied to the interpretations of its immediate subterms. Such interpretations are a classical objective in the world of interleaving models of process algebras (see, for example, [5, 9, 1]). Obtaining them can be far from trivial, especially if the notions of constructor interpretation and compositionality need to be adapted to features such as value passing (see, for example, [17]).

As for coalgebraic models, the issue of interpreting process constructors compositionally was already recognised by Aczel in his pioneering book [2,Chapter 8]. The constructions and proofs at this place were, however, somewhat ad hoc. A systematic solution was provided in the form of the *mixed terms technique* by Rutten [25] and Rutten and Turi [26]. Then, despite the importance of the π–calculus, there is no work on interpreting its constructors over coalgebraic models compositionally, let alone any work on obtaining such interpretations by employing the mixed terms technique.

To present our approach, we have to consider what has already been done

by Honsell et al. [16] and Lenisa [18]. We concentrate on [18], as it is more or less an extension of the theoretical part of [16]. This work introduces several final coalgebra models for the π–calculus. All of them are set–theoretic interleaving models and everyone is fully abstract with respect to some specific form of π–calculus bisimulation. Lenisa's results are particularly unique within the entire realm of π–calculus models in that every single semantic domain is given directly, even if it is fully abstract with respect to the closure of some ground bisimulation under name substitutions [18,Definition 8.4.1 and Proposition 8.4.2].—Adopting terminology from [28] and [8], the predominant approach with respect to layered π–calculus bisimulations can rather be described as reflecting the respective situation by means of a *ground model* and a *full model*; the latter is built on top of the former. Then, Lenisa's models also represent the transformation of π–calculus processes under renamings. This feature is, however, not exploited in obtaining interpretations of the constructors of the π–calculus in the above–described sense.

As in [16, 18], our models are situated within the universe of hypersets in the sense of [2,Part 1] and [12]. Moreover, on the π–calculus side, we concentrate on early ground bisimulation and early congruence [22]. These equivalences are standard ones; at the same time, early congruence is in a certain sense distinguished. Specifically, *barbed congruence* is a congruence with a generic, calculus–independent definition [23]. Its instantiation to the π–calculus yields early congruence [27].

The overview is completed in the course of the following outline of the paper.

Section 2 is concerned with technical preliminaries and with recalling the π–calculus and its early behavioural semantics.

As for Section 3, we note that π–calculus processes must be interpreted within name contexts [8, 11, 14, 16, 18, 28]. To this end, Section 3 is concerned with characterising early ground bisimulation as a CCS–like bisimulation over pairs of the form $P{:}E$, where P is a π–calculus term and E is an *estimate* of the unbound names of P. This property is necessary for the full abstraction proof in Section 4.

As for Section 4, we note that, unlike Lenisa's models, the π–calculus models presented herein exhibit the distinction between a ground model and a full model. The topic of Section 4 is the ground model. This model represents the transitions of processes relative to estimates of their free names, and it represents their transformations under renamings. It is fully abstract with respect to early ground bisimulation.

Every semantic object consists of a name estimate, of a transition set according to the familiar coalgebraic approach and a of a direct representation of the object's transformation under renamings. This particular scheme of associating the representations of transitions and renaming transformations is new. It seems natural and, what is more, it allows one to handle seman-

tic objects in many ways like terms. Specifically, the estimate can be seen as the set of free names of the semantic object, the transition set canonically determines the object's operational semantics and the third component canonically determines the object's transformation under renaming. This machinery, in turn, is exactly what we need in assigning an operational semantics to mixed π–calculus terms, eventually obtaining compositional interpretations of all π–calculus constructors except input prefixing.—There is no compositional interpretation of π–calculus input prefixing over a ground model without considering higher–order functionalities. The reason is that this constructor is also a binding operator roughly in the same sense as λ–abstraction, meaning that it does not preserve ground bisimilarity.

In Section 5 the full model is constructed on top of the ground model. It is fully abstract with respect to early congruence. The ground level constructor interpretations from Section 4 are lifted to the full level, again yielding compositionality. Finally, input prefixing is given an independent compositional interpretation over the full model.

Section 6 briefly discusses non–coalgebraic models for the π–calculus [8, 11, 14, 28]. From the perspective of the present paper, [14] and [28] are especially noteworthy, since they provide compositional interpretations for all π–calculus constructors including input prefixing.

Finally, Section 7 briefly discusses possible future work, namely the issue of accommodating the π–calculus versions of weak equivalences such as observation equivalence.

Because of space considerations, some proofs of the results herein are only sketched or omitted. Full proofs can be found in the technical report version [7].

Acknowledgment

The author would like to thank the anonymous referees of this paper for all their remarks and suggestions.

2 Preliminaries

To begin with, defining equalities are denoted by $:=$. Mere abbreviations of meta–level expressions are designated by $\equiv$.

2.1 Some Basics

We need to work a lot with (sets of) π–calculus names, name replacements, renamings and so on. To this end, the following notational machinery is convenient: First, let S be a set and let $\{x\}$ be another (singleton) set. Then the (ordinary) union of S and $\{x\}$ is denoted by $S + x$, and the (ordinary) asymmetric difference between S and $\{x\}$ is denoted by $S - x$. Second, a replace-

ment operator is given by $S\{x{\mapsto}x'\} := S{-}x{+}x'$ if $x \in S$ and $S\{x{\mapsto}x'\} := S$ if $x \notin S$. Third, the domain and range of a function f are denoted by $\mathrm{dom}(f)$ and $\mathrm{rng}(f)$, respectively. Function spaces on sets S and T occur in the following shapes:

- $(S \to T)$, the set of all functions f with $\mathrm{dom}(f) = S$ and $\mathrm{rng}(f) \subseteq T$
- $(S \rightarrowtail T)$, the set of all injective functions f with $\mathrm{dom}(f) = S$ and $\mathrm{rng}(f) \subseteq T$
- $(S \multimap T)$, the set of all functions f with $\mathrm{dom}(f) \subseteq S$ and $\mathrm{rng}(f) \subseteq T$
- $(S \rightarrowtail\!\!\!\!\multimap T)$, the set of all injective functions f with $\mathrm{dom}(f) \subseteq S$ and $\mathrm{rng}(f) \subseteq T$

Functions are identified with their respective graph, that is, by a function we understand a set of argument/value pairs. Moreover:

- $\downarrow f(x)$ is the usual defined–ness predicate
- $xf := x$ if not $\downarrow f(x)$ and $xf := f(x)$ if $\downarrow f(x)$
- $Sf := \{xf \mid x \in S\}$ whenever it is clear from the context that invoking the preceding definition would not make sense
- $f;g := g \circ f$ whenever $\mathrm{rng}(f) \subseteq \mathrm{dom}(g)$
- Meta–level λ–notation is used. An example is

$$f\{x{\mapsto}y\} := \lambda z \mid z \in \mathrm{dom}(f){+}x \,.\, \begin{cases} f(z) & \text{if } z \neq x \\ y & \text{if } z = x. \end{cases}$$

  Here and everywhere else, the clause behind the bar restricts the domain of the function. The domain is clear from the context if there is no bar and no clause.
- We denote by $f|_S$ the restriction a function f to $\mathrm{dom}(f) \cap S$.
- We denote by t_i the i–th component of a k–tuple t, given that $1 \leq i \leq k$.

Finally, we use the symbol CLASS as an abbreviation of "the category of classes and maps." Everything of the above is generalised to CLASS in the expected way.

2.2 The π–Calculus and Its Early Behavioural Semantics

The π–calculus is built over a name space X that is assumed to be countably infinite and ranged over by letters from the end of the alphabet. The classical π–calculus constructors are $\mathbf{0}$ (inaction), *pre.* (action prefixing), + (nondeterministic choice), | (parallel composition), $(\nu\, x)$ (restriction), $[x = y]$ (match) and ! (replication); the prefixes are $x(y)$ (input via x), $\bar{x}y$ (output of y via x) and τ (silent move). Terms are frequently called processes; they are ranged over by P, Q, ... As for name bindings, $x(y)$ binds y and $(\nu\, x)$ binds x. By $=_\alpha$ and $\mathrm{fn}(P)$ we denote α–convertibility and the set of free names of a

$$\textbf{(in)}\ \ x(y).P \xrightarrow{xz} P\{y\mapsto z\} \qquad \textbf{(pre)}\ \ pre.P \xrightarrow{pre} P \ \ \begin{array}{l} pre \text{ not an} \\ \text{input prefix} \end{array}$$

$$\textbf{(sum)}\ \ \frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'} \qquad \textbf{(par)}\ \ \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\ \ \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$$

$$\textbf{(com)}\ \ \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \textbf{(open)}\ \ \frac{P \xrightarrow{\bar{x}y} P'}{(\nu\, y)P \xrightarrow{\bar{x}(y)} P'}\ \ x \neq y$$

$$\textbf{(close)}\ \ \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu\, y)(P' \mid Q')}\ \ y \notin \mathrm{fn}(P)$$

$$\textbf{(res)}\ \ \frac{P \xrightarrow{\mu} P'}{(\nu\, x)P \xrightarrow{\mu} (\nu\, x)P'}\ \ x \notin \mathrm{n}(\mu) \qquad \textbf{(match)}\ \ \frac{P \xrightarrow{\mu} P'}{[x = x]P \xrightarrow{\mu} P'}$$

$$\textbf{(rep)}\ \ \frac{P \mid !P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'} \qquad \textbf{(alpha)}\ \ \frac{P =_\alpha Q \quad Q \xrightarrow{\mu} Q' \quad Q' =_\alpha P'}{P \xrightarrow{\mu} P'}$$

Table 1
Early SOS axioms and rules for the π–calculus. Only one symmetric half of **(sum)**, **(par)**, **(com)** and **(close)** is stated. Moreover, note that **(rep)** must be so, since we admit un–guarded replication. Also, we adopt **(alpha)**. For our purposes, we have to permit α–conversion on both sides of the transition in its precondition.

process P, respectively.

As for structural operational semantics (SOS), our goal of applying the mixed terms technique in an early setting is best served with an early SOS for the underlying calculus. Its axioms and rules are listed in Table 1. We note that **(open)**, **(close)** and the side condition associated with **(par)** realise the mechanism of scope extrusion and intrusion. We also note that there is a deviation from early SOS as it can be found in the literature (see, for example, [21]), namely that we dispense with bound input actions. These actions are of the form $x(y)$. We do employ actions of the form xz (free input), $\bar{x}y$ (free output) , $\bar{x}(y)$ (bound output) and τ (silent move). This simplification is justified by the fact that early bisimulation does not need to take account of both kinds of input actions explicitly; it suffices to mention only one kind in the definition (see, for example, [24]). What is more, dispensing with bound input actions only requires associating an appropriate side condition with **(close)**. No further modifications to the standard SOS are necessary.

