Math. Struct. in Comp. Science (2000), *vol.* 10, *pp.* 617–663. Printed in the United Kingdom

Elementary structures in process theory (1): Sets with renaming

KOHEI HONDA

Department of Computer Science, Queen Mary and Westfield College, University of London, E1 4NS, London, U.K.

Received 7 April 1997; revised 31 May 1999

We study a general algebraic framework that underlies a wide range of computational formalisms that use the notion of names, notably process calculi. The algebraic framework gives a rigorous basis for describing and reasoning about processes semantically, as well as offering new insights into existing constructions. The formal status of the theory is elucidated by introducing its alternative presentation, which is geometric in nature and is based on explicit manipulation of connections among nameless processes. Nameless processes and their relational theory form a coherent universe in their own right, which underlies existing graphical formalisms such as proof nets. We establish the formal equivalence between these two presentations, and illustrate how they can be used complementarily for the precise and effective description of diverse algebras and the dynamics of processes through examples.

1. Introduction

This paper studies a general algebraic framework that underlies a wide range of computational formalisms that use *names*. The notion of names is one of the fundamental elements in formalisms of computing. In logical and functional calculi, variables are used to denote unknown objects (holes) in formulae, and thus for quantification. In imperative programming languages, variables again play an essential role, though one variable may denote different values during computation. In the study of concurrent computing, process calculi (Milner 1980; Hoare 1985; Hennessy 1989; Baeton and Wejland 1990; Milner *et al.* 1989) pervasively use the idea of port names, which represent points of composition and interaction. Here the use of names bears a paramount importance that cannot be compared with the preceding situations, for the following reasons.

First of all, while semantic discussions of functional calculi can be carried out without variables just by taking closed terms, the same idea is far from applicable to process calculi (notice that 'closed terms' in this context, in the sense that no names occur free, are effectively equivalent to inactions in many behavioural equivalences), indicating that the semantics of processes crucially depends on names. Indeed, the idea of interacting processes itself leads to the notion of *points of interaction*, and it is names in processes that represent these entities. We also observe that the use of names is closely related to one of the basic aspects of concurrent computation, that is, non-determinism, in at

least two ways. First, names can represent the sharing of interaction points, which is a common phenomenon in real-life computing. Secondly, names serve to maintain identity despite possible changes of meaning during computation. A bank account yesterday and today might be two different objects with different behaviours, but are still considered to be the same account, because we can trace its identity using its name. Thus, any serious semantic discussion of concurrent computing may not be able to neglect the functionalities of names. Notice that these two points pertain not only to process calculi, but also to general computational phenomena that involve sharing or non-determinism. We may also note that, even though the semantics of sequential computation can sometimes dispense with names (in the form of variables), their use is pervasive in diverse functional calculi and programming languages, playing a key role for their tractable formalisation. Moreover a few recent embedding results of various calculi and languages into π-calculus (Milner *et al.* 1989; Milner 1990; Milner 1992; Sangiorgi 1992; Walker 1994) suggest that the manipulation of names is a basic, if so far hidden, mechanism even in functional computation.

The preceding discussion indicates that names are one of the most essential ingredients of theories of processes. Indeed, we use them when describing processes, we use them when reasoning about their behaviours, and we use them when we define behavioural equivalences. If names are thus important for processes, it would be useful to clarify the precise formal status of names in processes in a general setting. More concretely, we wish to articulate a common abstract stratum that underlies the usage of names in diverse name-based formalisms. If such a common stratum can be identified and its nature becomes clarified, the resulting theory would serve as a guiding principle for the description of, and for reasoning about, interacting processes at both syntactic and semantic levels, offering new insights into existing and potential constructions. This, then, is our aim in the present paper: to articulate the common abstract structures of processes with names in diverse process formalisms. To this end, we introduce a simple algebraic framework where each element in a collection is equipped with a set of free names, just as in the usual process calculi, and the renaming operations relate these elements. Despite its simplicity, the construction induces a non-trivial algebraic universe that offers a new insight into, and a rigorous basis for, the algebraic manipulation of processes. The formal status of the theory is further clarified by an alternative presentation, which is derived from an analysis of fine structures of the original theory. The alternative presentation is based on processes without names, but with the notion of explicit connection. Processes now arise as geometric objects, and such notions as reduction, transition and behavioural equivalences are given as certain geometric relationship between these objects. While this feature makes the theory less amenable to syntactic description, it leads to a mathematically lean construction in which the geometric aspect of processes is lucidly represented. It is notable that existing graphical formalisms such as proof nets and interaction nets (Girard 1987; Lafont 1990; Lafont 1995) offer significant instances of this theory. Nameless processes induce a self-contained set-like universe when equipped with analogues of relations and functions on them (which correspond to relations and homomorphisms in the name-based theory), offering an elucidation of the algebraic status of basic notions in process theories, such as operations and equations on interacting processes.

As a main technical result, we establish the formal equivalence between these two presentations, which offers a firm footing to move between a name-based formalism, such as π-calculus (Milner *et al.* 1989), and a nameless formalism, such as proof nets (Girard 1987; Lafont 1995), without losing any information, as well as providing a common technical basis for their understanding. Two different presentations offer different and complementary roles in the study of interaction-based computation: the name-based presentation offers a tractable and precise way for describing and reasoning about processes, which would, in particular, be suited for algebraic manipulation; while the nameless presentation would shed light on geometric nature of various constructions, which may not be apparent from the name-based presentation, and offers a rigorous framework to articulate graphical description of processes. We exhibit these complementary uses, which are ensured to commute with each other by the equivalence theorem, by applying the theory to a simple but non-trivial problem: the comparison of the expressive power of open process connection (as in CCS) and those based on closed connection (as in proof nets). We show, in a general setting, that these two are faithfully embeddable in each other. In this and other examples, we extensively use various notions of wires (which are agents that function as links between different locations), for whose description and manipulation the distinct roles of two presentations mentioned above are particularly apparent. In this way the notion of processes with names is rigorously positioned in two equivalent presentations; and distinct uses of these presentations are illustrated through examples. It is hoped that the constructions and insights obtained will be useful for further study into the theory of processes.

One basic reservation concerning the present theory should be noted. Names not only appear in formal systems but also in real computing. We then find that, in reality, names are often hierarchically organised, the relationship between names and their referents may not always be unique and fixed, and names may be carried as data and lost during communication. These subtle aspects of names, which clearly deserve close theoretical scrutiny, are not addressed in the present theory: the notion of names we shall treat is restricted to the most basic one, even though it does encompass the usage of names in so-called name passing calculi. Our belief in this context is that the present theory may serve as a useful cornerstone even for the study of those aspects of names that go beyond what the present framework addresses, at least for the purpose of articulating points of departures from, and similarities with, the simpler setting that the present theory addresses.

There have been many attempts to use category theory to represent semantically processes and their algebra, using both name-based and name-less presentations. We may even observe that an analogue of the translation from the name-based presentation to the nameless presentation is already found in the well-known practice of 'forgetting free variables' in the categorical semantics of functional calculi (for example, $x : \alpha \vdash x : \alpha$ and $y : \alpha \vdash y : \alpha$ become the same identity arrow even if x and y occur free). One may thus wonder whether the use of some suitable categories would serve our present purpose. The answer to this question is not simple: we may first observe that, while it is possible to formalise semantics of processes using categories in some cases, it may not be obvious that the semantics of existing and potential formalisms of processes would always be

cleanly representable in a categorical format in a way functional calculi found their basic semantic description in suitable categories, *cf.* Asperti and Longo (1991). In particular, we suspect that subtle notions of typed composition in concurrent processes may not always conform to categorical structures, unlike the case of typed function calculi. In contrast, the present study offers a framework in which we can directly capture the algebra and dynamics of processes as found in process calculi, as well as in graphical formalisms such as proof nets. At the same time, category theory offers rich algebraic tools for organising semantic constructions, which would be useful both for a direct description of processes (as in certain symmetric monoidal structures) and for meta-theoretical treatment (as we ourselves will use it in our equivalence result). We believe that the present work will be able to interact fruitfully with various categorical constructions for the semantics of concurrency, complementing existing approaches and offering new insights.

Related work

One of the key elements of the present theory is to treat processes as structured objects and build a relational theory based on them. An early precursor in this line of studies is the theory of composita (Nerode 1959; Aczel 1990; Aczel 1991), which studies structured objects with operations including substitution. Their theory differs in that they treat variables for structured objects rather than names, and that a variable-less presentation is not studied (hence there is no analogue of the equivalence theorem).

In the context of process theories, action structures (Milner 1993; Milner 1995a; Mifsud 1996; Mifsud *et al.* 1996; Gardner 1995) present and are used to study a common algebraic framework for diverse calculi of computation based on names and connections, which has been influential on the present work. There are a number of topics treated in this context that are closely related to the present work. As a general comparison, the aim of the present work lies in the articulation of a common abstract structure of names and processes that are common in varied formalisms and calculi, while action structures are aimed at offering a general setting and tools by which both statics and dynamics can be comprehensively described. A few technical aspects that relate directly to the present work are discussed in Section 7, where we shall also give comparisons between the present work with other general frameworks for the semantics of concurrency.

Another closely related area, interaction-based computation based on graphical presentation, has been extensively studied in the context of Linear Logic, *cf.* Girard (1987), Girard (1989), Girard *et al.* (1995), Lafont (1990) and Lafont (1995), and has given an important impetus to the present work. As we have noted, these formalisms arise as non-trivial instances of the nameless presentation of the present theory. In this context, our theory offers a rigorous account of basic features of their constructions, including algebra and dynamics, thus positioning them in a wider context, which may eventually lead to their exploitation in wider settings than those for which they were originally developed. Further discussion of related studies can be found in Section 7.

Outline of the paper

Section 2 introduces the name-based theory. Section 3 studies a fine structure of the name-based processes that eventually leads to its nameless counterpart. Section 4 develops the theory of nameless processes. Section 5 establishes the main result of the paper, the categorical equivalence between the two theories. Section 6 presents a small but non-trivial application of the present theory. Section 7 discusses further issues and gives comparisons with related work. Supplementary material is relegated to appendices for readability.

2. Processes with names (1): basic theory

2.1. *Rooted process sets*

The idea of processes is inherently associated with the notion of names. Let us take a familiar process calculus, such as CCS, CSP or ACP. We start from a set of terms, say $\mathscr{P}$, on which we have a few operations such as parallel composition and hiding, and a notion of dynamics is given by a reduction relation or a labelled transition relation over $\mathscr{P}$. On top of these constructions we stipulate an equivalence over terms based on some notion of interactive behaviour. Here each process (given as a term) owns a set of free names, and all involved operations and relations are closed under injective renaming: so is the reduction, so is the parallel composition, and so are the behavioural equivalences such as bisimilarity. The following definition extracts this common trait into a simple algebraic scheme. Below we write $\mathsf{Fin}(X)$ for the set of finite subsets of X.

Definition 2.1. (Rooted p-sets) Given a countable set $\mathscr{N}$ of names, ranged over by $a, b, c, \ldots$, an *$\mathscr{N}$-rooted process set*, which is often just called a *rooted p-set*, is given by a set $\mathscr{P}$ of *rooted processes*, ranged over by $P, Q, R, \ldots$, together with the following data:

(i) An operation $[\sigma] : \mathscr{P} \to \mathscr{P}$, written postfix, for each injection σ over $\mathscr{N}$. $[\sigma]$ is called a *name permutation by σ*, or *renaming by σ*.
(ii) A function $\mathscr{FN} : \mathscr{P} \to \mathsf{Fin}(\mathscr{N})$. $\mathscr{FN}(P)$ is called the *free names* of P.
(iii) Equations:
   1. $P[\sigma][\sigma'] = P[\sigma' \circ \sigma]$ (here $\circ$ is the functional composition).
   2. If $(\forall a \in \mathscr{FN}(P).\sigma(a) = a)$, then $P[\sigma] = P$.
   3. $\mathscr{FN}(P[\sigma]) = \sigma(\mathscr{FN}(P))$.