The set of actions, A, is ranged over by μ. It is given by $\mathrm{A} := \{xy \mid x, y \in \mathrm{X}\} \cup \{\bar{x}y \mid x, y \in \mathrm{X}\} \cup \{\bar{x}(y) \mid x, y \in \mathrm{X}\} \cup \{\tau\}$. We need some auxiliary operators on actions, namely n (names), on (object names) and bn (bound

names). They are defined as follows:

$\mu =$	xz	$\bar{x}y$	$\bar{x}(y)$	τ
$\mathrm{n}(\mu) =$	$\{x, z\}$	$\{x, y\}$	$\{x, y\}$	$\emptyset$
$\mathrm{on}(\mu) =$	$\{z\}$	$\{y\}$	$\{y\}$	$\emptyset$
$\mathrm{bn}(\mu) =$	$\emptyset$	$\emptyset$	$\{y\}$	$\emptyset$

Some of these operators have already appeared in Table 1. Another operator that has already appeared in Table 1 is the replacement of a name x by a name y: $P\{x{\mapsto}y\}$. It is assumed that any capture of y is prevented by appropriate α–conversion.

Elaborating on name replacement somewhat, a substitution is an element of the set (X $-\!\circ\!\to$ X); a renaming is an injective substitution. The set of substitutions (and renamings) is ranged over by σ, ρ, ... The simultaneous replacement of each free name x of a term P by $x\sigma$, where σ is some substitution, is denoted by $P\sigma$. Moreover, $x\{y{\mapsto}z\} := x$ if $x \neq y$ and $x\{y{\mapsto}z\} := z$ if $x = y$.—Note that this operator is analogous to but not the same as the replacement operator on sets from the preceding subsection.—Then, prefix expressions such as $x\sigma\cdot(y')$ or $x\{y{\mapsto}z\}\cdot(y')$ contain the "$\cdot$"–symbol as a purely visual delimiter between the subject and object parts. Finally, name replacement always takes precedence over any other operator or constructor. An example: $P|Q\sigma$ is in general not the same as $(P|Q)\sigma$, since $(P|Q)\sigma = P\sigma|Q\sigma$.

We are now ready for defining early ground bisimulation and early congruence. Early ground bisimulation is an adaptation of CCS–like bisimulation to the π–calculus; early congruence is the closure of early ground bisimulation under name substitutions. As its name suggests, early congruence is preserved by all π–calculus constructors; early ground bisimulation is preserved by all π–calculus constructors except input prefixing.

Definition 2.1 *(Early Ground Bisimulation, Early Congruence)*

(i) *A binary relation* $\boldsymbol{R}$ *on* π*–calculus processes is an* ***early ground simulation*** *if* $P\ \boldsymbol{R}\ Q$ *implies: Whenever* $P \xrightarrow{\mu} P'$ *and* $\mathrm{fn}(P, Q) \cap \mathrm{bn}(\mu) = \emptyset$*, then there exists a* Q' *s.t.* $Q \xrightarrow{\mu} Q'$ *and* $P'\ \boldsymbol{R}\ Q'$*. If both* $\boldsymbol{R}$ *and its inverse are early ground simulations, then* $\boldsymbol{R}$ *is an* ***early ground*** **bi*simulation***. ***Early ground bisimilarity***, $\dot{\sim}_{\mathrm{e}}$*, is the union of all early ground bisimulations.*

(ii) ***Early congruence***, $\sim_{\mathrm{e}}$*, is given by* $P \sim_{\mathrm{e}} Q$ *if* $P\sigma \dot{\sim}_{\mathrm{e}} Q\sigma$ *for every* σ*.*

We note that early ground bisimulation is a variant of strong bisimulation over CCS–like process algebras (cf. [19]), or a variant of strong bisimulation over labelled transition systems, for that matter (cf. [26]). A consequence is that early ground bisimilarity, being the union of all early ground bisimulations, is itself an early ground bisimulation.

The difference to the standard notion of strong bisimulation consists of the fact that the bisimulation game is constrained by the side condition $\mathrm{fn}(P,Q) \cap \mathrm{bn}(\mu) = \emptyset$. This modification is necessary since the set of free names of a π–calculus process P restricts what names can occur in object role in bound outputs or inputs of P. Without the modification, fewer π–calculus processes than conceptually sensible would be early ground bisimilar.

3 Characterising Early Ground Bisimulation

Having completed the preliminaries, our first step consists of showing that early ground bisimulation can be characterised as a CCS–like bisimulation, notwithstanding what has been pointed out at the end of the previous section. We follow closely an earlier characterisation of the late form of ground bisimulation [10].

The basis of the characterisation consists of introducing **estimated processes** of the form $P{:}E$, where E is a name **estimate**, that is, a subset of the name space X. Such a construct is **sound** if $\mathrm{fn}(P)$ is a subset of E. In particular $P{:}\mathrm{fn}(P)$ is always sound. An appropriate SOS of soundly estimated processes is derived from the early SOS as shown below. The idea, then, is to characterise $P \sim_{\mathrm{e}} Q$ as $P{:}E \sim_{\mathrm{CCS}} Q{:}E$, where $\sim_{\mathrm{CCS}}$ denotes CCS–like bisimulation over that SOS. This scheme means that one has to provide a common estimate E of both P and Q first. We refrain from giving much more explanation, since this section is an adaption and simplification of the corresponding part of [10].

Technically, an estimate must be such that $\mathrm{X} \setminus E$ is infinite. This condition makes sense because, for every P, $\mathrm{fn}(P)$ is finite. It is necessary since estimated processes need be closed under finite enlargements of the estimate. The set of estimates is ranged over by E and F; the set of soundly estimated processes is denoted by SE; the aforementioned SOS is given by the **estimation rule**:

$$\textbf{(estimation)} \qquad \frac{P \xrightarrow{\mu} P'}{P{:}E \xrightarrow{\mu} P'{:}E \cup \mathrm{on}(\mu)} \quad \mathrm{bn}(\mu) \cap E = \emptyset$$

Note that the estimate never shrinks. Specifically, the loss of free names due to discarded nondeterministic branches is not taken into account. The estimate is, in fact, updated only according to μ, and nothing else, so as to preserve soundness.

On this basis, it makes sense to instantiate the notion of CCS–like bisimulation to soundly estimated processes. The difference to ground bisimulation is that the bisimulation game is not restricted by any side condition.

Definition 3.1 *A binary relation* $\boldsymbol{R}$ *on* SE *is a* ***CCS–like simulation*** *if* $P{:}E \ \boldsymbol{R}\ Q{:}E$ *implies: Whenever* $P{:}E \xrightarrow{\mu} P'{:}E'$, *then there exists a* Q' *s.t.* $Q{:}E \xrightarrow{\mu} Q'{:}E'$ *and* $P'{:}E' \ \boldsymbol{R}\ Q'{:}E'$. *If both* $\boldsymbol{R}$ *and its inverse are CCS–like simulations, then* $\boldsymbol{R}$ *is a* ***CCS–like*** **bi*simulation***. *By* $\sim_{\mathrm{CCS}}$, *we denote*

***CCS–like bisimilarity**, that is, the union of all CCS–like bisimulations.*

Finally, the characterisation theorem has two parts: The first one is as described above; the second one is a corollary which states that early congruence is characterised by the closure of CCS–like bisimulation over soundly estimated processes under name substitutions. The idea of the first part consists of replacing the restriction of the bisimulation game in the definition of ground bisimulation. The substitute is the common estimate in combination with the fact that the transition semantics of every estimated process is canonically embedded in the transition semantics of the process itself. This embedding just forgets about all estimates.

Theorem 3.2 *(Characterisation) For all $P{:}E$, $Q{:}E \in \mathrm{SE}$:*

(i) *$P \dot{\sim}_{\mathrm{e}} Q$ if, and only if, $P{:}E \sim_{\mathrm{CCS}} Q{:}E$*

(ii) *$P \sim_{\mathrm{e}} Q$ if, and only if, $P\sigma{:}E\sigma \sim_{\mathrm{CCS}} Q\sigma{:}E\sigma$ for each $\sigma \in (\mathrm{X} \to \mathrm{X})$*

Proof. See [7]. □

4 The Ground Model

4.1 Semantic Domain and Semantic Mapping

At this place, it should be instructive to consider right away what ground interpretations look like. First of all, we recall that the π–calculus requires to interpret processes within name contexts; so the semantic mapping of the ground model, $[\![\_]\!]_{\mathrm{g}}$, is defined on the set SE. The shape of the ground interpretation of a soundly estimated process is then illustrated by the following recurrence property:

$$(1)\quad [\![P{:}E]\!]_{\mathrm{g}} = \Big\langle E,\ \{\langle \mu, [\![P'{:}E']\!]_{\mathrm{g}}\rangle \mid P{:}E \xrightarrow{\mu} P'{:}E'\},\ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E').[\![P\sigma{:}E']\!]_{\mathrm{g}}\Big\rangle$$

The first component of $[\![P{:}E]\!]_{\mathrm{g}}$ is the set E. The second component is a transition set. The third component is the direct representation of the transformation of $P{:}E$ under renamings, which is a function on all name replacements σ on E, and all E' with $E\sigma \subseteq E'$, that maps σ and E' to the ground interpretation of $P\sigma{:}E'$.—It is not feasible to adopt a more immediate scheme, where E' is not mentioned and σ is mapped to $[\![P\sigma{:}E\sigma]\!]_{\mathrm{g}}$. The reason has to do with the binding semantics of input prefixing and restriction. Specifically, if one wants to obtain compositional interpretations of these constructors within the basic framework of the present paper, then the binding aspect leads to situations where one has to consider how an estimated process is transformed when the estimate is extended. This requirement affects the full model directly and the ground model as a consequence of that (cf. Section 5). It can be met elegantly by adopting binary renaming functions of the above–described kind.

As for defining the semantic domain and the semantic mapping of the ground model, a functor $\mathrm{F_g}$ on CLASS is given by

$$\mathrm{F_g}(C) := \mathrm{Pow(X)} \times \mathrm{Pow(A} \times C) \times (((\mathrm{X} \rightarrowtail\!\!\!\!\circ\!\!\!\!\rightarrow \mathrm{X}) \times \mathrm{Pow(X)}) \multimap\!\!\!\!\circ\!\!\!\!\rightarrow C)$$

for every class C, and by

$$\mathrm{F_g}(f)(\langle E, T, r\rangle) := \Big\langle E,\ \ \{\langle \mu, f(S')\rangle \mid \langle \mu, S'\rangle \in T\},\ \ r;f \Big\rangle$$

for every map f, $\langle E, T, r\rangle \in \mathrm{F_g}(\mathrm{dom}(f))$. By straightforward reasoning, $\mathrm{F_g}$ is set–based and inclusion–preserving. Hence the largest point of $\mathrm{F_g}$, $\mathrm{J_g}$, exists. This class serves as semantic domain of the ground model. It is ranged over by θ.

To obtain the semantic mapping, we apply the special final coalgebra theorem (SFC). Hence, for every class C, we need a suitable V_C–translation for $\mathrm{F_g}$ [2]. To this end, note that $\mathrm{F_g}(C)$ is a set whose shape is transitively determined by the representation of pairs, tuples and functions. The image of $\mathrm{F_g}(C)$ under the V_C–translation for $\mathrm{F_g}$ is obtained by considering those transitive elements of this set that are from C. They are turned into urelements and that allows us to prove that $\mathrm{F_g}$ is uniform on maps (see [7]).

Next, a decomposition function $\mathrm{d_g} \in (\mathrm{SE} \to \mathrm{F_g(SE)})$ is given by

$$\begin{aligned}&\mathrm{d_g}(P{:}E)\\ &\quad := \Big\langle E,\ \ \{\langle \mu, P'{:}E'\rangle \mid P{:}E \xrightarrow{\mu} P'{:}E'\},\ \ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E').P\sigma{:}E' \Big\rangle\end{aligned}$$

The sought–after mapping is the final $\mathrm{F_g}$–coalgebra homomorphism from $\langle \mathrm{SE}, \mathrm{d_g}\rangle$ to $\langle \mathrm{J_g}, \mathrm{id_{J_g}}\rangle$:

$$\begin{array}{ccc} \mathrm{SE} & \xrightarrow{\quad [\![\text{-}]\!]_{\mathrm{g}} \quad} & \mathrm{J_g} \\ {\scriptstyle \mathrm{d_g}}\downarrow & & \| \\ \mathrm{F_g(SE)} & \xrightarrow[\mathrm{F_g}([\![\text{-}]\!]_{\mathrm{g}})]{} & \mathrm{F_g(J_g)} \end{array}$$

It exists since $\mathrm{F_g}$ is inclusion preserving and uniform on maps, meaning that we can apply the SFC. Returning to (1), proving this property is now a straightforward matter of rewriting. We also have the first major result:

Theorem 4.1 *(Ground Full Abstraction) For all $P{:}E$, $Q{:}E \in \mathrm{SE}$:*

$$P \mathrel{\dot{\sim}_{\mathrm{e}}} Q \text{ if, and only if, } [\![P{:}E]\!]_{\mathrm{g}} = [\![Q{:}E]\!]_{\mathrm{g}}$$

Proof. (Idea) By proving that, for all $P{:}E$, $Q{:}E \in \mathrm{SE}$:

$$P{:}E \sim_{\mathrm{CCS}} Q{:}E \text{ if, and only if, } [\![P{:}E]\!]_{\mathrm{g}} = [\![Q{:}E]\!]_{\mathrm{g}}$$

Then the rest follows by Theorem 3.2. □

On a final remark with respect to the shape of ground interpretations, it would theoretically suffice to represent all inputs with a new name via just one sample transition (cf. [8, 11, 14, 16, 18]). For the purposes of the present paper, however, not sampling such inputs is easier to handle overall. A side consequence is that the transition semantics is infinitely branching. This aspect, however, is accommodated by the hyperset framework without any problem (cf. [2]).

4.2 Constructor Interpretations

To proceed, consider the BNF–like grammar below. It is the starting point of applying the mixed terms technique within the framework of the ground model. One speaks of mixed terms because semantic objects are admitted as constants. For technical reasons, ordinary terms are also admitted as constants; they carry a "$\widehat{\ }$"–symbol whenever they appear in this role.

$$M ::= \theta \mid \widehat{P} \mid \mathbf{0} \mid pre^{-}.M \mid M + M \mid M \mid M \mid (\nu\, x)M \mid [x = y]M \mid\, !M$$
$$pre^{-} ::= \bar{x}y \mid \tau$$

There is no input prefixing because it is not possible to interpret this constructor on the ground level compositionally. The language of mixed terms is ranged over by M and N.

At this place, it should be pointed out that the mixed–terms technique is not syntactic, although its name suggests otherwise. For one, compositional constructor interpretations are the "right" ones from a strictly extensional viewpoint. Moreover, the intuition behind transition sets and similar representations is clearly operational. Mixed terms are an attempt at working with it as explicitly and efficiently as possible. Here, there is no additional information other than the representation of transformations under renamings. That makes it actually difficult to imagine how compositional constructor interpretations over the ground model could be attained in any inherently better way than by using mixed terms.

The next step consists of assigning mixed terms an early SOS by duplicating the early π–calculus SOS. As the π–calculus SOS involves the $\mathrm{fn}(\_)$–operator and α–conversion, these components must be duplicated first. The $\mathrm{fn}(\_)$–operator is given by straightforward structural induction:

$$\begin{aligned}
\mathrm{fn}(\theta) &:= E \quad \text{given that } \theta = \langle E, T, r\rangle \\
\mathrm{fn}(\widehat{P}) &:= \mathrm{fn}(P) \\
\mathrm{fn}(\gamma^{-}(M_1, \ldots, M_{\mathrm{ar}(\gamma^{-})})) &:= \mathrm{n}(\gamma^{-}) \cup \textstyle\bigcup_{i\in\{1,\ldots,\mathrm{ar}(\gamma^{-})\}} \mathrm{fn}(M_i) \\
\mathrm{fn}((\nu\, x)M) &:= \mathrm{fn}(M) - x
\end{aligned}$$

Here, γ^{-} ranges over all constructors of mixed terms except the new–operator,

that is, it can be of the form $\mathbf{0}$, $pre^{-}.$, $+$, $|$, $[x = y]$ or $!$ with

$$\mathrm{n}(\gamma^{-}) := \begin{cases} \emptyset & \text{if } \gamma^{-} \neq \bar{x}y. \text{ and } \gamma^{-} \neq [x = y] \\ \{x, y\} & \text{if } \gamma^{-} = \bar{x}y. \text{ or } \gamma^{-} = [x = y]. \end{cases}$$

The arity of γ^{-} is denoted by $\mathrm{ar}(\gamma^{-})$ and $\gamma^{-}(M_1, \ldots, M_{\mathrm{ar}(\gamma^{-})})$ denotes a term that is built according to the BNF above with γ^{-} as outermost constructor and with M_1, ... and $M_{\mathrm{ar}(\gamma^{-})}$ as immediate sub–terms. For example, $+(M_1, M_2)$ denotes $M_1 + M_2$.

Then, we need to duplicate the name replacement operator before we can duplicate the notion of α–convertibility. Analogously to the previous one, both of these notions are given by straightforward structural induction. The name replacement operator is given by four defining equations, as follows:

$$\langle E, T, r\rangle\{x \mapsto y\} := \begin{cases} \langle \emptyset, \emptyset, \emptyset\rangle & \text{if not } \downarrow\theta' \\ \theta' & \text{if } \downarrow\theta' \end{cases}$$

where $\theta' \equiv r(\mathrm{id}_E\{x \mapsto y\}, E\{x \mapsto y\})$

$$\widehat{P}\{x \mapsto y\} := \widehat{P\{x \mapsto y\}}$$
$$\gamma^{-}(M_1, \ldots, M_{\mathrm{ar}(\gamma^{-})})\{x \mapsto y\} := \gamma^{-}\{x \mapsto y\}\,(M_1\{x \mapsto y\}, \ldots, M_{\mathrm{ar}(\gamma^{-})}\{x \mapsto y\})$$

where

$$\gamma^{-}\{x \mapsto y\} := \begin{cases} \gamma^{-} & \text{if } \gamma^{-} \neq \bar{u}v. \text{ and } \gamma^{-} \neq [u = v] \\ \bar{u}\{x \mapsto y\}\cdot v\{x \mapsto y\} & \text{if } \gamma^{-} = \bar{u}v \\ [u\{x \mapsto y\} = v\{x \mapsto y\}] & \text{if } \gamma^{-} = [u = v] \end{cases}$$

$$((\nu\, x)M)\{y \mapsto z\} := \begin{cases} (\nu\, x)M & \text{if } x = y \\ (\nu\, x')M\{x \mapsto x'\}\{y \mapsto z\} & \text{if } x \neq y \text{ and } x = z \\ (\nu\, x)M\{y \mapsto z\} & \text{if } x \neq y \text{ and } x \neq z \end{cases}$$

where, in the case of $x \neq y$ and $x = z$, x' must be chosen s.t. $x' \notin \mathrm{fn}(M)+y+z$. One possibility of doing so consists of fixing an arbitrary function new : Pow(X) $\to$ X s.t. new(X) $\notin$ X if X is finite. Whenever the need arises, this function could be applied to the set $\mathrm{fn}(M)+y+z$ to obtain x'.

The notion of α–convertibility is given as follows:

$$\begin{aligned}
&\theta =_\alpha \theta' \quad \text{if } \theta = \theta' \\
&\widehat{P} =_\alpha \widehat{P'} \text{ if } P =_\alpha P' \\
&\gamma^-(M_1, \ldots, M_{\mathrm{ar}(\gamma^-)}) =_\alpha \gamma^-(M'_1, \ldots, M'_{\mathrm{ar}(\gamma^-)}) \\
&\qquad \text{if } M'_1 =_\alpha M'_1, \ \ldots \text{ and } M_{\mathrm{ar}(\gamma^-)} =_\alpha M'_{\mathrm{ar}(\gamma^-)} \\
&(\nu\, x)M =_\alpha (\nu\, x')M' \\
&\qquad \text{if } M\{x \mapsto x'\} =_\alpha M' \ \text{ provided that } x' \notin \mathrm{fn}(M) - x
\end{aligned}$$

All of these definitions are indeed completely analogous to their counterparts within the framework of ordinary terms. The only difference consists of the fact that semantic objects and processes that appear as constants need to be accommodated. Specifically, the free names of a semantic object are determined by its first component; the transformation of a semantic object under name replacement is determined by its third component; the restriction of $=_\alpha$ to semantic objects is identical to the diagonal on $\mathrm{J_g}$; a process that appears as a constant behaves exactly in the same way as the process itself.

Eventually, it is possible to duplicate the early SOS from Table 1, adding an axiom for semantic objects and an axiom for process terms that appear as constants. The ensuing set of axioms and rules can be found in Table 2. Within this framework, the transitions of semantic objects are determined by second components, that is, by their transition sets (cf. [25]). Processes that appear as constants behave exactly in the same way as the processes themselves. What is left is once again completely analogous to the non–mixed case.