Given a rooted p-set, we write $P \overset{\mathscr{N}}{\sim} Q$ when $P = Q[\sigma]$ for some σ. Immediately, $\overset{\mathscr{N}}{\sim}$ is an equivalence relation. Hereafter we fix some $\mathscr{N}$. A rooted process is often denoted by the underlying set of processes, provided no confusion arises.

Note that $P[id_{\mathscr{N}}] = P$ by (iii)-2. A rooted process P denotes an entity having several interface points, which are represented by (or *rooted to*) its free names. The injective renaming acts on processes and relate them by name correspondence. The free name function makes defnite the range of names that a process owns and which, therefore, bijective renaming has real effects on.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

Remarks 2.2. The actual significance of setting $\mathscr{FN}(P)$ to be finite lies in making it of strictly less cardinality than $\mathcal{N}$, so that the complement of $\mathscr{FN}(P)$ in $\mathcal{N}$ has again the same cardinality as $\mathcal{N}$. Under this condition, we can set $\mathscr{FN}(P)$ to be infinite and all the main results in this paper hold using essentially the same proof method.

We now offer a few examples of rooted p-sets.

Examples 2.3. (Rooted p-sets)

(i) The simplest rooted p-set is the empty set. A singleton set is also a rooted p-set, whose unique process has no free names, so each renaming acts as identity. In this way any set can be considered as being a rooted p-set.

(ii) A simple, but important, example is the set $\mathcal{N}$ of names itself. Each name is regarded as a rooted process with $\mathscr{FN}(a) = \{a\}$, and the renaming operation is just taken literally. Note there is only one $\overset{\mathcal{N}}{\sim}$-equivalence class in this case. A similar example is $\mathsf{Fin}(\mathcal{N})$, the set of finite subsets of $\mathcal{N}$, which has a countable number of $\overset{\mathcal{N}}{\sim}$-equivalence classes, one for each finite cardinal. Notice that any permutation on $A \in \mathsf{Fin}(\mathcal{N})$ leaves A unchanged.

(iii) Diverse process calculi induce such structures: we take terms as rooted processes (modulo α-convertibility if there is a binding; or even modulo a so-called structural congruence), augmented with $[\sigma]$ and $\mathscr{FN}(P)$ defined in a natural way (if names and co-names are present as in CCS, it is best to consider, say, $a.P$ and $\overline{a}.Q$, as being distinct constructors with the same name a). Then they form a rooted p-set. Other important examples are various typed process calculi, where we regard $\Gamma \vdash P$ as a rooted process whose free names are those of Γ. Furthermore, typed and untyped λ-calculi can be regarded as inducing such structures where free variables are regarded as free names. Later we show how synchronisation trees (Milner 1980; Winskel 1984), a basic semantic representation of processes, also induce rooted p-set.

We note some immediate consequences of Definition 2.1. The stated logical equivalences will often be used implicitly from now on. In the proof below and henceforth, we often write, for example, $\binom{abc}{cba}$ to denote a permutation on $\mathcal{N}$, understanding it acts as identity on names that are not mentioned.

Proposition 2.4. Assume $A \subset \mathcal{N}$. Then the following three conditions are equivalent.

(i) $A \supset \mathscr{FN}(P)$.
(ii) $\forall a \in A.\sigma(a) = a$ implies $P[\sigma] = P$.
(iii) $\sigma_1 \upharpoonright A = \sigma_2 \upharpoonright A$ implies $P[\sigma_1] = P[\sigma_2]$.

Proof. (i)⇒(ii) is immediate. If (ii) holds, $\sigma_1 \upharpoonright A = \sigma_2 \upharpoonright A$ implies $\forall a \in A.\ \sigma_2^{-1} \circ \sigma_1(a) = a$, hence $P[\sigma_1][\sigma_2^{-1}] = P$, that is, $P[\sigma_1] = P[\sigma_2]$, thus (iii) holds. If (iii) holds but (i) does not, take any $a \in \mathscr{FN}(P) \backslash A$ and e fresh, then $\binom{ae}{ea} \upharpoonright A = id_{\mathcal{N}} \upharpoonright A$, hence $P[\binom{ae}{ea}] = P$, which is a contradiction. □

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

2.2. *Algebra of rooted process sets*

As with any study of algebraic structures, our study of rooted p-sets start from relations and maps over them.

Definition 2.5. (compatible relations and homomorphisms)

(i) A *compatible relation* $\mathscr{R}$ from $\mathscr{P}$ to $\mathscr{P}'$ is a relation from $\mathscr{P}$ to $\mathscr{P}'$ such that, for each renaming σ, $P\mathscr{R}Q \Rightarrow P[\sigma]\mathscr{R}Q[\sigma]$

(ii) A *homomorphism* is a compatible relation that is also a function. *Isomorphisms* are bijective homomorphisms. We write $\mathscr{P} \simeq \mathscr{P}'$ if there is an isomorphism between $\mathscr{P}$ and $\mathscr{P}'$.

Proposition 2.6. A compatible relation $\mathscr{R}$ is a homomorphism iff $\mathscr{R}$ is total on its domain and, moreover, $P\mathscr{R}Q$ and $P[\sigma]\mathscr{R}Q'$ imply $Q' = Q[\sigma]$.

Proof. Both directions are immediate from the definition. □

Note that the compatibility only requires closure under renamings, without mentioning the free name function. Relations and maps closed under name permutations are ubiquitous in process theories. This is because, intuitively speaking, the essential use of names in relating processes lies in the *specification of structural correspondence on interfaces between processes*. Take an example of CCS, $a.b.0 + b.a.0 \sim a.0|b.0$, with $\sim$ being the strong bisimilarity. Here a and b are significant precisely because they specify how these two processes are structurally related. So *provided the same correspondence is maintained, we can permute names freely*, for example $l.m.0 + m.l.0 \sim l.0|m.0$ again. We may also note that $\sim$ is *not* closed under non-bijective renaming (which is often true in other process calculi (Milner *et al.* 1989; Boreale and Sangiorgi 1996)). This suggests that, for our framework to be inclusive enough, it will be better to start from injective renaming.

We write $\mathscr{R}_1\mathscr{R}_2$ or $\mathscr{R}_2 \circ \mathscr{R}_1$ for the standard relational composition of two compatible relations. It is easy to see that homomorphisms and compatible relations are each closed under the composition (and that, in addition, the union, intersection and difference of two compatible relations are again compatible). We thus obtain categories, which may be considered as general universes where theories of name-based processes are carried out.

Definition 2.7. RPS denotes the category of rooted process sets and homomorphisms, while RPS_{rel} denotes the category of rooted process sets and compatible relations.

As discussed in Appendix A, RPS is a topos (a category-theoretic generalisation of the category of sets), so it is equipped with all standard set-theoretic operations, including relational manipulations and algebras. Moreover, the standard method of deriving a relational universe from a topos gives us precisely RPS_{rel}, clarifying a basic categorical status of these universes.

We proceed with our study of concrete constructions on rooted processes. We first note

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

the following fact, which shows that, while Definition 2.5 does not mention the free name function when defining a homomorphism, we can recover the information *a posteriori*.

Proposition 2.8. Given a homomorphism $\mathscr{F}$, suppose Q is in its image. Then $\mathscr{FN}(Q) = \bigcap_{Q=\mathscr{F}(P')} \mathscr{FN}(P')$. In particular, if $\mathscr{F}$ is an isomorphism, $\mathscr{FN}(P) = \mathscr{FN}(\mathscr{F}(P))$.

Proof. Suppose $Q = \mathscr{F}(P)$. If $a \in \mathscr{FN}(Q) \backslash \mathscr{FN}(P)$, then $\mathscr{FN}(Q[\binom{af}{fa}]) \neq \mathscr{FN}(Q)$ (f fresh) by Definition 2.1, but $Q[\binom{af}{fa}] = \mathscr{F}(P[\binom{af}{fa}]) = \mathscr{F}(P) = Q$, which is a contradiction. But if $a \in \mathscr{FN}(P) \backslash \mathscr{FN}(Q)$, then $\mathscr{F}(P[\binom{af}{fa}]) = Q[\binom{af}{fa}] = Q$ (f fresh) with $a \notin P[\binom{af}{fa}]$, and hence we are done. □

The above result shows that simply having the closure under renaming nonetheless has a non-trivial algebraic outcome. It also shows that isomorphisms determine not only how a renaming operates on processes, but also the free names processes own (thus in effect determining another essential internal structure of rooted processes called symmetries, see Section 3 later). So, in most imaginable circumstances, treating rooted process sets modulo isomorphisms is enough; and it is technically convenient that this is automatically ensured by having only closure under renaming.

Next we show how products, relations and quotients of rooted p-sets can again be considered as rooted p-sets. These closure properties are the basis for the algebraic manipulation of rooted p-sets. Appendix A shows how they correspond to standard category-theoretic constructions, which in particular means they are determined up to isomorphisms, once given their defining properties.

Definition 2.9.

(i) (Product) Given $\mathscr{P}$ and $\mathscr{Q}$, the *product* of $\mathscr{P}$ and $\mathscr{Q}$, written $\mathscr{P} \times \mathscr{Q}$, is the set-theoretic product of $\mathscr{P} \times \mathscr{Q}$, on which we define: (a) $\langle P,\ Q\rangle[\sigma] \stackrel{\text{def}}{=} \langle P[\sigma],\ Q[\sigma]\rangle$, and (b) $\mathscr{FN}(\langle P,\ Q\rangle) = \mathscr{FN}(P) \cup \mathscr{FN}(Q)$. Furthermore, for an arbitary family $\{\mathscr{P}_i\}_{i\in I}$, their product $\prod_{i\in I} \mathscr{P}_i$ is given by $\{\langle P_i\rangle_{i\in I} | P_i \in \mathscr{P}_\rangle, \bigcup_i \mathscr{FN}(P_i) \in \mathsf{Fin}(\mathscr{N})\}$, with $\mathscr{FN}(\langle P_i\rangle \stackrel{\text{def}}{=} \bigcup \mathscr{FN}(P_i))$ and $\langle P_i\rangle\rho \stackrel{\text{def}}{=} \langle P_i\rho\rangle$.

(ii) (Relation) Any compatible relation can be viewed as a rooted p-set precisely as (i) above.

(iii) (Quotient) Given any congruence (that is, compatible and equivalence) relation on $\mathscr{P}$, we construct $\mathscr{P}/\sim$, the *quotient by* $\sim$, as the set theoretic quotient with operations: (a) $[P]_\sim[\sigma] \stackrel{\text{def}}{=} [P\sigma]_\sim$ and (b) $\mathscr{FN}([P]_\sim) \stackrel{\text{def}}{=} \bigcap_{P'\sim P} \mathscr{FN}(P')$. A *natural map* associated with $\sim$ on $\mathscr{P}$ is a surjective homomorphism from $\mathscr{P}$ to $\mathscr{P}/\sim$ defined in the standard way, which always exists.

In addition, union and intersection of rooted p-sets are the obvious set-theoretic operations, inheriting the free names and the renaming from the original rooted p-sets. We observe that each operation in process algebras, such as prefix and parallel composition,

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

is nothing but a homomorphism from a product rooted p-set to a rooted p-set, so they follow the construction in (i)[†].

Further examples of the constructions in Definition 2.9 are given in the next subsection.

2.3. *Examples*

The first example shows how the quotient in Definition 2.9 indeed underlies the well-known semantic construction in process theories.