At this place, it should be pointed out that the language of mixed terms contains a lot of junk. Not only that a lot of constants from $\mathrm{J_g}$ are not reached by $[\![\_]\!]_\mathrm{g}$: Un–reached objects and, in consequence, mixed terms may also be "unreasonable" wrt. the free names and name replacement operators, α–conversion and the SOS. Specifically, just about every consistency condition satisfied by this machinery within the framework of processes is violated within the framework of mixed terms. For example,

$$\mathrm{fn}(P\{x \mapsto y\}) = \mathrm{fn}(P)\{x \mapsto y\}$$

is a valid proposition whereas, at the same time, "$\mathrm{fn}(\theta\{x \mapsto y\}) = \mathrm{fn}(\theta)\{x \mapsto y\}$" and "$\mathrm{fn}(M\{x \mapsto y\}) = \mathrm{fn}(M)\{x \mapsto y\}$" are wrong. Another example is

$$P \xrightarrow{\bar{x}(y)} P' \text{ implies } \mathrm{fn}(P') \subseteq \mathrm{fn}(P) + y,$$

since "$\theta \xrightarrow{\bar{x}(y)} \theta'$ implies $\mathrm{fn}(\theta') \subseteq \mathrm{fn}(\theta) + y$" and "$M \xrightarrow{\bar{x}(y)} M'$ implies $\mathrm{fn}(M') \subseteq \mathrm{fn}(M) + y$" are wrong. This situation is so since we do not consider many–

$$\textbf{(object)}\ \ \theta \xrightarrow{\mu} \theta' \ \text{ provided that } \theta = \langle E, T, r\rangle \text{ with } \langle \mu, \theta'\rangle \in T$$

$$\textbf{(process)}\ \ \widehat{P} \xrightarrow{\mu} \widehat{P'} \ \text{ provided that } P \xrightarrow{\mu} P'$$

$$\textbf{(pre}^{-}\textbf{)}\ \ pre^{-}.M \xrightarrow{pre^{-}} M \qquad \textbf{(sum)}\ \ \frac{M \xrightarrow{\mu} M'}{M + N \xrightarrow{\mu} M'}$$

$$\textbf{(par)}\ \ \frac{M \xrightarrow{\mu} M'}{M \mid N \xrightarrow{\mu} M' \mid N}\ \ \mathrm{bn}(\mu) \cap \mathrm{fn}(N) = \emptyset$$

$$\textbf{(com)}\ \ \frac{M \xrightarrow{xy} M' \quad N \xrightarrow{\bar{x}y} N'}{M \mid N \xrightarrow{\tau} M' \mid N'} \qquad \textbf{(open)}\ \ \frac{M \xrightarrow{\bar{x}y} M'}{(\nu\, y)M \xrightarrow{\bar{x}(y)} M'}\ \ x \neq y$$

$$\textbf{(close)}\ \ \frac{M \xrightarrow{xy} M' \quad N \xrightarrow{\bar{x}(y)} N'}{M \mid N \xrightarrow{\tau} (\nu\, y)(M' \mid N')}\ \ y \notin \mathrm{fn}(M)$$

$$\textbf{(res)}\ \ \frac{M \xrightarrow{\mu} M'}{(\nu\, x)M \xrightarrow{\mu} (\nu\, x)M'}\ \ x \notin \mathrm{n}(\mu) \qquad \textbf{(match)}\ \ \frac{M \xrightarrow{\mu} M'}{[x = x]M \xrightarrow{\mu} M'}$$

$$\textbf{(rep)}\ \ \frac{M \mid !M \xrightarrow{\mu} M'}{!M \xrightarrow{\mu} M'} \qquad \textbf{(alpha)}\ \ \frac{M =_{\alpha} N \quad N \xrightarrow{\mu} N' \quad N' =_{\alpha} M'}{M \xrightarrow{\mu} M'}$$

Table 2
Early structural axioms and rules for mixed terms.

sorted coalgebras. That would allow us to regard estimates as sorts, and that, in turn, would allow us to redefine the functor $\mathrm{F_g}$ so as to impose an appropriate set of consistency conditions on the ground model. These conditions would then be inherited by the above–mentioned machinery on the language of mixed terms. We do not proceed in this way because of three reasons: First, the junk contained in $\mathrm{J_g}$ does not interfere with the non–junk as long as one sticks to the constructor interpretations introduced below. The very fact that these interpretations are compositional is the reason for that. Second, many–sorted coalgebras would require modifying the underlying theory of coalgebras over the universe of hypersets. Third, not ruling out anything avoids the danger of being too restrictive; the ground model as it is includes every semantic object that could ever be of interest. As an aside, unreasonable semantic objects seem to be less abundant but not absent in the case of Lenisa's models, too.

Next, we duplicate the name estimate mechanism: An **estimated mixed term** is of the form $M{:}E$, and the set of all such constructs is denoted by ME. Analogously to the above material, the estimation rule over estimated mixed

terms is completely analogous to its non–mixed counterpart:

$$\textbf{(estimation)} \qquad \frac{M \xrightarrow{\mu} M'}{M{:}E \xrightarrow{\mu} M'{:}E \cup \mathrm{on}(\mu)} \quad \mathrm{bn}(\mu) \cap E = \emptyset$$

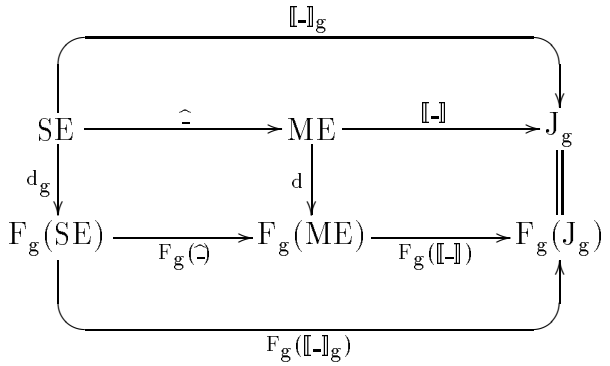
An element of ME is not required to be sound. This trick allows us to attain total interpretations of the constructors of the π–calculus, that is, interpretations that are defined on both junk and non–junk.

The last prerequisite is the semantic mapping on ME. It is the final $\mathrm{F_g}$–coalgebra homomorphism from $\langle \mathrm{ME}, \mathrm{d} \rangle$ to $\langle \mathrm{J_g}, \mathrm{id_{J_g}} \rangle$, where

$$\begin{aligned} &\mathrm{d}(M{:}E) \\ &:= \Big\langle E,\ \ \{\langle \mu, M'{:}E' \rangle \mid M{:}E \xrightarrow{\mu} M'{:}E'\},\ \ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E')\,.\, M\sigma{:}E' \Big\rangle, \end{aligned}$$

meaning that, once again, everything is completely analogous to the non–mixed case. We denote the mapping by $[\![\_]\!]$.

To sum up, let $\widehat{\_}$ be the canonical embedding of SE into ME, that is, the embedding that maps every $P{:}E$ to $\widehat{P}{:}E$. Then the overall situation is depicted by the following diagram:



It is easy to see that the left inner square of this diagram commutes, so the whole diagram commutes, so the semantic mappings are compatible:

Lemma 4.2 $[\![P{:}E]\!]_{\mathrm{g}} = [\![\widehat{P}{:}E]\!]$

Having put all prerequisites in place, the ground interpretation of any arbitrary γ^- is an element of the map space $(\mathrm{J_g^{ar(\gamma^-)}} \to \mathrm{J_g})$, where $\mathrm{J_g^{ar(\gamma^-)}}$ comprises all argument vectors of length $\mathrm{ar}(\gamma^-)$. We denote this operation by γ_{g}^-. It is given by

$$\gamma_{\mathrm{g}}^-(\theta_1, \ldots, \theta_{\mathrm{ar}(\gamma^-)}) := [\![\gamma^-(\theta_1, \ldots, \theta_{\mathrm{ar}(\gamma^-)}){:}\mathrm{n}(\gamma^-) \cup E]\!]$$

with $E := \bigcup_{i \in \{1,\ldots,\mathrm{ar}(\gamma^-)\}} \mathrm{fn}(\theta_i)$. That is, the result of the application of γ_{g}^- to $\theta_1, \ldots$ and $\theta_{\mathrm{ar}(\gamma^-)}$ is given by the ground interpretation of the estimated mixed term $\gamma^-(\theta_1, \ldots, \theta_{\mathrm{ar}(\gamma^-)}){:}\mathrm{n}(\gamma^-) \cup E$, where E comprises the free names of θ_1,

... and $\theta_{\mathrm{ar}(\gamma^-)}$. This scheme is indeed the same as in [25]. The only difference is that $\mathrm{n}(\gamma^-)$ and E have to be extracted, so as to estimate $\gamma^-(\theta_1, \ldots, \theta_{\mathrm{ar}(\gamma^-)})$.

To finish, the ground interpretation of $(\nu\, x)$, $x \in \mathrm{X}$, is a family of elements of the map space $(\mathrm{J_g} \to \mathrm{J_g})$. There is one family member for each estimate E. It is denoted by $(\nu\, x)_{E\,\mathrm{g}}$ and its definition is

$$(\nu\, x)_{E\,\mathrm{g}}\theta := [\![(\nu\, x)\theta{:}E]\!]\,.$$

That is, the result of the application of $(\nu\, x)_{E\,\mathrm{g}}$ to θ is given by the ground interpretation of the estimated mixed term $(\nu\, x)\theta{:}E$. The annotation with E is necessary since the new–operator is a name binder. Specifically, the only theoretical alternative would be a restriction operator given by (a) "$(\nu\, x)_{\mathrm{g}}\langle E, T, r\rangle$" $:=$ $[\![(\nu\, x)\langle E, T, r\rangle{:}E{-}x]\!]$, (b) "$(\nu\, x)_{\mathrm{g}}\langle E, T, r\rangle$" $:=$ $[\![(\nu\, x)\langle E, T, r\rangle{:}E]\!]$" or (c) "$(\nu\, x)_{\mathrm{g}}\langle E, T, r\rangle$" $:=$ $[\![(\nu\, x)\langle E, T, r\rangle{:}E{+}x]\!]$. In all cases it is easy to see that compositionality would not hold: The reason is that the compositionality property must always be $[\![(\nu\, x)P{:}E]\!]_{\mathrm{g}} =$ "$(\nu\, x)_{\mathrm{g}}[\![P{:}E{+}x]\!]_{\mathrm{g}}$". In case (a) this equality breaks down if $x \in E$, namely via $[\![(\nu\, x)P{:}E]\!]_{\mathrm{g}} = \langle E, \ldots\rangle \neq \langle E{+}x{-}x, \ldots\rangle =$ "$(\nu\, x)_{\mathrm{g}}[\![P{:}E{+}x]\!]_{\mathrm{g}}$". In case (b) and case (c) it breaks down in very similar manners if $x \notin E$.