Examples 2.10. (Synchronisation trees) We assume a basic knowledge of CCS (Milner 1989). A *synchronisation tree* representing a CCS process is a tree whose edges are labelled from CCS actions. We restrict our attention to those synchronisation trees where two occurrences of the same action from each node never lead to strongly bisimilar trees as derivatives. Then each equivalence class of CCS terms with respect to the strong bisimilarity on terms (which we write $\sim$) can be assigned a unique corresponding synchronisation tree. Note that CCS terms form a rooted p-set, while $\sim$ is a compatible equivalence relation on them. We now observe:

(i) (Quotient) Let T be a synchronisation tree corresponding to $[P]_\sim$. Writing s for a series of actions, we have $T \stackrel{s}{\longrightarrow}$ iff $\exists Q \in [P]_\sim.\ Q \stackrel{s}{\longrightarrow}$. Taking $\mathscr{FN}(T) = \{a \mid T \stackrel{sa}{\longrightarrow} \wedge\ a \in \mathscr{FN}(\alpha)\}$ ($\mathscr{FN}(\alpha)$ is taken syntactically), we immediately know $\mathscr{FN}(Q) \supset \mathscr{FN}(T)$ and, for any name $a \in \mathscr{FN}(Q) \setminus \mathscr{FN}(T)$, we can find $P' \sim P$ such that $a \notin \mathscr{FN}(P')$ (which is proved by induction on the structure of terms). This shows $\mathscr{FN}(T) = \cap_{P' \sim P} \mathscr{FN}(P')$, which conforms to the description in Definition 2.9 (iii). Thus these synchronisation trees give a concrete presentation of the quotient of CCS terms modulo $\sim$.

(ii) (Homomorphism) The semantic function from CCS-terms to the set of synchronisation trees gives the natural map associated with $\sim$ in the sense of Definition 2.9 (iii).

(iii) (Transition relation as a compatible relation) Let us analyse the transitions in a synchronisation tree in more detail. They induce a transition relation $T \stackrel{\alpha}{\longrightarrow} T'$, and, observing the set of labels induce a rooted p-set, gives a ternary compatible relation. Note that whenever $T \stackrel{\alpha}{\longrightarrow} T'$ we have $\mathscr{FN}(\alpha) \subset \mathscr{FN}(T) \supset \mathscr{FN}(T')$.

Similarly, synchronisation trees for other process calculi give examples of the constructions in Definition 2.9, with possible adaptations as necessary. One interesting case is when name passing is involved, as in π-calculus (Milner *et al.* 1989; Milner 1992). Here each node is explicitly equipped with free names so that they always contain free names in their immediate actions but are disjoint with 'new' names, and the free names of its derivative is given by the original names plus those newly introduced (if any) by the action, the latter binding their subsequent occurrences. This gives a coherent way of capturing the behaviour

[†] During the proof-reading of the present manuscript, we learned about Fraenkel–Mostowski's work in which a set-theoretic universe is formed analogously to Godel's constructible sets, albeit starting from infinite atoms (with a powerset operation along the line of arbitrary products in Definition 2.9 (i) above). It is notable that the above operations directly correspond to set-theoretic operations in their universe, as well as to categorical constructions given in Appendix A. See Section 7.2 for further discussions. I would like to thank Andrew Pitts for communicating to me about the work by Fraenkel and Mostowski.

of name passing processes. As a further ramification, some other behavioural equivalences that give rise to concrete semantic representations, *cf.* Hennessy (1989), also conform to the present notion of quotient. Furthermore, the construction in Proposition 2.9 (iii) is also applicable when we do not (yet) know a concrete semantic representation of the quotient by a given behavioural equivalence, as in so-called reduction-based behavioural equivalences (Milner and Sangiorgi 1992; Honda and Yoshida 1993), so that quotients are equipped with more structures than simple set-theoretic quotients.

Next we present another basic example, to which we shall often refer later. It is a basic form of π-calculus, originally given in the context of action structures (Milner 1995a). The example is also interesting in that it indicates how the notion of non-injective substitution can be seamlessly incorporated into the present framework.

Examples 2.11. (Essential π-calculus) Define the set of terms by

$$P \ ::= \ \overline{a}b \ \mid \ ab \ \mid \ P|Q \ \mid \ (va)P \ \mid \ [a=b] \ \mid \ [a] \ \mid \ 0$$

where in each of the first, second and fifth expressions we assume $a \neq b$. We call $[a=b]$ a *wire*, while $[a]$ is a *degenerate wire*. In the standard way, we assume $(va)P$ induces a binding, and the notion of α-conversion $\equiv_\alpha$ is defined accordingly. The set of terms, always considered modulo $\equiv_\alpha$, clearly gives a rooted process set, taking free names syntactically. We then quotient the terms by the congruence generated by the following rules:

(i) $P \equiv Q$ when $P \equiv_\alpha Q$.
(ii) $P|0 \equiv P$, $P|Q \equiv Q|P$, $(P|Q)|R \equiv P|(Q|R)$.
(iii) $(va)P \equiv P$ if $a \notin \mathscr{FN}(P)$, $(vab)P \equiv (vba)P$, and $(va)P|Q \equiv (va)(P|Q)$ when $a \notin \mathscr{FN}(Q)$.
(iv) $[a=b] \equiv [b=a]$, $[a=b]|[a=b] \equiv [a=b]$,
$[a=b]|P \equiv [a=b]|(P[\binom{ab}{ba}])$, $(va)[a=b] \equiv [b]$, $(va)[a] \equiv 0$.
(v) $[a]|P \equiv P$ $a \in \mathscr{FN}(P)$.

Note the use of renaming in (iv). The induced equality is closed under renaming and two operations: thus the quotient again assumes the rooted process set with two renaming-closed operations inherited from the original algebra. The dynamics is given by the reduction relation on the quotient, closing the following rule by two operations:

$$ab \,|\, \overline{a}c \ \rightarrow [b=c]. \tag{1}$$

Thus a receptor ab and a message $\overline{a}c$ interact to generate an open substitution (note that this can be applied to the case when $b=c$ by setting ab above as $(ve)(ae|[e=c])$, resulting in $(ve)([e=c]|[e=c]) \equiv [c])$. Clearly the relation is again renaming-closed. We can further consider a reduction-based behavioural congruence based on input/output predicates in the manner of Milner and Sangiorgi (1992) and Honda and Yoshida (1993). The resulting equality is again closed under renaming and induces a quotient algebra. Furthermore, if we assume reduction-closure in the sense of Honda and Yoshida (1993), the reduction relation is also carried over to this algebra.

Now write $P\{b/a\}$ for $(va)(P|[a=b])$, and $a(x)P$ for $(vx)(ax|P)$. Then we easily deduce that Rule (1) gives us the following reduction:

$$a(x)P \ | \ \overline{a}b \ \rightarrow \ P\{b/x\} \tag{2}$$

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

capturing the usual notion of reduction in π-calculus (except P above can interact with the outside even before substitution, which is because we lack the synchronisation machinery given by prefix (Milner 1980; Girard 1987; Milner 1995a), but this can be introduced easily by adding prefix operators for input and output without binding). Note the crucial use of a wire that represents name substitution: it is notable that, as reported in Honda and Yoshida (1993), such wires are indeed semantically existent in a fully expressive fragment of the π-calculus. We shall see in Section 6 that the nameless version of the essential π-calculus can be derived from the above construction, illuminating the geometric content of the original construction.

3. Processes with names (2): symmetries

3.1. *Symmetry presentation of rooted process sets*

In this subsection we study a fine structure of rooted processes induced by renaming, which will eventually lead to a name-less presentation of processes. The structure is called symmetries, and its definition is simply given as follows (we fix some rooted process set).

Definition 3.1. (Symmetries) A *symmetry of* P is a permutation, say σ, of $\mathscr{FN}(P)$ that, seen as a permutation over $\mathscr{N}$ by extension, is such that $P[\sigma] = P$. The set of symmetries of P and their functional composition form a group, denoted $sym(P)$.

Symmetries arise inevitably in processes.

Examples 3.2. (Symmetries)

(i) In $\varnothing$ there is no process so there is no symmetry. In $\mathscr{N}$ each process has only one name so there is only a trivial symmetry (that is, the identity). In $\mathsf{Fin}(\mathscr{P})$, however, each process has full symmetries, that is, for each 'process' $A \in \mathsf{Fin}(\mathscr{P})$, the set of all permutations on A is the symmetries of A.

(ii) In CCS, a permutation $\binom{ab}{ba}$ is a symmetry of a process '$a.0|b.0$', seen modulo strong bisimilarity. Similarly, '$a(x).(\bar{b}x|\bar{c}x)$' in π-calculus modulo structural equality has a symmetry $\binom{abc}{acb}$. As can be seen, symmetry in processes typically emerges when an agent contains a subterm of the form $P|(P\sigma)$ modulo commutative monoid law (Berry and Boudol 1990). We also note that, by considering terms modulo various equivalences, we may gain more symmetries: for example '$a.0|\tau.b.0$' owns a non-trivial symmetry up to the weak bisimilarity. Such symmetries will be called *abstract symmetry*, and can be considered as a semantic property of a process. In contrast, λ-terms (viewed as processes with variables as names) only own trivial symmetry under any standard equalities; this comes from the non-commutative nature of applicative structure.

(iii) Symmetries are also used in Milner (1989, Section 4) to reduce the size of a bisimulation. Another example of their use in proving properties of processes is a separation result due to Palamidessi (Palamidessi 1997), in which the invariance of symmetries under the progression of dynamics is used to prove that mobile processes with mixed sums cannot be embedded into those without sums. These examples show how symmetries can play a useful role in proving semantic properties of processes.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

By combining the abstract symmetries we noted in (ii) above and possible ramifications of Palamidessi's development, we may consider a general use of symmetries to calculate the semantic difference between two processes. By showing one process behaviourally (say at the transition level) does not own certain symmetries that the other process has, we can immediately establish they are semantically incompatible. It would be an interesting subject of study whether there is some method by which such a calculation can be done mechanically.

The following results show that symmetries also give basic insights into the algebraic constructions we have already presented.

Proposition 3.3.

(i) Let $\mathscr{R}$ be a compatible relation seen as a rooted process set. Suppose $\langle P,\ Q\rangle \in \mathscr{R}$. Then $sym(\langle P,\ Q\rangle) = \{\sigma \upharpoonright \mathscr{FN}(\langle P,\ Q\rangle) \mid \sigma(P) = P \ \wedge\ \sigma(Q) = Q\}$.

(ii) Let $\sim$ be a congruence. Then the symmetries of a rooted process $[P]_\sim \in \mathscr{P}/\sim$ are given by $sym([P]_\sim) \stackrel{\text{def}}{=} \{\sigma \upharpoonright \mathscr{FN}([P]_\sim) \mid \sigma(P) \sim P\}$.

(iii) Let $\mathscr{F}$ be a homomorphism. Then $Q = \mathscr{F}(P)$ and $\rho \in sym(P)$ implies $\rho \upharpoonright \mathscr{FN}(Q) \in sym(Q)$. If, moreover, $\mathscr{F}$ is an isomorphism, we have $sym(P) = sym(Q)$.

Proof. (i) and (ii) are easy. For (iii), $\mathscr{FN}(Q) \subset \mathscr{FN}(P)$ by Proposition 2.8. If $\rho \in sym(P)$, $Q[\rho] = \mathscr{F}(P[\rho]) = \mathscr{F}(P) = Q$, hence (formally by Lemma 3.8 later) $\rho \upharpoonright \mathscr{FN}(Q) \in Q$. If $\mathscr{F}$ is an isomorphism, $\mathscr{FN}(P) = \mathscr{FN}(Q)$, hence $sym(P) \supset sym(Q)$. Taking $\mathscr{F}^{-1}$ gives the converse. □

Note that:

— In (i), a symmetry of $\langle P,\ Q\rangle$ should come from one symmetry of P and another of Q whose values coincide at $\mathscr{FN}(P) \cap \mathscr{FN}(Q)$. In this case, therefore, symmetries decrease in general.

— In contrast, in (ii) above, we only require P and $P[\sigma]$ to be equal up to the congruence. So even if no symmetry except identity exists originally, a new symmetry would arise after quotienting. Thus *a quotient in general increases symmetries of processes* (it should be noted that a quotient may 'cut off' unrelated names, for example, when some name is semantically insignificant in a process, so that the actual number of symmetries may decrease; however, the original symmetries are all inherited within the reduced set of names).

— In (iii), we know homomorphisms generally increase symmetries (in accordance with their close connection to quotient constructions). Further, isomorphisms completely characterise symmetries in addition to free names.

We now show that, in a rooted p-set, *symmetries characterise everything*. More precisely, if we pick up one symmetry group from each $\stackrel{\mathscr{N}}{\sim}$-equivalence class, we have all the essential information about the rooted p-set, up to isomorphism. Thus, not only do symmetries offer significant information on algebraic structures of processes, but they are in effect all we need to determine a rooted p-set. The construction follows. The proof uses elementary notions from the theory of group actions and is relegated to the next subsection.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

Proposition 3.4. (Representation of a rooted p-set by symmetries) Given $\mathscr{P}$ and $I = \mathscr{P}/\overset{\mathscr{N}}{\sim}$, a *symmetry presentation* of $\mathscr{P}$, written α, β, and so on, is an I-indexed family of symmetries, say $\{sym(P_i)\}_{i\in I}$, such that $P_i \in i$ for all $i \in I$. Then α and β, possibly from two different rooted p-sets, are *isomorphic*, written $\alpha \leftrightarrow \beta$, when there is a bijection F between them such that: $F(\alpha_i) = \sigma \cdot \alpha_i \cdot \sigma^{-1}$, where α_i denotes the i'th member of α. Then $\leftrightarrow$ is an equivalence. Moreover:

(i) If α and β come from the same rooted process set, then $\alpha \leftrightarrow \beta$ always.
(ii) By *(i)*, write $\mathscr{P} \leftrightarrow \mathscr{P}'$ when a pair of symmetry presentations of $\mathscr{P}$ and $\mathscr{P}'$ are isomorphic. Then we have $\mathscr{P} \leftrightarrow \mathscr{P}'$ if and only if $\mathscr{P} \simeq \mathscr{P}'$.

Proposition 3.4 shows that each object in RPS, and hence in RPS_{rel}, allows a representation by a family of its symmetries. This representation is concise: all name variants of a process, which are in general countably many, are represented by a single symmetry group. The symmetry presentations are even equipped with an idea of isomorphism, but that is all, so we have not much other than the object part of the corresponding categories. We may then ask 'What notions correspond to the *arrow parts* of RPS_{rel} and RPS?' That is, what are the relations and maps for these symmetry presentations? Sections 4 and 5 will answer these questions.

3.2. *A group-theoretic perspective on rooted process sets*

This subsection proves Proposition 3.4 using a few elementary notions from the theory of group actions. Those readers who are not interested in technical details can safely skip the present subsection. We note that, nevertheless, there may be some interests in the technical development, since it presents an alternative viewpoint on rooted p-sets from a group theoretic perspective. We use the following notions from theory of group actions, *cf.* Lang (1993).

Definition 3.5. In the following, let G be a group and X be a set, and consider a group action of G on X. We write gx for the action of g on x.

(i) (Morphism) Given two G-sets X and Y, a *morphism* from X to Y is a function $F : X \to Y$ such that $F(gx) = g(Fx)$ for any $g \in G$ and $x \in X$. A morphism is an *isomorphism* if it is bijective.
(ii) (Orbits) A maximal subset Y of X such that, for any $y_1, y_2 \in Y$, we have $y_2 = gy_1$ for some g, is called an *orbit* of X. Orbits partition X and thus induce an equivalence relation.
(iii) (Transitive G-set) A G-set X is *transitive* if for any $x_1, x_2 \in X$, we can write $x_2 = gx_1$. So, in (ii), each orbit is a transitive G-set, and X as a whole is a disjoint union of these transitive G-sets.
(iv) (Conjugate) Given a subgroup G' of G, we have $G'' = gG'g^{-1}$ $(= \{gg'g^{-1} | g' \in G'\})$ is another subgroup of G. Then we say G' and G'' are *conjugate*. Note conjugate subgroups are isomorphic.
(v) (Isotropy) Let G_x be $\{g \mid g \in G \text{ such that } gx = x\}$. Then G_x forms a subgroup for each $x \in X$, called the *isotropy subgroup* of x. Note that if $y = gx$, we have $G_y = gG_xg^{-1}$, that is, G_x and G_{gx} are always conjugate.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

Notation 3.6. $S_{\mathcal{N}}$ denotes the group of all permutations over $\mathcal{N}$, with the function composition as the multiplication. We write $S_{\mathcal{N}\langle P\rangle}$ for the isotropy group of P under $S_{\mathcal{N}}$.

Using the above terminology, we observe that a rooted p-set is nothing but a group action of $S_{\mathcal{N}}$ over processes (because $P[\sigma \circ \sigma'] = P[\sigma'][\sigma]$ and $P[\mathbf{id}] = P$), augmented with an extra function $\mathcal{FN}(\cdot)$ that relates the permutations and elements of the set. Also note that $P \overset{\mathcal{N}}{\sim} Q$ precisely means P and Q are in the same orbit (so given some $\mathcal{P}$, the equivalence classes from $\overset{\mathcal{N}}{\sim}$ are orbits in $\mathcal{P}$ under $S_{\mathcal{N}}$). We observe that, by (iv) above, $S_{\mathcal{N}\langle P[\rho]\rangle}$ is given by $\rho \cdot S_{\mathcal{N}\langle P\rangle} \cdot \rho^{-1}$, which is isomorphic (in the above sense) to $S_{\mathcal{N}\langle P\rangle}$. We now observe that any G-set can be represented by the group structures of each orbit as follows.

Proposition 3.7. Two G-sets X and Y are isomorphic if and only if there is a bijective correspondence between their orbits and, moreover, in each pair of orbits $\langle X_i, Y_j\rangle$ under this correspondence, we have $G_x = gG_yg^{-1}$ for some $x \in X_i$, $y \in Y_j$ and $g \in G$.

The proof is given in Appendix A. We also have the following lemma.

Lemma 3.8.

(i) Given any isotropy $\rho \in S_{\mathcal{N}\langle P\rangle}$, its restriction (of the domain) to $\mathcal{FN}(P)$ gives a permutation of $\mathcal{FN}(P)$, which is indeed a symmetry of P.
(ii) (Representation of isotropy) $S_{\mathcal{N}\langle P\rangle} = S_{\mathcal{N}\langle Q\rangle}$ if and only if $sym(P) = sym(Q)$.

Proof. For (i), if ρ is not a bijection over $\mathcal{FN}(P)$, then $\rho(\mathcal{FN}(P)) \neq \mathcal{FN}(P)$, which is a contradiction. Then use Proposition 2.4 (iii). This already shows that the 'if' direction of (ii) is immediate. For the 'only if' direction, if we know that $S_{\mathcal{N}\langle P\rangle} = S_{\mathcal{N}\langle Q\rangle}$ implies $\mathcal{FN}(P) = \mathcal{FN}(Q)$, we get $sym(P) = sym(Q)$, again by Proposition 3.8. But suppose this is not the case, that is, for some a we have, for example, $a \in \mathcal{FN}(P)\backslash\mathcal{FN}(Q)$, assuming $S_{\mathcal{N}\langle P\rangle} = S_{\mathcal{N}\langle Q\rangle}$. Take f fresh, then $\binom{af}{fa}$ is in $S_{\mathcal{N}\langle Q\rangle}$ but not in $S_{\mathcal{N}\langle P\rangle}$ by Definition 2.1 (iii)-3, which is a contradiction. □

We can now establish Proposition 3.4. Since $P \overset{\mathcal{N}}{\sim} Q$ implies $S_{\mathcal{N}\langle P\rangle}$ and $S_{\mathcal{N}\langle Q\rangle}$ are conjugate, and, by Lemma 3.8, this implies the conjugacy of $sym(P)$ and $sym(Q)$ again, (i) follows. For (ii), the 'only if' direction follows from Proposition 3.7 and Lemma 3.8. For the 'if' direction, suppose $\mathcal{P} \simeq \mathcal{P}'$, that is, there is an isomorphism $\mathcal{F} : \mathcal{P} \to \mathcal{P}'$. Clearly each orbit in $\mathcal{P}$ is one-one mapped to an orbit in $\mathcal{P}'$, and, by Proposition 3.3 (iii), this map preserves symmetries, hence we are done.

4. Nameless processes

4.1. *Process sets*

In this section we present a theory that is essentially equivalent to what we have seen in the preceding sections, but is based on a more geometric representation, and exposes functionalities of names in relations and algebras of processes from a different perspective. Though this has a close relationship with our development so far, and remembering the question posed at the end of Section 3.1, we prefer to start by presenting the theory without referring to the preceding development, to make the basic ideas come out as

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

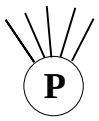

Fig. 1. A Process

succinctly and clearly as possible. Later we will come back to the question of how the two theories are related.

The following is the primary object of study in this and the next section.

Definition 4.1. (Process sets) *A (non-rooted) process set*, which we will often simply call a *p-set*, is a triple $\langle \mathbf{P}, \mathscr{H}, \mathscr{S} \rangle$ where:

(i) $\mathbf{P}$ is the set of *(non-rooted) processes*, ranged over by $p, q, r, \ldots$.
(ii) $\mathscr{H}$ assigns, to each p, a set of *handles* of p. A handle is denoted by $h, h', \ldots$.
(iii) $\mathscr{S}$ assigns, to each p, a subgroup of the group of permutations over $\mathscr{H}(p)$, called *symmetries on p*.

A *substructure* of a p-set is a subset of processes together with two assignments inherited from the original structure. The *arity* of a process, written $ar(p)$, is given by the cardinality of $\mathscr{H}(p)$, which we assume finite. Often the underlying set $\mathbf{P}$ denotes a p-set, in which case we write $\underline{\mathbf{P}}$ to denote the set of processes.

More concisely, a p-set is a set-indexed family of concrete permutation groups, each over a finite set. We may think of a process as a simple geometric object with multiple discrete points of connection and interaction.

Such structures are abundant in the theory of computing, either implicitly or explicitly.

Examples 4.2.

(i) Any set is a p-set, considering each element as a handleless process. In particular, we write I for a (distinguished) singleton set seen as a p-set.
(ii) Another simple example is a p-set with only one process that has only one handle. This p-set is written **1**. Similarly, **2** denotes a p-set with one process, which has two handles with trivial symmetry (that is, only identity).
(iii) As a different example, we can take a collection of hypergraphs as a p-set, whose nodes are considered to be handles, with graph-theoretic symmetries. The next example can be considered as a refinement of this example.
(iv) Girard's Proof Nets (Girard 1987; Girard 1996) form a p-set, where we regard the end points of nets as handles. Later we shall see how operations and dynamics on nets can be formulated in the present setting. The natural graph-theoretic symmetries induce symmetries. Similarly, Lafont's Interaction Nets (Lafont 1990; Lafont 1995; Lafont 1996) form a p-set, where we regard nets as processes. As Lafont discussed in Lafont (1995), there are natural symmetries in some nets beyond simple graph-theoretic ones. Related examples are graphical formalisms representing reduction of λ-calculi (Lamping 1990; Gonthier *et al.* 1992) and process graphs in Yoshida (1994).

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

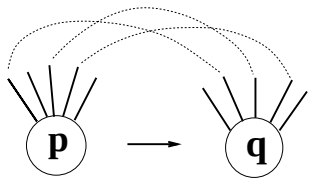

Fig. 2. A correspondence

(v) The set of arrows in a category gives p-sets. Each arrow is considered as a process having two handles with no symmetries.
(vi) Finally, we give a somewhat abstract, yet basic, example: the set of all pairs of form $\langle n,\ G\rangle$, where n is a natural number considered as a finite set $\{0, 1, \ldots, n-1\}$ (with $0 = \varnothing$), and G is a permutation group on n, gives a p-set. For convenience, we assume that if $\langle n,\ G\rangle$ and $\langle n,\ G'\rangle$ are isomorphic as group actions (*cf.* Definition 3.5 (iii)), then we take only one. The resulting p-set is denoted $\boldsymbol{\omega}$. Notice the exact image of any process in any p-set can be found in $\boldsymbol{\omega}$. It has a substructure $\boldsymbol{\omega}_{\mathsf{full}}$ by taking all and only $\langle n,\ G\rangle$ such that G is the group of all permutations on n.