Turning back to $(\nu\, x)_{E\,\mathrm{g}}$, note that the application of $(\nu\, x)_{E\,\mathrm{g}}$ to θ makes practical sense only if $\mathrm{fn}(\theta) \subseteq E{+}x$, although the result is always defined in the technical sense.

Finally, the above interpretations are the right ones at least extensionally. The reason is that they are compositional:

Theorem 4.3 *(Compositionality) For all* $P_1{:}E_1, \ldots, P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)} \in \mathrm{SE}$*:*

$$[\![\gamma^-(P_1, \ldots, P_{\mathrm{ar}(\gamma^-)}){:}\mathrm{n}(\gamma^-) \cup \textstyle\bigcup_{i\in\{1,\ldots,\mathrm{ar}(\gamma^-)\}} E_i]\!]_{\mathrm{g}} = \gamma^-_{\mathrm{g}}([\![P_1{:}E_1]\!]_{\mathrm{g}}, \ldots, [\![P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)}]\!]_{\mathrm{g}})$$

Moreover, for each $P{:}E \in \mathrm{SE}$*:*

$$[\![(\nu\, x)P{:}E]\!]_{\mathrm{g}} = (\nu\, x)_{E\,\mathrm{g}}[\![P{:}E{+}x]\!]_{\mathrm{g}}$$

Proof. (Idea) Structurally the same as the proof of the compositionality result in [25] (see appendix). □

5 The Full Model

5.1 Semantic Domain and Semantic Mapping

Having completed the ground model, the third step essentially consists of closing $[\![\_]\!]_{\mathrm{g}}$ under name substitutions, so as to obtain full abstraction wrt. early congruence in the first place. It is, on this basis, possible to interpret all π–calculus constructors compositionally. The ensuing model is therefore called the full model.

Analogously to (1), the following recurrence equation illustrates the precise shape of the full interpretation of any arbitrary sound estimation $P{:}E$:

$$(2) \qquad [\![P{:}E]\!]_{\mathrm{f}} = \Big\langle E,\ \ \lambda\sigma\,.\,[\![P\sigma{:}E\sigma]\!]_{\mathrm{g}},\ \ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E')\,.\,[\![P\sigma{:}E']\!]_{\mathrm{f}} \Big\rangle$$

The first and the third component are completely analogous to their counterparts within the framework of the ground model; the second component is a function on $(\mathrm{X} \to \mathrm{X})$, which maps every σ to the ground interpretation of $[\![P\sigma{:}E\sigma]\!]_{\mathrm{g}}$. Intuitively, this scheme represents the double nature of the π–calculus process P: For one, P may be regarded as a function in its free names, since input prefixing applied to P may turn every such name into a placeholder. Besides that, P is also a process in the usual sense, since it has a transitional semantics. This aspect is modelled by the possibility of applying the middle component of $[\![P{:}E]\!]_{\mathrm{f}}$ to the identity substitution, so as to obtain the ground interpretation of $P{:}E$.

The technical basis of (2) is a functor F_{f} on CLASS that is given by

$$\mathrm{F}_{\mathrm{f}}(C) := \mathrm{Pow}(\mathrm{X}) \times ((\mathrm{X} \to \mathrm{X}) \to \mathrm{J}_{\mathrm{g}}) \times (((\mathrm{X} \rightarrowtail\!\!\!\!\circ\ \mathrm{X}) \times \mathrm{Pow}(\mathrm{X})) \multimap C)$$

for every class C, and by

$$\mathrm{F}_{\mathrm{f}}(f)(\langle E, s, r\rangle) := \langle E, s, r; f\rangle$$

for every map f, $\langle E, s, r\rangle \in \mathrm{F}_{\mathrm{f}}(\mathrm{dom}(f))$. Analogously to the situation in connection with F_{g}, it is straightforward to see that F_{f} is set–based and inclusion–preserving. Thus, the largest fixed point of F_{f}, J_{f}, exists. This class is ranged over by φ and ψ .

Next, a V_C–translation for F_{f}, where C is any arbitrary class, can be obtained in practically the same way as the corresponding V_C–translation within the framework of the ground model. It is thus possible to apply the SFC, so as to obtain the semantic mapping of the full model. To this end, a function $\mathrm{d}_{\mathrm{f}} \in (\mathrm{SE} \to \mathrm{F}_{\mathrm{f}}(\mathrm{SE}))$ is given by

$$\mathrm{d}_{\mathrm{f}}(P{:}E) := \Big\langle E,\ \ \lambda\sigma\,.\,[\![P\sigma{:}E\sigma]\!]_{\mathrm{g}},\ \ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E')\,.\,P\sigma{:}E' \Big\rangle.$$

The mapping, $[\![\_]\!]_{\mathrm{f}}$, is the final F_{f}–coalgebra homomorphism from $\langle \mathrm{SE}, \mathrm{d}_{\mathrm{f}}\rangle$ to $\langle \mathrm{J}_{\mathrm{f}}, \mathrm{id}_{\mathrm{J}_{\mathrm{g}}}\rangle$. Theorem 5.1 is the full abstraction theorem. Its proof rests on Theorem 3.2(2), Theorem 4.1 and the fact that the transition from $[\![\_]\!]_{\mathrm{g}}$ to $[\![\_]\!]_{\mathrm{f}}$ is basically the same as the transition from early ground bisimulation to early congruence.

Theorem 5.1 *For all $P{:}E$, $Q{:}E \in \mathrm{SE}$:*

$$P \sim_{\mathrm{e}} Q \text{ if, and only if, } [\![P{:}E]\!]_{\mathrm{f}} = [\![Q{:}E]\!]_{\mathrm{f}}.$$

5.2 Constructor Interpretations

5.2.1 Interpreting Non–Input Constructors

For every γ^-, its interpretation over the full model, γ^-_{f}, is an element of the map space $(\mathrm{J}_{\mathrm{f}}^{\mathrm{ar}(\gamma^-)} \to \mathrm{J}_{\mathrm{f}})$. Informally, the result of applying γ^-_{f} to arguments $\langle E_1, s_1, r_1\rangle, \ldots, \langle E_{\mathrm{ar}(\gamma^-)}, s_{\mathrm{ar}(\gamma^-)}, r_{\mathrm{ar}(\gamma^-)}\rangle$ has the following components:

(i) The names of γ^- plus the union of the free names of all argument objects.

(ii) A function that maps every substitution σ to the result of applying the ground interpretation of $\gamma^-\sigma$, $\gamma^-\sigma_{\mathrm{g}}$, to the vector that ensues from applying the main component of each individual argument to σ. This scheme means that γ^-_{g} is essentially lifted from the ground to the full level in a point–wise fashion.

(iii) A function that describes how the result of applying γ^-_{f} to the argument vector is transformed under renamings. Specifically, all σ, E' with $\sigma \in (E \rightarrowtail E')$ are mapped to the result of applying $\gamma^-\sigma_{\mathrm{f}}$ to the vector that ensues from transforming each individual argument via the restriction of σ to that argument's free names. The target estimate, E', is always the same. If all these subsidiary transformations are defined, then the main transformation is also defined.

Notation. Henceforth, we denote $\gamma^-\sigma$ by $/\gamma^-\sigma/$ for every γ^- and every σ. The extra syntax is intended to help readability throughout this section. We do not write "$(\gamma^-\sigma)$" to avoid that "$(\gamma^-\sigma)$" gets confused with restriction.

As for defining γ^-_{f}, this goal is met in the familiar coalgebraic style by considering all constructs of the form $\underline{\gamma^-_{\mathrm{f}}(\langle E_1, s_1, r_1\rangle, \ldots, \langle E_{\mathrm{ar}(\gamma^-)}, s_{\mathrm{ar}(\gamma^-)}, r_{\mathrm{ar}(\gamma^-)}\rangle)}$, decomposing them exactly as suggested by the recurrence equation and applying the SFC. Because it is so straightforward, we skip that and just state the equation:

$$
\begin{aligned}
(3)\quad & \gamma^-_{\mathrm{f}}(\langle E_1, s_1, r_1\rangle, \ldots, \langle E_{\mathrm{ar}(\gamma^-)}, s_{\mathrm{ar}(\gamma^-)}, r_{\mathrm{ar}(\gamma^-)}\rangle) \\
& := \Big\langle E, \\
& \qquad \lambda\sigma \,.\, /\gamma^-\sigma/_{\mathrm{g}}(s_1(\sigma), \ldots, s_{\mathrm{ar}(\gamma^-)}(\sigma)), \\
& \qquad \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E') \text{ and } \downarrow\varphi''_i \text{ for each } i \in \{1, \ldots, \mathrm{ar}(\gamma^-)\} \,.\, \varphi' \Big\rangle,
\end{aligned}
$$

where:

$$
E := \mathrm{n}(\gamma^-) \cup \textstyle\bigcup_{i \in \{1,\ldots,\mathrm{ar}(\gamma^-)\}} E_i
$$

$$
\gamma^-\sigma := \begin{cases} \gamma^- & \text{if } \gamma^- \text{ is not of the form } \bar{x}y \text{ or of the form } [x = y] \\ \bar{x}\sigma \cdot y\sigma & \text{if } \gamma^- \text{ is of the form } \bar{x}y \\ [x\sigma = y\sigma] & \text{if } \gamma^- \text{ is of the form } [x = y] \end{cases}
$$

$$\varphi' \equiv /\gamma^-\sigma/_{\mathrm{f}}(\varphi''_1, \ldots, \varphi''_{\mathrm{ar}(\gamma^-)})$$
$$\varphi''_i \equiv r_i(\sigma_{|E_i}, E') \text{ for each } i \in \{1, \ldots, \mathrm{ar}(\gamma^-)\}$$

The full interpretation of $(\nu\, x)$, $(\nu\, x)_{E\mathrm{f}}$, is a family of elements of the map space $(\mathrm{J_f} \to \mathrm{J_f})$; here, the fact that the constructor in question is a name binder again requires considering a family of interpretations that is indexed by all name estimates. Analogously to the situation above, stating the appropriate recurrence equation is sufficient to see how $(\nu\, x)_{E\mathrm{f}}$ is defined:

$$\begin{aligned}(\nu\, x)_{E\mathrm{f}}\langle F, s, r\rangle := \Big\langle E,\ & \\ & \lambda\sigma\mathbin{.}\ (\nu\, x')_{E\sigma\ \mathrm{g}} s(\sigma\{x{\mapsto}x'\}), \\ & \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E') \text{ and } {\downarrow}\varphi''\mathbin{.}\ \varphi'\Big\rangle,\end{aligned}$$

where

$$\varphi' \equiv (\nu\, x')_{E'\,\mathrm{f}}\varphi'' \text{ and } \varphi'' \equiv r(\sigma_{|F-x}\{x{\mapsto}x'\}, E'{+}x')$$

with $x' \notin (E{-}x)\sigma$ in both cases. The replacement of x by x' models α–conversion. Also, applying $(\nu\, x)_{E\mathrm{f}}$ to $\langle F, s, r\rangle$ makes practical sense only if $F{-}x \subseteq E$, although $(\nu\, x)_{E\mathrm{f}}$ is total in the technical sense. We refrain from explaining the interpretation in more detail, since the necessity for modelling α–conversion is the only difference to the situation in connection with the interpretation of constructors of the form γ^-.