In many senses, a p-set can be considered as a generalisation of a set: instead of a collection of elements, we have a collection of processes, where each process is inherently equipped with its own 'interface structure', that is, its handles and symmetries. This interface structure captures what has been accomplished by names in the name-based theory, that is, the place of connection and interaction of computing agents.

Discussion 4.3. (On size) We can relax two restrictions on size from Definition 4.1. First, we may as well take the number of handles to be a set of any size. Second, the collection of processes themselves may form a proper class (such is the case when, for example, we regard a locally small category as a p-set). With these two extensions, we can still develop the theory in this section without any change, even though we shall stick to the present simple framework, which will be enough for many applications.

4.2. *Correspondence and p-relations*

Our study of algebra of (non-rooted) process sets starts from a basic way of relating two processes, called *correspondence*.

Definition 4.4.

(i) A *correspondence* is a triple $\langle p_1,\ \delta,\ p_2\rangle$ in which δ is an injective partial function from $\mathscr{H}(p_1)$ to $\mathscr{H}(p_2)$ (see Figure 2). The *domain* of $\langle p,\ \delta,\ q\rangle$, written $\mathtt{dom}(\langle p,\ \delta,\ q\rangle)$, is p and its codomain, written $\mathtt{cod}(\langle p,\ \delta,\ q\rangle)$, is q. The *inverse* of a correspondence $\langle p,\ \delta,\ q\rangle$, written $\langle p,\ \delta,\ q\rangle^{-1}$, is $\langle q,\ \delta^{-1},\ p\rangle$.
(ii) A correspondence $\langle p,\ \delta,\ q\rangle$ is *surjective* (respectively, *total*) if δ is surjective (respectively, total). It is *symmetry preserving* when, for each $\rho \in \mathscr{S}(p)$, there is some $\rho' \in \mathscr{S}(q)$ such that $\delta \circ \rho = \rho' \circ \delta$. It is *symmetry reflecting* if its inverse is symmetry

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

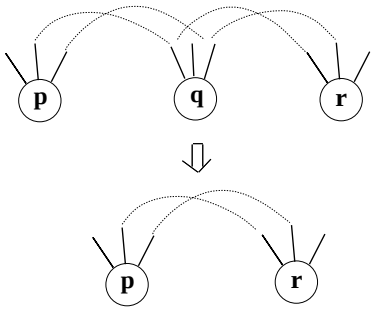

Fig. 3. Composition of correspondences

preserving. A correspondence that is surjective and symmetry preserving is often written with the symbols $f, g, \ldots$.

(iii) We say $\langle p,\ \delta,\ q\rangle$ and $\langle p,\ \delta',\ q\rangle$ are *equivalent*, written $\delta \sim \delta'$, if, for some $\rho \in \mathscr{S}(p)$ and $\rho' \in \mathscr{S}(q)$, we have $\rho' \circ \delta \circ \rho = \delta'$.

(iv) Given $\langle p,\ \delta_1,\ q\rangle$ and $\langle p,\ \delta_2,\ q\rangle$, we write $\langle p,\ \delta_1,\ q\rangle \leqslant \langle p,\ \delta_2,\ q\rangle$ when $\delta_1 \subset \delta_2$.

(v) Let $\langle p,\ \delta_1,\ q\rangle$ and $\langle q,\ \delta_2,\ r\rangle$ be correspondences. Then their *composition* is another correspondence $\langle p,\ \delta_2 \circ \delta_1,\ r\rangle$ (see Figure 3).

By abuse of notation, we often write $\delta, \delta', \ldots$ for correspondences. In this case, we write $\underline{\delta}$ for the component injective partial function of δ. We note the following proposition.

Proposition 4.5.

(i) If $\delta_1 : p \to q$ and $\delta_2 : q \to r$ are both surjective (symmetry preserving or symmetry reflecting), then $\delta_2 \circ \delta_1$ is again surjective (symmetry preserving, or symmetry reflecting, respectively).

(ii) If $\delta_1 \sim \delta_2$, then δ_1 is surjective (total, symmetry preserving or symmetry reflecting, respectively) iff δ_2 is also.

(iii) Write $\mathbf{id}_p$ for the identity correspondence on p. Then $\delta \sim \mathbf{id}_p$ iff δ is total, surjective and symmetry preserving/reflecting.

We may regard correspondences as an analogue of *tuples* in the present setting: then their collections form what may be regarded as *relations* in the present setting.

Definition 4.6.

(i) (p-relation) A *p-relation* $\mathfrak{R}$ *from* $\mathbf{P}_1$ *to* $\mathbf{P}_2$ is a set of correspondences with domains in $\mathbf{P}_1$ and codomains in $\mathbf{P}_2$ such that if $\delta \in \mathfrak{R}$ and $\delta' \sim \delta$, then $\delta' \in \mathfrak{R}$ again. We write $p\mathfrak{R}_\delta q$ to mean p is related to q in $\mathfrak{R}$ by δ.

(ii) (Composition) Given $\mathfrak{R}_1$ from $\mathbf{P}$ to $\mathbf{Q}$ and $\mathfrak{R}_2$ from $\mathbf{Q}$ to $\mathbf{R}$, their *composition* $\mathfrak{R}_2 \circ \mathfrak{R}_1$ is a p-relation from $\mathbf{P}$ to $\mathbf{R}$ given as: $\{\, \delta \mid$ for some $\delta_1 \in \mathfrak{R}_1$ and $\delta_2 \in \mathfrak{R}_2$ such that $\mathtt{dom}(\delta_2) = \mathtt{cod}(\delta_1)$ we have $\delta \geqslant \delta_2 \circ \delta_1$, $\delta_1 \geqslant \delta_2^{-1} \circ \delta$, and $\delta_2 \geqslant \delta \circ \delta_1^{-1}\,\}$.

(iii) (Product, identity, equivalence) The *product* of $\mathbf{P}_1$ and $\mathbf{P}_2$, written $\mathbf{P}_1 \times \mathbf{P}_2$, is the set of all correspondences between $\mathbf{P}_1$ and $\mathbf{P}_2$. The *identity p-relation* on $\mathbf{P}$ is the set of all symmetries (as correspondences) on processes of $\mathbf{P}$, written $\mathbf{ID}_\mathbf{P}$. A p-relation $\mathfrak{R}$

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

on **P** (that is, from **P** to **P**) is *reflexive* if $\mathbf{ID}_{\mathbf{P}} \subset \mathfrak{R}$, *transitive* if $\mathfrak{R} \circ \mathfrak{R} \subset \mathfrak{R}$, and *symmetric* if $\mathfrak{R}^{-1} = \mathfrak{R}$. If the three hold together, $\mathfrak{R}$ is an *equivalence p-relation.*

Observe that we can take the union, intersection, and inverse of a family of p-relations just as we manipulate set-theoretic relations. The composition in Definition 4.6 (ii) says that, in relational composition, we arbitrarily connect those handles of processes not touched in the component connections. As we shall see later, this is the exact counterpart of relational composition of compatible relations in the name-based theory in the present setting. By the definition, we can write any correspondence in $\mathfrak{R}_2 \circ \mathfrak{R}_1$ as $(\delta_2 \circ \delta_1) \uplus \delta'$, where $\uplus$ is the disjoint union. Using this, we can readily check that $\circ$ is associative. Finally, because any p-relation is closed under symmetries on both sides, **ID** acts as the left and right identity. Significant examples of p-relations are given in Section 4.5.

One remark concerning the presentation of p-relations should be made. As can be seen from the definition, the basic building block of a p-relation is a correspondence modulo $\sim$, which we shall call an *abstract correspondence*, rather than a concrete correspondence. Moreover, it is an abstract correspondence that faithfully captures a single geometric situation of connection between two processes. However, abstract correspondences are sometimes more difficult to handle than the concrete ones. For example, composition of two abstract correspondences may *not* form another (single) abstract correspondence. To see this, given $p[h]$, $q[i_1, i_2]$, and $r[j]$ (here $[h]$ *etc.* shows the handles a process owns) with $\mathscr{S}(q) = \{\binom{i_1 i_2}{i_2 i_1}, \binom{i_1 i_2}{i_1 i_2}\}$. Now, if we compose $\{\langle p,\ \{h \mapsto i_1\},\ q\rangle, \langle p,\ \{h \mapsto i_2\},\ q\rangle\}$ and $\{\langle q,\ \{i_1 \mapsto j\},\ r\rangle, \langle q,\ \{i_2 \mapsto j\},\ r\rangle\}$, both being abstract correspondences, we get $\{\langle p,\ \{h \mapsto j\},\ q\rangle,\ \langle p,\ \varnothing,\ q\rangle\}$, which is the union of *two* abstract correspondences. However, to view a p-relation as an aggregate of abstract correspondences will often be useful for a conceptual understanding of the various notions in the subsequent technical development.

Discussion 4.7. Concerning the relational composition, it is worth noting that the following simpler way of composing relations, where we take straightforward elementwise composition

$$\mathfrak{R}_2 \circ' \mathfrak{R}_2 \stackrel{\text{def}}{=} \{\delta_2 \circ \delta_1 \mid \delta_1 \in \mathfrak{R}_1 \ \wedge\ \delta_2 \in \mathfrak{R}_2 \ \wedge\ \mathtt{dom}(\delta_2) = \mathtt{cod}(\delta_1)\} \tag{3}$$

does *not* yield the counterpart of the relational composition in the name-based theory (this corrects our erroneous statement in Honda (1995a), which says the contrary). Nevertheless, the composition in (3) above induces a coherent relational theory, and for some important classes of p-relations these two notions coincide (*cf.* Proposition 4.10 (i)).

4.3. *P-maps*

In this section we introduce some p-relations that act as functions in the present setting.

Definition 4.8. (p-map) A *p-map* is a p-relation $\mathfrak{R}$ such that: (1) $\mathfrak{R}$ contains, for each process, say p, in the source, a correspondence whose domain is p, (2) If $\delta, \delta' \in \mathfrak{R}$ and, moreover, their domain coincides, then $\delta \sim \delta'$, and (3) $\delta \in \mathfrak{R}$ implies δ is surjective and symmetry preserving. A p-map whose inverse is also a p-map is called an *isomorphism.*

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

A *partial p-map* is a p-relation that satisfies (2) and (3) of the above, but not necssarily (1). Both partial and total p-maps are ranged over by $\mathscr{F}$, $\mathscr{G}$, etc.

Note that a p-map consists of a collection of abstract correspondences, one for each process in the domain. Note also our definition of isomorphisms is stronger than bijective p-maps: each component injection should become a symmetry-preserving/reflecting bijection. Clearly, identities are isomorphisms in this sense. Some examples of p-maps follow.

Examples 4.9. (p-maps)

(i) Between sets (seen as p-sets), p-maps are precisely set-theoretic functions. Also, there is a unique p-map from $\varnothing$ to each p-set. Similarly, there is a unique p-map from each p-set to I.

(ii) There is a unique p-map from any p-set to $\boldsymbol{\omega}$ such that each component correspondence is bijective, symmetry-preserving and reflecting. If we take $\boldsymbol{\omega}_{\mathsf{full}}$, we again have a unique injective p-map from each process structure such that each component correspondence is bijective but may not be symmetry reflecting.

(iii) A functor is not only a p-relation but is also a p-map, even if most p-maps between categories are *not* functors.

Other prominent examples of p-maps are operations on processes in graphical formalisms such as proof nets, see Section 4.5. The basic properties of p-maps and isomorphisms follow.

Proposition 4.10.

(i) Given two p-maps composable as p-relations, their composition is again a p-map. Moreover, in this case the composition coincides with the relational composition given by Equation (3) in Discussion 4.7.

(ii) Let $\mathfrak{I}$ be an isomorphism. Then $\mathfrak{I}$ is bijective on processes. If $\langle p,\ \delta,\ q\rangle \in \mathfrak{I}$, then δ is also bijective. Moreover, $\mathscr{S}(p)$ and $\mathscr{S}(q)$ are isomorphic in the following way:

$$\mathscr{S}(q) = \delta \cdot \mathscr{S}(p) \cdot \delta^{-1}. \tag{4}$$

Conversely, $\mathbf{P}$ and $\mathbf{Q}$ are isomorphic iff there is a bijection between $\underline{\mathbf{P}}$ and $\underline{\mathbf{Q}}$ such that, for each pair $\langle p,\ q\rangle$, there is a bijection δ on handles for which Equation (4) holds.