Finally, the compositionality theorem is as expected:

Theorem 5.2 *(Compositionality) For all* $P_1{:}E_1, \ldots, P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)} \in \mathrm{SE}$*:*

$$\begin{aligned}[\![\gamma^-(P_1, \ldots, P_{\mathrm{ar}(\gamma^-)}){:}\mathrm{n}(\gamma^-) \cup \textstyle\bigcup_{i\in\{1,\ldots,\mathrm{ar}(\gamma^-)\}} E_i]\!]_{\mathrm{f}} & \\ = \gamma^-_{\mathrm{f}}([\![P_1{:}E_1]\!]_{\mathrm{f}}, \ldots, [\![P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)}]\!]_{\mathrm{f}})&\end{aligned}$$

Moreover, for each $P{:}E \in \mathrm{SE}$*:*

$$[\![(\nu\, x)P{:}E]\!]_{\mathrm{f}} = (\nu\, x)_{E\mathrm{f}}[\![P{:}E{+}x]\!]_{\mathrm{f}}$$

Proof. Consider some γ^-, where $k := \mathrm{ar}(\gamma^-)$. Suppose also $P_1{:}E, \ldots, P_k{:}E \in \mathrm{SE}$ with $\mathrm{n}(\gamma^-) \subseteq E$. The main part of the proof consists of showing

$$[\![\gamma^-(P_1, \ldots, P_k){:}E]\!]_{\mathrm{f}} = \gamma^-_{\mathrm{f}}([\![P_1{:}E]\!]_{\mathrm{f}}, \ldots, [\![P_k{:}E]\!]_{\mathrm{f}}). \tag{4}$$

First of all, we rewrite the left hand side of (4). By (2):

$$(5)\qquad [\![\gamma^{-}(P_1,\ldots,P_k){:}E]\!]_{\mathrm{f}} = \Big\langle E,\ \lambda\sigma\,.\,[\![/\gamma^{-}\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E\sigma]\!]_{\mathrm{g}},\ \lambda\sigma,E' \mid \sigma\in(E\rightarrowtail E')\,.\,\varphi' \Big\rangle,$$

where $\varphi' \equiv [\![/\gamma^{-}\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E']\!]_{\mathrm{f}}$. Thus, by Theorem 4.3:

$$(6)\qquad [\![\gamma^{-}(P_1,\ldots,P_k){:}E]\!]_{\mathrm{f}} = \Big\langle E,\ \lambda\sigma\,.\,/\gamma^{-}\sigma/_{\mathrm{g}}([\![P_1\sigma{:}E\sigma]\!]_{\mathrm{g}},\ldots,[\![P_k\sigma{:}E\sigma]\!]_{\mathrm{g}}),\ \lambda\sigma,E' \mid \sigma\in(E\rightarrowtail E')\,.\,\varphi \Big\rangle,$$

where $\varphi' \equiv [\![/\gamma^{-}\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E']\!]_{\mathrm{f}}$.

As for the right–hand side of (4), by (3):

$$(7)\qquad \gamma^{-}_{\mathrm{f}}([\![P_1{:}E]\!]_{\mathrm{f}},\ldots,[\![P_k{:}E]\!]_{\mathrm{f}}) = \Big\langle E,\ \lambda\sigma\,.\,/\gamma^{-}\sigma/_{\mathrm{g}}([\![P_1\sigma{:}E\sigma]\!]_{\mathrm{g}},\ldots,[\![P_k\sigma{:}E\sigma]\!]_{\mathrm{g}}),\ \lambda\sigma,E' \mid \sigma\in(E\rightarrowtail E')\,.\,\varphi' \Big\rangle,$$

where $\varphi' \equiv /\gamma^{-}\sigma/_{\mathrm{f}}([\![P_1\sigma{:}E']\!]_{\mathrm{f}},\ldots,[\![P_k\sigma{:}E']\!]_{\mathrm{f}})$. Note that $[\![P_1\sigma{:}E']\!]_{\mathrm{f}}$ is defined for each $i\in\{1,\ldots,k\}$, so Theorem 5.2 has been applied correctly. At this place, (4) follows from Aczel's solution lemma. Specifically, we introduce a set variable $\underline{[\![Q{:}F]\!]_{\mathrm{f}}}$ for every sound estimate $Q{:}F$. On this basis, we consider the equation system given by

$$(8)\qquad \underline{[\![\gamma^{-}(P_1,\ldots,P_k){:}E]\!]_{\mathrm{f}}} = \Big\langle E,\ \lambda\sigma\,.\,/\gamma^{-}\sigma/_{\mathrm{g}}([\![P_1\sigma{:}E\sigma]\!]_{\mathrm{g}},\ldots,[\![P_k\sigma{:}E\sigma]\!]_{\mathrm{g}}),\ \lambda\sigma,E' \mid \sigma\in(E\rightarrowtail E')\,.\,\varphi' \Big\rangle,$$

where $\varphi' \equiv \underline{[\![/\gamma^{-}\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E']\!]_{\mathrm{f}}}$. Because of (6), the family

$$([\![\gamma^{-}(P_1,\ldots,P_k){:}E]\!]_{\mathrm{f}})_{\gamma^{-}(P_1,\ldots,P_k):E\,\in\,\mathrm{SE}}$$

is the unique solution to this system. At the same time, because of (7), the family

$$(\gamma^{-}_{\mathrm{f}}([\![P_1{:}E]\!]_{\mathrm{f}},\ldots,[\![P_k{:}E]\!]_{\mathrm{f}}))_{\gamma^{-}(P_1,\ldots,P_k):E\,\in\,\mathrm{SE}}$$

is another solution, so

$$[\![\gamma^-(P_1,\ldots,P_k){:}E]\!]_{\mathrm{f}} = \gamma^-_{\mathrm{f}}([\![P_1{:}E]\!]_{\mathrm{f}},\ldots,[\![P_k{:}E]\!]_{\mathrm{f}}).$$

Having shown (4), suppose $P_1{:}E_1, \ldots, P_k{:}E_k \in \mathrm{SE}$ with $E := \mathrm{n}(\gamma^-) \cup \bigcup_{i\in\{1,\ldots,k\}} E_k$. Then (3) entails

$$\begin{aligned}
&\gamma^-_{\mathrm{f}}([\![P_1{:}E_1]\!]_{\mathrm{f}},\ldots,[\![P_k{:}E_k]\!]_{\mathrm{f}}) \\
&\qquad = \Big\langle E, \\
&\qquad\quad \lambda\sigma\mathbin{.}\, /\gamma^-\sigma/_{\mathrm{g}}([\![P_1\sigma{:}E_1\sigma]\!]_{\mathrm{g}},\ldots,[\![P_k\sigma{:}E_k\sigma]\!]_{\mathrm{g}}), \\
&\qquad\quad \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E')\mathbin{.}\, \varphi' \qquad\qquad \Big\rangle,
\end{aligned}$$

where $\varphi' \equiv /\gamma^-\sigma/_{\mathrm{f}}([\![P_1\sigma{:}E']\!]_{\mathrm{f}},\ldots,[\![P_k\sigma{:}E']\!]_{\mathrm{f}})$. Then, by Theorem 4.3 and (4),

$$\begin{aligned}
&\gamma^-_{\mathrm{f}}([\![P_1{:}E_1]\!]_{\mathrm{f}},\ldots,[\![P_k{:}E_k]\!]_{\mathrm{f}}) \\
&\quad = \Big\langle E,\ \ \lambda\sigma\mathbin{.}\, [\![/\gamma^-\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E\sigma]\!]_{\mathrm{g}},\ \ \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E')\mathbin{.}\, \varphi' \Big\rangle,
\end{aligned}$$

where $\varphi' \equiv [\![/\gamma^-\sigma/(P_1\sigma,\ldots,P_k\sigma){:}E']\!]_{\mathrm{f}}$. At this place, the general compositionality property is a direct consequence of (5).

Finally, the compositionality of the interpretation of the restriction constructor can be shown in essentially the same way. □

5.2.2 Interpreting Input Prefixing

Obtaining full interpretations of the π–calculus constructors except input prefixing is essentially a matter of lifting their ground interpretations in a pointwise fashion. Compositionality is then proven with the help of Theorem 4.3 (see Subsection 5.2.1). The compositional interpretation of input prefixing on the full level must be independent of all of that; the reason is that input prefixing does not preserve early ground bisimulation, as this property entails that there is no compositional interpretation of input prefixing on the ground level to build on.

For every prefix $x(y)$, the interpretation of $x(y).$ is a family of elements of the map space $(\mathrm{J}_{\mathrm{f}} \to \mathrm{J}_{\mathrm{f}})$. As in the case of the interpretations of the new–operator, considering a family of interpretations is necessary since $x(y)$ is a name binder. The index set is also the same, that is, it is the set of all name estimates.

The basic idea of defining the action of $x(y)_{E\mathrm{f}}$ on some object $\langle F, s, r\rangle$ from J_{f} is simple: For each $z \in \mathrm{X}$, the free input action xz is put in front of $s(\mathrm{id}_{\mathrm{X}}\{y{\mapsto}z\})$, modelling the effect of input prefixing on the transitional level. The ensuing object, however, must actually be an element of J_{f} and, at the

same time, we want to obtain compositionality. This combination appears to entail a certain amount of inherent complexity. The best way of dealing with it seems to be the divide–and–conquer strategy described in the sequel.

Its cornerstone is an auxiliary class of semantic objects, $\mathrm{J_a}$, which is "in between" $\mathrm{J_g}$ and $\mathrm{J_f}$. The elements of $\mathrm{J_a}$ are triples of the form $\langle E, i, r\rangle$, where r describes the transformations of the triple under injective name replacements. The second component, i, is always a partial function from X to $\mathrm{J_g}$. Technically, $\mathrm{J_a}$ is the largest fixed point of a functor on CLASS, $\mathrm{F_a}$, which is is given by

$$\mathrm{F_a}(C) := \mathrm{Pow(X)} \times (\mathrm{X} \multimap \mathrm{J_g}) \times (((\mathrm{X} \rightarrowtail \mathrm{X}) \times \mathrm{Pow(X)}) \multimap C)$$

for every class C, and by

$$\mathrm{F_a}(f)(\langle E, i, r\rangle) := \langle E, i, r;f\rangle$$

for every map f, $\langle E, i, r\rangle \in \mathrm{F_a}(\mathrm{dom}(f))$. The auxiliary class is ranged over by φ_a.