Proof. For (i), the only non-trivial part is the uniqueness modulo $\sim$. Suppose $\delta_2' \circ \delta_1'$ and $\delta_2 \circ \delta_1$ are in composed p-maps that share the same domain. By definition, $\delta_1 \sim \delta_1'$, hence (the domains coincide and) $\delta_2 \sim \delta_2'$, hence we can write $\delta_1' = \rho_2 \circ \delta_1 \circ \rho_1$ and $\delta_2' = \pi_3 \circ \delta_1 \circ \pi_2$, where $\rho_{1,2}$ and $\pi_{1,2}$ are symmetries of the respective processes. We now calculate (using the symmetry preservation)

$$\begin{aligned} \delta_2' \circ \delta_1' &\overset{\text{def}}{=} \pi_3 \circ \delta_2 \circ \pi_2 \circ \rho_2 \circ \delta_1 \circ \rho_1 \\ &= \pi_3 \circ \pi_3' \circ \delta_2 \circ \delta_1 \circ \rho_1 \quad \sim \delta_2 \circ \delta_1 \end{aligned}$$

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

where π_i are appropriate symmetries, and hence we are done. The simpler presentation of the composition follows because if $\delta \leqslant \delta'$ and δ' is surjective, then $\delta = \delta'$. (ii) is immediate. □

Thus we have the categories for the theory of name-free processes.

Definition 4.11. PS denotes the category of p-sets and p-maps, while PS_{rel} denotes the category of p-sets and p-relations.

In Section 5 we shall show that PS (respectively, PS_{rel}) is categorically equivalent to RPS (respectively, RPS_{rel}). In particular, this means that PS is also a topos with infinitary products, which ensures our understanding of the theory of p-set as an analogue of set theory. Other basic facts on PS include: categorical isomorphisms are precisely isomorphisms in the sense of Definition 4.8; epimorphisms are p-maps that cover the whole set of processes in the target; and monomorphisms are those that are injective on processes and, moreover, each component correspondence is surjective.

4.4. *Products, quotient and algebra*

The analogue of the constructions in Definition 2.9 looks like the following in the present setting. They offer fundamental strata for algebraic manipulation of non-rooted processes. They again correspond to respective categorical counterpart in PS, see Appendix A. Note that, in particular, this means the constructions are unique up to isomorphism. It also indicates that the operation of the product is commutative and associative up to isomorphism. Below we write $\mathtt{im}(\delta)$ (respectively, $\mathtt{pre}(\delta)$) for the image (respectively, pre-image) of δ as a partial function.

Definition 4.12.

(i) (Product) Define a p-set $\mathbf{P} \times \mathbf{Q}$ by the following data: the set of processes are given by representative elements of the quotient set $\underline{\mathbf{P}} \times \underline{\mathbf{Q}}/\sim$, handles of each $\langle p,\ \delta,\ q\rangle$ in the set are given by $(\mathscr{H}(p)\backslash\mathtt{im}(\delta^{-1})) \uplus \delta \uplus (\mathscr{H}(q)\backslash\mathtt{im}(\delta))$, and the symmetries are given by $\{\langle \rho_1,\ \rho_2\rangle \mid \rho_2^{-1} \circ \delta \circ \rho_1 = \delta,\ \rho_1 \in \mathscr{S}(p_1) \wedge \rho_2 \in \mathscr{S}(p_2)\}$, where $\langle \rho_1,\ \rho_2\rangle$ acts as

$$\begin{aligned} \langle \rho_1,\ \rho_2\rangle(h_1) &= \rho_1(h_1) \quad (h_1 \in \mathscr{H}(p)\backslash\mathtt{pre}(\delta)) \\ \langle \rho_1,\ \rho_2\rangle(h_1 \mapsto h_2) &= \rho_1(h_1) \mapsto \rho_2(h_2). \\ \langle \rho_1,\ \rho_2\rangle(h_2) &= \rho_2(h_2) \quad (h_2 \in \mathscr{H}(q)\backslash\mathtt{im}(\delta). \end{aligned}$$

We call $\mathbf{P} \times \mathbf{Q}$ the *product* of $\mathbf{P}$ and $\mathbf{Q}$.

(ii) (Quotient) Let $\sim$ be an equivalence p-relation. Define $\mathscr{H}(p) \restriction\sim$ as $\{h \in \mathscr{H}(p) \mid p \sim_\delta p \ \Rightarrow\ h \in \mathtt{im}(\delta)\}$. Then the *quotient* of $\mathbf{P}$ by $\sim$, written $\mathscr{H}(p)/\sim$, is given by the representative elements of the quotient set of $\underline{\mathbf{P}}$ by $\sim$, for each of which, say p, we assign handles using $\mathscr{H}(p) \restriction\sim$ (taken in $\mathbf{P}$) and assign symmetries using $\{\delta \restriction (\mathscr{H}(p) \restriction\sim) \mid p \sim_\delta p\}$, that is, we take the restriction of reflexive correspondences to $\mathscr{H}(p)|\mathfrak{R}$.

While the constructions in Definition 4.12 may look more complex than the corresponding ones for rooted processes, they are actually quite simple geometrically. We first consider

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

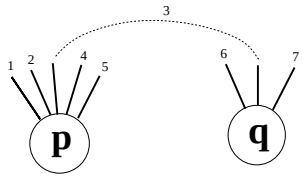

Fig. 4. A correspondence as a process

how a correspondence becomes a process in (i). Figure 4 describes a singleton correspondence from p, which has five handles, to q, which has three. This is viewed as a process which has: (1) four handles unconnected in p, (2) one handle as the result of the identification, and (3) two handles left unconnected in q, so in total seven handles, as numbered in Figure 4. A permutation on the new handles is a symmetry precisely when a combination of a symmetry of p and that of q together leaves this whole object unchanged (in the present case this means each symmetry should be identity on the respective connected handle).

In the quotient construction in (ii), we are taking the part of handles of an arbitrary representative process of each equivalence class, which is the image of all connections to the process. Symmetries are given by all correspondences acting on that part. Both are invariant (up to conjugation) regardless of the choice of processes. Because an equivalence is reflexive, they include all original symmetries. In both (i) and (ii), the degenerate case, that is when the relevant p-sets are just sets, describes precisely the set-theoretic product and quotient. This also clarifies how the present theory generalises set-theoretic notions by incorporating additional information for its 'elements'. Note how this construction offers an alternative understanding of the constructions in Definition 4.12.

As in the name-based theory, products and quotients play a basic role in the theoretical treatment of processes: the former in the context of algebraic operations and the latter in the context of equational reasoning. Below we write $\prod_{1\leqslant i\leqslant n}\mathbf{P}$ for the repeated product $\mathbf{P}_1\times\mathbf{P}_2\times\ldots\times\mathbf{P}_n$.

Definition 4.13.

(i) *An operation from* $\{\mathbf{P}_i\}_{1\leqslant i\leqslant n}$ *to* $\mathbf{P}$ is a p-map from $\prod_{1\leqslant i\leqslant n}\mathbf{P}_i$ to $\mathbf{P}$. If $\mathbf{P}_i=\mathbf{P}$ for each i, then it is an *n-ary operation on* $\mathbf{P}$.

(ii) Similarly, *a partial operation from* $\{\mathbf{P}_i\}_{1\leqslant i\leqslant n}$ is a partial p-map from $\prod_i\mathbf{P}_i$ to $\mathbf{P}$. Again, if $\mathbf{P}_i=\mathbf{P}$ for each i, we say it is an *n-ary partial operation on* $\mathbf{P}$.

Operations induce an algebra in the same way as in algebra over sets. Such an algebra would be called *p-algebras*. Understanding the concrete content of this 'operation' elucidates the nature of such algebras, as well as that of their name-based counterpart. Consider a binary operation, say $\odot:\mathbf{P}^2\to\mathbf{P}$. In such an operation, we specify two processes from $\mathbf{P}$ together with the way the handles of these processes should be connected. Then $\odot$ determines how this composition results in a new process of $\mathbf{P}$, with each handle of that process inherited from the original processes. If $\odot$ is a partial operation, then what we have is the situation where, for some ways of connecting processes, the composition is not allowed (by being undefined). Thus partial operations can preclude

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

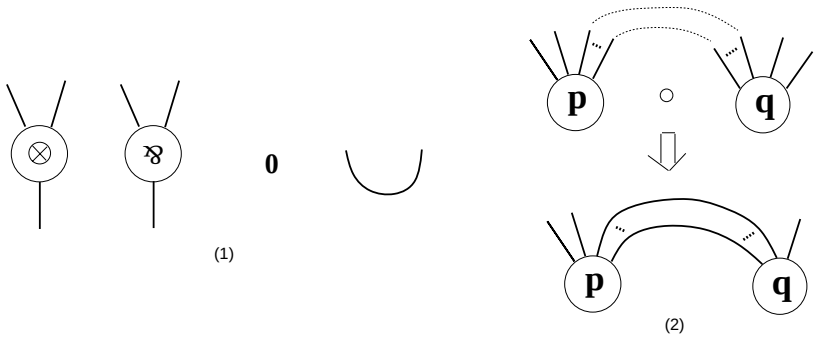

Fig. 5. Proof nets: atoms and composition

certain configurations from being formed, which is the essence of typed composition of processes in comparison with untyped ones (Honda 1996).

Finally, we observe that the key attributes of the present relational theory, including the fact that PS becomes a topos, comes from certain algebraic properties of partial injections between concrete permutation groups, which can be concisely axiomatised. Such an axiomatic account helps us understand at what points the present theory enriches the theory of sets.

4.5. *Example*

In this subsection we show how proof nets (Girard 1987) induce p-sets and associated constructions. We also mention other known examples of p-sets at the end of the section. For simplicity, we restrict our attention to the the multiplicative fragment of proof nets, and do not consider types (which can be easily incorporated by using partial p-maps). The construction makes it possible to articulate precisely the algebra and dynamics of nets. The presentation follows that of Lafont (Lafont 1995).

Examples 4.14. Proof nets are constructed inductively, starting from atoms and composing two nets. Figure 5 (1) gives atoms, where edges are considered as handles (of processes). We assume the last atom, which is called a *wire* (*cf.* Example 2.11), has the full symmetries; for the others we assume only trivial symmetries (that is, the identities). The incorporation of the wire as an atom is important in order to clarify the algebraic status of the nets. In (2) of Figure 5, we schematically depict how two processes get composed. Note that the operation is 'composition with hiding', that is, the connected ends are no longer interfaces to the outside. We assume that the composed nets own symmetries induced by obvious geometric symmetries. It is easy to verify that these operations define p-maps, so that we now have a p-algebra on the set of nets.

Building on the algebra, the dynamics is given by the closure of the rule in Figure 6 under the operation.

Finally, we observe that the equality on proof nets induced by the normalisation procedure induces a quotient p-set that inherits the original algebraic operations. The quotient adds symmetries (by definition), but does *not* drop any handles. Because a net

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

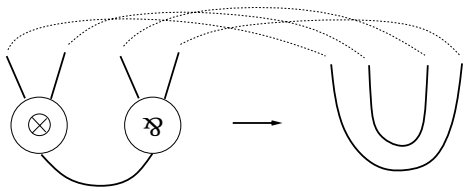

Fig. 6. Proof nets: dynamics

always has a normal form, a congruence class corresponds essentially to a normal form of all the nets that eventually reduce to it. One interesting question would be whether such a normal form has no fewer symmetries than any net in the equivalence class or not (this can be called the *symmetry theorem* for proof nets). Categorical and other semantics of proof nets also induce such quotient constructions.