Next, an auxiliary prefixing operator, x_a with $x_\mathrm{a} \in (\mathrm{J_a} \to \mathrm{J_g})$ for each $x \in \mathrm{X}$, is introduced. The intuition is that x_a acts on triples of the form $\langle E, i, r\rangle$ by putting xz in front of $i(z)$ for each $z \in \mathrm{X}$. Its definition requires an auxiliary class of mixed terms, an auxiliary decomposition and an application of the SFC. The class and the decomposition are exactly as suggested by the recurrence equation below. At the same time, the equation is all what is needed in the sequel. For this reason, we do not state anything else.

$$\begin{aligned} x_\mathrm{a}.\langle E, i, r\rangle = \Big\langle & E, \\ & \{\langle xz, i(z)\rangle \mid z \in \mathrm{X} \text{ and } \downarrow i(z)\}, \\ & \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E') \text{ and } \downarrow r(\sigma, E')\,.\; x\sigma_\mathrm{a}.r(\sigma, E')\Big\rangle. \end{aligned}$$

Note that applying x_a to $\langle E, i, r\rangle$ makes practical sense only if $x \in E$, although x_a is total in the technical sense.

The link between $\mathrm{J_f}$ and $\mathrm{J_a}$ is provided by another auxiliary operator, mrg, with $\mathrm{mrg} \in (\mathrm{X} \times \mathrm{J_f} \times (\mathrm{X} \to \mathrm{X}) \times \mathrm{Pow(X)} \to \mathrm{J_a})$. This one is called the *merge operator*. As a technical preliminary, a function $\mathrm{inc}_{S,T}$ is given by $\mathrm{inc}_{S,T} := \lambda x \mid x \in S \cap T\,.\; x$ for every set S and every set T. The basic intuition behind mrg is that it acts on argument vectors of the form $(y, \langle E, s, r\rangle, \sigma, F)$ by applying s to the restriction of σ to $\mathrm{X}{-}y$. An element of $\mathrm{J_a}$ is mainly formed via a function that maps each $z \in \mathrm{X}$ to

$$s(\sigma\{y{\mapsto}z\})_3(\mathrm{inc}_{(E+y)(\sigma\{y\mapsto z\}),F+z}, F{+}z),$$

given that the application of $s(\sigma\{y{\mapsto}z\})_3$ to the inclusion and the estimate is defined. Intuitively, this construction means that the first compo-

nent of $s(\sigma\{y{\mapsto}z\})$ is replaced by $F{+}z$. In other words, applying mrg to $(y, \langle E, s, r\rangle, \sigma, F)$ makes practical sense only if $(E{-}y)\sigma \subseteq F$. Just like x_{a}, however, mrg is total in the technical sense.

As in the case of x_{a}, stating the recurrence equation is sufficient:

$$\begin{aligned}\mathrm{mrg}(y, \langle E, s, r\rangle, \sigma, F) = \Big\langle & F, \\ & \lambda z \mid \downarrow\theta\mathbf{.}\ \theta, \\ & \lambda\rho, F' \mid \rho \in (F \rightarrowtail F')\mathbf{.}\ \mathrm{mrg}(y, \langle E, s, r\rangle, \sigma\rho, F')\Big\rangle,\end{aligned}$$

where

$$\theta \equiv s(\sigma\{y{\mapsto}z\})_3(\mathrm{inc}_{(E+y)(\sigma\{y\mapsto z\}),F+z}, F{+}z).$$

Eventually, we refrain also from introducing the auxiliary class of mixed terms and the decomposition associated with the interpretation of input prefixing, since they are exactly as suggested by the recurrence equation. This one puts everything together:

$$\begin{aligned}x(y)_{E\mathrm{f}}.\langle F, s, r\rangle = \Big\langle & E, \\ & \lambda\sigma\mathbf{.}\ x\sigma_{\mathrm{a}}.\mathrm{mrg}(y, \langle F, s, r\rangle, \sigma, E\sigma), \\ & \lambda\sigma, E' \mid \sigma \in (E \rightarrowtail E') \text{ and } \downarrow\varphi''\mathbf{.}\ \varphi'\Big\rangle,\end{aligned}$$

where

$$\varphi' \equiv x\sigma\cdot(y')_{E'\mathrm{f}}.\varphi'' \text{ and } \varphi'' \equiv r(\sigma|_{F-y}\{y{\mapsto}y'\}, E'{+}y')$$

with $y' \notin (F{-}y)\sigma$. The replacement of y by y' models α–conversion. Note also that applying $x(y)_{E\mathrm{f}}$ to $\langle F, s, r\rangle$ makes practical sense only if $F{-}y \subseteq E$, although $x(y)_{E\mathrm{f}}$ is total in the technical sense.

Finally, Theorem 5.3 is the compositionality theorem.

Theorem 5.3 *(Compositionality)*

$$[\![x(y)P{:}E]\!]_{\mathrm{f}} = x(y)_{E\mathrm{f}}.[\![P{:}E{+}y]\!]_{\mathrm{f}}$$

Proof. (Idea) Somewhat similar to the proof of Theorem 5.2. Specifically, the main part is to prove

$$[\![(x(y).P)\sigma{:}F]\!]_{\mathrm{g}} = x\sigma_{\mathrm{a}}.\mathrm{mrg}(y, [\![P{:}E{+}y]\!]_{\mathrm{f}}, \sigma, F). \tag{9}$$

The method consists of rewriting both sides by means of recurrence equations until they are close enough for applying Aczel's solution lemma.

Then one can prove the theorem itself. Apart from applying (9), the method is the same as in the case of the main part. □

6 Other Interleaving Models for the π–Calculus

To conclude this paper, we note first that other interleaving models for the π–calculus [8, 11, 14, 28] use domains and categories as their mathematical machinery; hypersets and coalgebras do not play any role. Another key idea of [8, 11, 14, 28] consists of representing transformations under renamings as categorical arrows. This particular aspect contrasts the tupling approach adopted herein, which entails that every semantic object contains a direct representation of the transformations under renamings.

As for constructor interpretations, [8] considers parallel composition, restriction and replication. In [11] all constructors are considered but the interpretation of input prefixing "cannot be purely compositional" [11,page 50 of the proceedings volume], as opposed to the interpretations of the other constructors. In [14] and [28] compositional interpretations of all constructors including input prefixing are provided. These interpretations, however, are respectively carried out over a specific ground model; within this scheme, the interpretation of input prefixing requires a higher–order functionality, so as to cope with the problem that this constructor does not preserve any standard ground equivalence. This approach is akin to what has been done in [17] to interpret input prefixing as a constructor of value passing processes. In the present paper the functionality of the interpretation is a non–higher–order $J_f \to J_f$. This property may be regarded as reflecting that π–calculus name passing is more uniform than value passing, since every name can equally serve as a channel identifier and as an input place holder.

7 Possible Future Work

As for a hint to possible future work, we might reconsider what has been stated at the end of Section 2. At this place, it has been pointed out that early ground bisimulation is a form of strong bisimulation. The first crucial point about that is the idea of observation, which is of fundamental importance within standard process algebra methodology. It leads away from strong bisimulation to so–called weak notions of behavioural equivalence. Classical examples of such notions are observation equivalence (cf. [19]), observation congruence (ibid.), failures equivalence (cf. [15]), testing equivalence (cf. [9, 13]) and branching bisimilarity (cf. [4]).

The second crucial point is that more or less all of these weak equivalences have canonical instantiations within the framework of the π–calculus (cf. [24]). It is therefore important to study coalgebraic π–calculus models that are fully abstract with respect to such notions. Without providing (compositional) constructor interpretations, Honsell et al. and Lenisa have already done that

[16, 18].

Hence, it is left to find coalgebraic π–calculus models that exhibit both full abstraction with respect to some weak notion of behavioural equivalence and compositional constructor interpretations. The analogous program has already been carried out within the framework of CCS–like process algebras [3]. The full abstraction touchstone of this work was observation congruence. Shortly after [3] had been presented, it was even shown that the mixed terms technique [6] can be used in a very similar framework to obtain compositional constructor interpretations. Therefore, the author of the present paper speculates whether future work might consist of incorporating the results from [3] and [6] into what has been presented herein.

References

[1] S. Abramsky. A Domain Equation for Bisimulation. *Information and Computation*, 92:161–218, 1991.

[2] P. Aczel. *Non–well–founded Sets*. Stanford University, 1988.

[3] P. Aczel. Final Universes of Processes. In *Mathematical Foundations of Programming Semantics*, LNCS 802, pages 1–28. Springer–Verlag, 1994. MFPS conference.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[5] J.W. de Bakker and J.I. Zucker. Processes and the Denotational Semantics of Concurrency. *Information and Control*, 54:70–120, 1982.

[6] M. Baldamus. A Non–well–founded Sets Semantics for Observation Congruence over Full CCS. Technical Report 1994/31, computer science department, Berlin University of Technology, 1994.

[7] M. Baldamus. Adding Enrichments to Refined Interleavings: A New Model for the π–Calculus. Technical Report MATH–AL–01–1999, Institute for Algebra, Dresden University of Technology, January 1999.

[8] L. Cattani, G. Winskel, and I. Stark. Presheaf Models for the π–Calculus. In *Category Theory in Computer Science*, LNCS 1290, pages 106–126. Springer–Verlag, 1997. CTCS conference.

[9] R. De Nicola. *Testing Equivalences and Fully Abstract Models for Communicating Processes*. PhD Thesis CST–36–85, Department of Computer Science, The University of Edinburgh, 1985.

[10] G.-L. Ferrari, U. Montanari, and P. Quaglia. The Weak Late π–Calculus Semantics as Observation Equivalence. In *Concurrency Theory*, LNCS 962, pages 57–71. Springer–Verlag, 1995. CONCUR conference.

[11] M. Fiore, E. Moggi, and D. Sangiorgi. A Fully Abstract Model for the π–Calculus. In *Logic in Computer Science*, pages 43–54. IEEE Computer Society Press, 1996. LICS symposium.

[12] M. Forti and F. Honsell. A General Construction of Hyperuniverses. *Theoretical Computer Science*, 156:203–215, 1996.

[13] M. Hennessy. *Algebraic Theory of Processes.* The MIT Press, 1988.

[14] M. Hennessy. A Fully Abstract Denotational Semantics for the π–Calculus. Technical Report 96:04, School of Cognitive and Computing Sciences, University of Sussex, 1996.

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

[16] F. Honsell, M. Lenisa, U. Montanari, and M. Pistore. Final Semantics for the π–Calculus. In *Programming Concepts and Methods*, pages 225–243. Chapman & Hall, 1998. PROCOMET conference.

[17] A. Ingólfsdóttir. *Semantic Models for Communicating Processes with Value–Passing.* Report 8/94, School of Cognitivce and Computing Sciences, University of Sussex, 1994.

[18] M. Lenisa. *Themes in Final Semantics.* PhD thesis, Università di Pisa–Udine, 1998.

[19] R. Milner. *Communication and Concurency.* Prentice Hall International, 1989.

[20] R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[21] R. Milner, J. Parrow, and D. Walker. Modal Logics for Mobile Processes. In *Concurreny Theory*, LNCS 527, pages 45–60. Springer–Verlag, 1991. Concur conference.

[22] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I/II. *Information and Computation*, 100:1–77, 1992. Journal version of a technical report from 1989.

[23] R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Automata, Languages and Programming*, LNCS 623, pages 685–695. Springer–Verlag, 1992. ICALP conference.

[24] P. Quaglia. *The π–Calculus with Explicit Substitutions.* PhD Thesis TD–09/96, University of Pisa–Genova–Udine, 1996.

[25] J. Rutten. Processes as Terms: Non–well–founded Models for Bisimulation. *Mathematical Structures in Computer Science*, 2:257–275, 1992.

[26] J. Rutten and D. Turi. Initial Algebra and Final Coalgebra Semantics for Concurrency. In *A Decade of Concurrency — Reflections and Perspectives*, LNCS 803, pages 530–583. Springer–Verlag, 1994. REX school/symposium.

[27] D. Sangiorgi. *Expressing Mobility in Process Algebras: First–order and Higher–order Paradigms.* PhD Thesis CST–99–93, Department of Computer Science, The University of Edinburgh, 1993.

[28] I. Stark. A Fully Abstract Domain Model for the π–Calculus. In *Logic in Computer Science*, pages 36–42. IEEE Computer Society Press, 1996. LICS symposium.

Appendix: Proving Theorem 4.3

Statement. For all $P_1{:}E_1, \ldots, P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)} \in \mathrm{SE}$:

$$[\![\gamma^-(P_1,\ldots,P_{\mathrm{ar}(\gamma^-)}){:}\mathrm{n}(\gamma^-) \cup \textstyle\bigcup_{i\in\{1,\ldots,\mathrm{ar}(\gamma^-)\}} E_i]\!]_{\mathrm{g}} = \gamma^-_{\mathrm{g}}([\![P_1{:}E_1]\!]_{\mathrm{g}},\ldots,[\![P_{\mathrm{ar}(\gamma^-)}{:}E_{\mathrm{ar}(\gamma^-)}]\!]_{\mathrm{g}})$$

Moreover, for each $P{:}E \in \mathrm{SE}$:

$$[\![(\nu\, x)P{:}E]\!]_{\mathrm{g}} = (\nu\, x)_{E\,\mathrm{g}}[\![P{:}E{+}x]\!]_{\mathrm{g}}$$

The first step of the proof of Theorem 4.3 consists of introducing the notion of CCS–like bisimulation on mixed estimations:

Definition A.1 *A binary relation* $\boldsymbol{R}$ *on* ME *is a* CCS–like simulation *if* $M{:}E\ \boldsymbol{R}\ N{:}E$ *implies: Whenever* $M{:}E \xrightarrow{\mu} M'{:}E'$*, then there exists a* N' *s.t.* $N{:}E \xrightarrow{\mu} N'{:}E'$ *and* $M'{:}E'\ \boldsymbol{R}\ N'{:}E'$*. If both* $\boldsymbol{R}$ *and its inverse are CCS–like simulations, then* $\boldsymbol{R}$ *is a* CCS–like *bi*simulation*. By* $\sim_{\mathrm{CCS}}$*, we denote* CCS–like bisimilarity*, that is, the union of all CCS–like bisimulations.*

Lemmas A.2 to A.5 put in place a number of preliminaries of the main proof. Lemma A.2 pertains to sound estimations of the form $\widehat{P}{:}E$. It states that $\widehat{P}{:}E$ is bisimilar to the mixed estimation that ensues from regarding the outermost constructor of P as a mixed constructor.

Lemma A.2

(i) *If* $\gamma^-(P_1,\ldots,P_{\mathrm{ar}(\gamma^-)}){:}E$ *is sound, then* $\gamma^-(\widehat{P_1,\ldots,P_{\mathrm{ar}(\gamma^-)}}){:}E \sim_{\mathrm{CCS}} \gamma^-(\widehat{P_1},\ldots,\widehat{P_{\mathrm{ar}(\gamma^-)}}){:}E$.

(ii) *If* $(\nu\, x)P{:}E$ *is sound, then* $\widehat{(\nu\, x)P}{:}E \sim_{\mathrm{CCS}} (\nu\, x)\widehat{P}{:}E$.

Proof. (Outline) Part (1) is immediate if γ^- is $\mathbf{0}$ or a dynamic constructor, that is, if γ^- is of the form pre^-, $+$ or $[x=y]$. The static constructors make up both the rest of Part (1) and Part (2). To deal with them, one considers the language of mixed terms over process constants and static constructors, that is, the language given by

$$L ::= \widehat{P} \;\Big|\; L \mid L \;\Big|\; (\nu\, x)L \;\Big|\; !L.$$

For every L, we denote by $\underline{L}$ the result of turning L inductively into a constant. An example of that is $\underline{(\widehat{P_1} \mid \widehat{P_2}) \mid \widehat{P_3}} = \underline{(P_1 \mid P_2) \mid P_3}$. What is left is to prove that the set $\{\langle \underline{L}{:}E, L{:}E\rangle \mid L{:}E \text{ is sound}\}$ is a CCS–like bisimulation. This proof, though tedious, is a straightforward matter of transition induction.

One only commutes applications of **(process)** with applications of **(par)**, **(com)**, **(close)**, **(open)**, **(res)**, **(rep)** and **(alpha)**. That is possible since **(par)**, ... and **(alpha)** as they appear in Table 1 and Table 2 are mutually isomorphic, and since the constantification operator commutes with the free names operator and with α–convertibility. □

In the sequel of this section we restrict our attention to objects reached by $[\![\_]\!]_{\mathrm{g}}$. The reason is that the proof of Theorem 4.3 would become spoiled if we considered semantic objects and mixed terms that are unreasonable in the sense described in Section 4. For the sake of brevity, an object that is reached by $[\![\_]\!]_{\mathrm{g}}$ is called *generated*. The set of all mixed terms over generated objects is ranged over by G and H; the set of all sound estimations of mixed terms over generated objects is denoted by GE.

To proceed, Lemma A.3 states that the restriction of the semantic mapping on mixed estimations to GE is fully abstract wrt. CCS–like bisimulation.

Lemma A.3 *For all* $G{:}E$, $H{:}E \in \mathrm{GE}$, $G{:}E \sim_{\mathrm{CCS}} H{:}E$ *if and only if* $[\![G{:}E]\!] = [\![H{:}E]\!]$.

Proof. See [7]. □

Lemma A.4 states that each element of GE is bisimilar to the estimation that consists of its own interpretation and its own estimate.

Lemma A.4 $G{:}E \sim_{\mathrm{CCS}} [\![G{:}E]\!]{:}E$ *for each* $G{:}E \in \mathrm{GE}$

Proof. A straightforward matter of showing that a specific binary relation on GE is a CCS–like bisimulation. This relation is given by the tuple set $\{\langle G{:}E, [\![G{:}E]\!]{:}E\rangle \mid G{:}E \in \mathrm{GE}\}$. □

Lemma A.5 states that the restriction of $\sim_{\mathrm{GE}}$ is a congruence. On a preliminary note, a binary relation $\boldsymbol{R}$ on GE is a *congruence* if it is preserved by all constructors of mixed terms. We denote this property by $\mathrm{C}(\boldsymbol{R})$. Technically, $\mathrm{C}(\boldsymbol{R})$ is the conjunction of (1) and (2), where:

(i) $G_i{:}E_i \;\boldsymbol{R}\; H_i{:}E_i$ for each $i \in \{1, \ldots, \mathrm{ar}(\gamma^-)\}$ implies $\gamma^-(G_1, \ldots, G_{\mathrm{ar}(\gamma^-)}){:}n(\gamma^-) \cup E \;\boldsymbol{R}\; \gamma^-(H_1, \ldots, H_{\mathrm{ar}(\gamma^-)}){:}n(\gamma^-) \cup E$, where $E := \bigcup_{i \in \{1,\ldots,\mathrm{ar}(\gamma^-)\}} E_i$

(ii) $G{:}E{+}x \;\boldsymbol{R}\; H{:}E{+}x$ implies $(\nu\, x)G{:}E \;\boldsymbol{R}\; (\nu\, x)H{:}E$

Lemma A.5 $\mathrm{C}(\sim_{\mathrm{GE}})$

Proof. By showing that a congruent closure of $\sim_{\mathrm{GE}}$ is a CCS–like bisimulation up to α–conversion (see [7]). □

Proof of Theorem 4.3. We consider only restriction, since dealing with the other non–input constructors is practically the same. The compositionality property in question can be proved as shown below. As a prerequisite, Lemma A.4 entails $G{:}E{+}x \sim_{\mathrm{CCS}} [\![G{:}E{+}x]\!]{:}E{+}x$ whenever $G{:}E{+}x \in \mathrm{GE}$.

Then

$$(\mathrm{A}) \qquad (\nu\, x)G{:}E \sim_{\mathrm{CCS}} (\nu\, x)[\![G{:}E{+}x]\!]{:}E,$$

by taking into account Lemma A.5.

$$\begin{aligned}
[\![(\nu\, x)P{:}E]\!]_{\mathrm{g}} &= [\![\widehat{(\nu\, x)P}{:}E]\!] && ;\ \text{Lemma 4.2}\\
&= [\![(\nu\, x)\widehat{P}{:}E]\!] && ;\ \text{Lemmas A.2 and A.3}\\
&= [\![(\nu\, x)[\![\widehat{P}{:}E{+}x]\!]{:}E]\!] && ;\ \text{(A), Lemma A.3}\\
&= (\nu\, x)_{E\,\mathrm{g}}[\![\widehat{P}{:}E{+}x]\!] && ;\ \text{def. of } (\nu\, x)_{E\,\mathrm{g}}\\
&= (\nu\, x)_{E\,\mathrm{g}}[\![P{:}E{+}x]\!]_{\mathrm{g}} && ;\ \text{Lemma 4.2}
\end{aligned}$$

□