We can similarly formalise the dynamics and algebras of various graphical formalisms representing the reduction in λ-calculus (Lamping 1990; Gonthier *et al.* 1992). Geometric presentations of π-calculi, including those of its combinatory versions, as found in Milner (1994), Parrow (1993) and Yoshida (1994), can also be reformulated in the present framework. In the latter examples, the composition of graphs becomes open, that is, the original handles are inherited in the resulting process even after their connection. As a different example, the composition of arrows in any category is a partial binary operation, similarly n-ary functors are n-ary operations on categories as p-sets. Also, a quotient in a category is a quotient in the present sense. To understand categorical constructions in this way is still a long way from being able to explain the general significance of these constructions; however, it does offer a basic viewpoint when categories are used for representing semantics of interacting processes, for example as in game-based semantics (Abramsky *et al.* 1994; Hyland and Ong 1994).

5. From symmetries to nameless processes and back

5.1. *Translations*

In this section we establish the following main result of the present paper: that the name-based theory and name-less theory are essentially equivalent.

Theorem 5.1. (Main Theorem: Equivalence) RPS and PS are categorically equivalent. Similarly, RPS_{rel} and PS_{rel} are categorically equivalent.

We establish the theorem as follows. First, in this subsection, we present translations of one theory into the other, in both directions. These translations offer concrete ways of moving between a name-based theory and the corresponding name-less one. As illustration, we present two significant examples of the translation. One is the name-free presentation of the essential π-calculus in Section 2.3, while the other is the name-based presentation of the fragment of proof nets in Section 4.5. Finally, we establish Theorem 5.1, by showing

that those translations result in objects, relations and maps mediated by isomorphisms. Throughout this section we fix some $\mathcal{N}$.

We start with a formulation of the mutual translation at the level of objects. Let us first go back to Proposition 3.4, where we proved that a symmetry presentation of a rooted p-set maintains all essential information of the original structure, up to isomorphism. However, we already know that such a presentation is nothing but a p-set.

Proposition 5.2. Suppose α is a symmetry presentation of some rooted p-set, that is, a family $\{sym(P_i)\}_{i\in I}$ where I is the equivalence classes by $\overset{\mathcal{N}}{\sim}$ and $P_i \in i$ for each i. Then this defines a p-set, with the following data:

(i) Processes are given by I,
(ii) Handles are given by $\mathscr{H}(i) = \mathscr{FN}(P_i)$, and
(iii) Symmetries are given by $\mathscr{S}(i) = sym(P_i)$.

Moreover, we have $\alpha \leftrightarrow \beta$ if and only if $\alpha \simeq \beta$.

Proof. The 'only if' direction of the last statement follows from Proposition 4.10 (ii). The others are straightforward. □

Taken with Proposition 3.4, this shows that we always get isomorphic p-sets from isomorphic rooted p-sets by taking any of latter's symmetry presentations. We will now illustrate the translation with some simple examples; more complex examples are discussed in the next subsection.

(i) Take any set as a rooted p-set, *cf.* Example 2.3 (i). Then each of its orbits (*cf.* Definition 3.5 (ii)) is a singleton, so its symmetry presentation is itself, which is indeed a p-set.
(ii) Take $\mathcal{N}$ as a p-set. It has a single orbit, its element having only one name. Therefore its symmetry presentation is nothing but **1**, as given in Example 4.2 (ii), up to isomorphism.
(iii) Next take $\{\langle a,\ b\rangle | a \neq b,\ \ a, b \in \mathcal{N}\}$, which again has a single orbit. One can easily see that its symmetry presentation is **2**.
(iv) Take $\mathsf{Fin}(\mathcal{N})$. This has an infinite number of orbits (an orbit for each finite cardinality), and the symmetries are full. Thus it exactly corresponds to $\boldsymbol{\omega}_{\mathsf{full}}$.

Next we show how a p-set can be translated into the corresponding rooted p-set.

Definition 5.3. Given a p-set $\mathbf{P}$, we define a rooted p-set by:

(i) Rooted processes: all pairs of form $\langle p,\ \Psi\rangle$ where $p \in \mathbf{P}$ and Ψ is an injection from $\mathscr{H}(p)$ to $\mathcal{N}$.
(ii) Operations: $[\sigma]$ for each permutation σ on $\mathcal{N}$ given by $\langle p,\ \Psi\rangle[\rho] \stackrel{\text{def}}{=} \langle p,\ \rho \circ \Psi\rangle$, and $\mathscr{FN}(\cdot)$ defined by $\mathscr{FN}(\langle p,\ \Psi\rangle) \stackrel{\text{def}}{=} \mathtt{im}(\Psi)$.

Next define $\approx$ so that $\langle p,\ \Psi\rangle \approx \langle p,\ \Psi'\rangle \ \Leftrightarrow\ \Psi' = \Psi \circ s$ for some $s \in \mathscr{S}(p)$. Then $\approx$ is a congruence (a compatible equivalence), and we set $[\![\mathbf{P}]\!] \stackrel{\text{def}}{=} \{\langle p,\ \Psi\rangle\}/\approx$, the quotient given as in Definition 2.9 (iii).

In the above definition, $\approx$ assigns the necessary symmetries on the resulting quotient. It is easy to check that $\approx$ does define a congruence, so that the quotient is a rooted p-set

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

again. The map $[\![\mathbf{P}]\!]$ acts on p-sets in the following way: $[\![X]\!] \simeq X$ for any set X, $[\![\mathbf{1}]\!] \simeq \mathcal{N}$ (the latter seen as a rooted p-set following Example 2.3 (ii)), $[\![\mathbf{2}]\!] \simeq \{\langle a, b\rangle | a \neq b\}$. These examples suggest the function $[\![\cdot]\!]$ does give the inverse of the translation of rooted p-sets into their symmetry presentations, up to isomorphism.

We now move to the mapping at the level of morphisms. While it is enough to show the construction for p-maps to establish Theorem 5.1, it is no more difficult to introduce the translation at the level of general p-relations, which we give below.

Definition 5.4. (Relational mappings)

(i) Given $\mathscr{R} : \mathscr{P} \to \mathscr{P}'$, we define $\mathscr{R}^\circ$ as the set of correspondences from $\langle\!\langle \mathscr{P} \rangle\!\rangle$ to $\langle\!\langle \mathscr{P}' \rangle\!\rangle$ generated by

$$\langle P, Q\rangle \in \mathscr{R} \quad \Rightarrow \quad \langle p, \pi \circ \mathbf{id}_{\mathscr{FN}(P)\cap\mathscr{FN}(Q)} \circ \rho^{-1}, q\rangle \in \mathscr{R}^\circ,$$

where $p = [P]_{\overset{\mathcal{N}}{\sim}}$ and $q = [Q]_{\overset{\mathcal{N}}{\sim}}$, assuming that, from the $\overset{\mathcal{N}}{\sim}$-equivalence classes of P and Q, $\langle\!\langle \cdot \rangle\!\rangle$ selects, respectively, $sym(P[\rho])$ and $sym(Q[\pi])$ as symmetry presentations.

(ii) Given $\mathfrak{R} : \mathbf{P}_1 \to \mathbf{P}_2$, we define $\mathfrak{R}^\star$ as the relation from $[\![\mathbf{P}_1]\!]$ to $[\![\mathbf{P}_2]\!]$ generated by

$$\langle p, \delta, q\rangle \in \mathfrak{R} \ \wedge \ \delta = \Psi_2^{-1} \circ \Psi_1 \quad \Rightarrow \quad \langle [\langle p, \Psi_1\rangle]_\approx, [\langle q, \Psi_2\rangle]_\approx\rangle \in \mathfrak{R}^\star,$$

where $\Psi_1 : \mathscr{H}(p) \to \mathcal{N}$ $\Psi_2 : \mathscr{H}(q) \to \mathcal{N}$ are injections.

In (i), we can easily check that a pair $\langle P, Q\rangle$ and its name variant $\langle P[\delta], Q[\delta]\rangle$ always result in the identical abstract correspondence (observe that a tuple would generate more than one correspondences if symmetries of P and Q are non-trivial: however, these correspondences are always mutually equivalent). In (ii), we generate the tuples by labelling each (non-rooted) process so that the coinciding names give the original correspondence precisely. Notice that two equivalent correspondences necessarily give rise to the same set of tuples. Thus it generates all name variants from a single abstract correspondence. It is worthwhile at this stage to look at a couple of concrete examples of the translations.

5.2. *Examples of translations*

To illustrate how the translations work, we give two substantial examples in this subsection. First we show a translation of the essential π-calculus in Example 2.11 into the corresponding nameless form. Following Definition 5.3, we start by taking the symmetry presentation of involved rooted processes.

Examples 5.5. (The essential π-calculus: a nameless presentation) In Figure 7 we present a presentation of the essential π-calculus in the name-less format. (1) gives a symmetry presentation of atomic agents, while (2) and (3) show two operations by which we generate a set of (hyper) graphs starting from those atomic agents. Here 0 denotes the empty graph. Filled circles give handles (corresponding to free names in the original presentation), which we may call *open edges*, while unfilled circles are not counted as handled (corresponding to bound names), which we may call *closed edges*. Notice that, when performing composition in (2), the collapsed edges are still open in the resulting configuration. It is only via the hiding operation in (3), which is to be considered as an operation of type $\mathbf{1} \times \mathscr{P} \to \mathscr{P}$

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

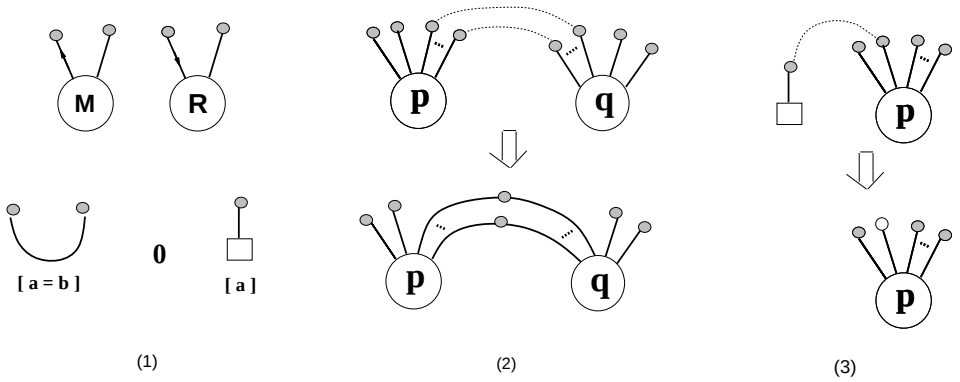

Fig. 7. Essential π-calculus: atoms and composition

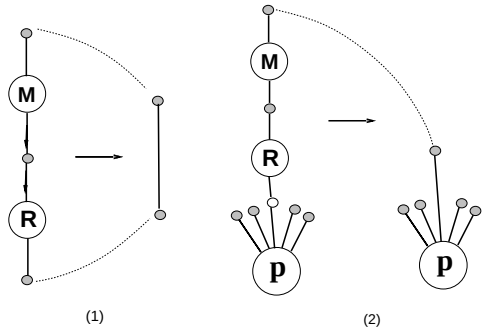

Fig. 8. Essential π-calculus: dynamics

where $\mathscr{P}$ is a set of generated graphs, that an open edge gets closed. Since only open edges are handles, these operations only mention open edges. For these composed graphs to become symmetry presentations, we need to consider those symmetries induced by the structural congruence in Example 2.11. Because they are (hyper) graphs, the equations in (i), (ii) and (iii) of Examples 2.11 already hold, and similarly for the first equation of (iv). Other equations should be explicitly stated – their graphical presentations are easy to draw and hence are omitted. We can then capture the dynamics of the operation by the p-relation generated by Rule (1) of Figure 8. Notice how the geometric content of the dynamics is made clear in this presentation. (2) shows the dynamics corresponding to the usual π-calculus in Equation (2) of Example 2.11 (assuming b in the equation is not in $\mathscr{FN}(P)$ and omitting obvious correspondences).

Other interesting examples of the translation of the name-based constructions into the name-less ones can be found when we translate the synchronisation trees in Example 2.10 into their nameless counterpart. In particular, if we take the synchronisation trees of the π-calculus modulo, say, the weak bisimilarity following Section 2.3, we obtain an explicit representation of new name generation and reception as nameless correspondences,

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

exposing some aspects of name passing interaction in a geometric form. This structure has a further interest in that there is a certain simplification in the case of functional computation that induces categories of games studied in games semantics (Abramsky *et al.* 1994; Hyland and Ong 1994) – see Honda and Yoshida (1997) for an example of a reverse translation.

As an example of the translation in the other direction, we show how the multiplicative fragment of proof nets in Example 4.14 can be given a name-based presentation. The result is a precise syntactic representation of proof nets, which we can use to reason about their equalities *etc.* in a tractable way. For the translation, we use equations from Berry and Boudol's Chemical Abstract Machine (Berry and Boudol 1990) in the form given in Milner (1990).

Examples 5.6. (Proof nets, the name-based presentation) In the following translation, we first introduce terms that are the result of assigning names to a (non-rooted) p-set that arises as a free algebra generated from the (non-rooted) atoms and operations in Example 4.14 (so we translate nameless 'terms' to name-based terms). We then quotient those terms by the equations, which gives us the necessary abstraction, including symmetries. Terms are given as follows:

$$P \;::=\; \otimes(abc) \mid \wp(abc) \mid P \circ Q \mid 0 \mid [a = b]$$

where we assume names in each of $\otimes(abc)$, $\wp(abc)$ and $[a = b]$ are all pairwise distinct. The set of free names for each term, $\mathcal{FN}(P)$, is given by $\mathcal{FN}(\otimes(abc)) = \mathcal{FN}(\wp(abc)) = \{a, b, c\}$, $\mathcal{FN}([a = b]) = \{a, b\}$, $\mathcal{FN}(0) = \varnothing$, and $\mathcal{FN}(P \circ Q) = (\mathcal{FN}(P) \cup \mathcal{FN}(Q)) \setminus (\mathcal{FN}(P) \cap \mathcal{FN}(Q))$. Terms are then considered modulo α-conversion $\equiv_\alpha$, which is the congruence generated from

$$P \circ Q \;\equiv_\alpha\; P[\binom{\tilde{a}\tilde{c}}{\tilde{c}\tilde{a}}] \circ Q[\binom{\tilde{a}\tilde{c}}{\tilde{c}\tilde{a}}]$$

where $\{\tilde{a}\} \subset \mathcal{FN}(P) \cap \mathcal{FN}(Q)$, and names in $\tilde{c}$ are all fresh. Names in $\mathcal{FN}(P) \cap \mathcal{FN}(Q)$ are *bound* in $P \circ Q$. Renaming operations, then, act syntactically, avoiding the capture of names using $\equiv_\alpha$. These data give a rooted process structure. We then further quotient these terms by the congruence on terms (which by definition includes $\equiv_\alpha$) generated by

(i) $P \circ 0 \equiv P$, $P \circ Q \equiv Q \circ P$, and $(P \circ Q) \circ R \equiv P \circ (Q \circ R)$ when $\mathcal{FN}(P) \cap \mathcal{FN}(Q) \cap \mathcal{FN}(R) = \varnothing$.

(ii) $[a = b] \equiv [b = a]$ and $P \circ [a = b] \equiv P[\binom{ab}{ba}]$ when $a \in \mathcal{FN}(P)$ and $b \notin \mathcal{FN}(P)$.

We observe the last equation in (ii) gives the case when one side of a wire is connected to an edge of a net; since we simply recover the original net, the equation follows. This shows that the name-based presentation makes some of the implicit algebraic features in the name-less presentation explicit. Also notice that, as in this case, renaming and symmetries play a fundamental role in the definition. Finally, the dynamics is derived from the compatible closure of

$$\otimes(abc) \circ \wp(aef) \;\longrightarrow\; [b = f] \circ [c = e]$$

where we take the expressions modulo $\equiv$.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

The name-based presentation enables us to do precise algebraic manipulation of processes, for which we show a few examples. First, the following equation is easy to derive:

$$[a = b] \circ [b = c] \circ [c = a] \;\equiv\; [a = b] \circ [b = a].$$

Observe that the right-hand side cannot be made simpler. This corresponds to the *cyclic wire* discussed in Lafont (1995). Using this, we have

$$\otimes(abc) \circ \wp(abc) \;\equiv\; \otimes(abc) \circ \wp(ab'c') \circ [b = b'] \circ [c = c'] \;\longrightarrow\; [b = b'] \circ [b' = b],$$

which is another example discussed in Lafont (1995). As another interesting property, we can show

$$P \circ [a = b] \;\equiv\; P[\binom{ab}{ba}] \circ [a = b].$$

Indeed, if neither a nor b occurs free in P, there is nothing to prove: if only a (or b) occurs, we use the last rule of (iii) twice. If both occur free, we can use $\equiv_\alpha$.

5.3. *Proof of Theorem 5.1*

We now prove Theorem 5.1, showing that these translations offer bi-directional bridges between two theories. To do this, we should show that in each way of composing these translations, we always return to the structure that is isomorphic to the original one. For a smooth technical development, we fix a function $\langle\!\langle\, \cdot\, \rangle\!\rangle$ that, given any rooted p-set, gives its distinguished symmetry presentation. We then have the following proposition.

Proposition 5.7.

(i) For any $\mathbf{P}$ and $\mathbf{Q}$, $\mathbf{P} \simeq \mathbf{Q}$ if and only if $[\![\mathbf{P}]\!] \simeq [\![\mathbf{Q}]\!]$.
(ii) For any rooted p-set $\mathscr{P}$, we have $[\![\langle\!\langle \mathscr{P} \rangle\!\rangle]\!] \simeq \mathscr{P}$.
(iii) For any p-set $\mathbf{P}$, we have $\langle\!\langle [\![\mathbf{P}]\!] \rangle\!\rangle \simeq \mathbf{P}$.

Proof. For the 'only if' direction of (i), we translate the isomorphism in (non-rooted) p-sets into that of rooted ones, which is easy. (ii) is trivial. For (iii), given p-sets $\mathbf{P}$ and $\langle\!\langle [\![\mathbf{P}]\!] \rangle\!\rangle$, we first note that, by $[\![\cdot]\!]$, each $p \in \mathbf{P}$ is translated into a single orbit of rooted processes that includes $P = [\langle \Psi,\ p \rangle]_\approx$, for which $\mathscr{S}(p)$ and $sym(P)$ are conjugate by Ψ. Then $\langle\!\langle \cdot \rangle\!\rangle$ maps this to $sym(P')$, where P and P' are in the same orbit, which is, hence, conjugately isomorphic to $sym(P)$. Thus we have now constructed a bijection between processes that induces a conjugate isomorphism for each pair. Finally, these together imply the 'if' direction of (i). □

We observe the following key property of the translation in Definition 5.4.

Proposition 5.8. For a compatible relation $\mathscr{R}$, we have $\mathscr{F}_2 \circ \mathscr{R}^{\circ\star} \circ \mathscr{F}_1 = \mathscr{R}$ where $\mathscr{F}_{1,2}$ are isomorphisms. Similarly, for a p-relation $\mathfrak{R}$, we have $\mathfrak{R}_2 \circ \mathfrak{R}^{\star\circ} \circ \mathfrak{R}_1 = \mathfrak{R}$ where $\mathfrak{R}_{1,2}$ are isomorphisms.

Proof. The mediating isomorphisms are given by those that relate the object-level translations in Proposition 5.7. The statement is then verified by first showing that $(\,\cdot\,)^\circ$ and $(\,\cdot\,)^\star$ inversely relate the compatible closure of a single tuple, on the one hand, and

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

an abstract correspondence on the other hand, up to isomorphisms. The second stage is then to verify that both maps are continuous with respect to the arbitrary join of (p-) relations. The first part is easy when you notice that the composition with isomorphisms can be presented by the composition given in Equation (3) of Discussion 4.7, while the second part is direct from the definition of the translations. □

We can now conclude the proof of Theorem 5.1 by the following results.

Proposition 5.9.

(i) If $\mathscr{F}$ is a homomorphism (of rooted p-sets), then $\mathscr{F}^\circ$ is a p-map.
(ii) If $\mathfrak{F}$ is a p-map, then $\mathfrak{F}^\star$ is a homomorphism.
(iii) Given homomorphisms $\mathscr{F}_1 : A \to B$ and $\mathscr{F}_2 : B \to C$, we have $(\mathscr{F}_2 \circ \mathscr{F}_1)^\circ = \mathscr{F}_2^\circ \circ \mathscr{F}_1^\circ$. Similarly, given p-maps $\mathfrak{F}_1 : \mathbf{P} \to \mathbf{Q}$ and $\mathfrak{F}_2 : \mathbf{P} \to \mathbf{Q}$, we have $(\mathfrak{F}_2 \circ \mathfrak{F}_1)^\star = \mathfrak{F}_2^\star \circ \mathfrak{F}_1^\star$.

The proof of the proposition, which is mechanical, is relegated to Appendix B. This in particular shows that the three defining conditions for p-maps, *cf.* Definition 4.8, precisely give rise to the properties of its translation being functions. The simpler presentation of the composition in the case of p-maps, *cf.* Proposition 4.10 (i), makes the verification of the translation of the categorical composition easier than the general case of p-relations. This gives the equivalence between RPS and PS. We then note, as shown in Appendix A, that RPS_{rel} and PS_{rel} are the canonical relational universes corresponding to RPS and PS, respectively, so that the equivalence between the latter two immediately leads to the equivalence between RPS_{rel} and PS_{rel}. We have thus established Theorem 5.1, answering the question we posed at the end of Section 3.

6. Rooted and non-rooted presentations at work

6.1. *Relating two forms of parallel composition*

In this section we give a small application of the present theory. In both theory and practice, we can find several basic ways of composing processes in parallel. Amongst others, two forms are widely used: one is based on 'closed' connection (for example, proof nets), while the other is based on 'open' connection (for example, π-calculus). Now, it has often been argued that each is representable in the other, that is, 'private' connection can be represented by the combination of an 'open' connection and hiding, *cf.* Milner (1980), and, conversely, we can represent the latter by the former by introducing 'sharing' agents, *cf.* Kahn and MacQueen (1977) and Bawden (1986). However, there do not seem to be any formal statements on such embeddings, and the suggested ideas always pertain to concrete formalisms.

One difficulty in such discussions is that there is no common mathematical framework in which we can even *formulate* the problem with sufficient generality, for example, without referring to specific formalisms. Clearly we need to start by placing these two forms of parallel composition in a general setting of algebra over processes, then relate them via certain mappings. In the following we apply the theories we have developed in the preceding sections just for this purpose. As we shall soon discuss, the problem can

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

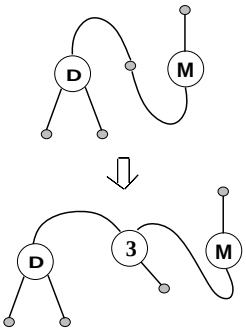

Fig. 9. Using 3

be elucidated using a geometric presentation, with a specific form of wires to represent sharing. This aspect is best understood in the context of nameless processes, where the idea of connection is exposed in an essential way. At the same time, the precise form of embeddings and its correctness are best addressed in the name-based formalism, especially the algebraic calculations that we need for the establishment of the result.

Before going into technical discussions, we will illustrate the basic ideas of the translations. First, the embedding of 'closed connection' into 'open connection' can already be found in Milner (1980), and is relatively straightforward – we represent a closed connection of P and Q, say $P \circ Q$, by $(\nu A)(P|Q)$, where (νA) is the name hiding (with $A = \mathcal{FN}(P) \cap \mathcal{FN}(Q)$) and $|$ is the open composition. For the other direction, which is a little more complex, we use an agent with three handles, written 3, for decomposing a shared handle by open connection into a sharing agent together with closed connection (see Kahn and MacQueen (1977), Bawden (1986) and Gardner (1995) for related ideas). As an example, assume we have two (nameless) processes, one called M (with two handles) and another called D (with three handles), for which we assume trivial symmetries and that they share one handle by open composition (so the composed handle is still open to further connection). The situation is depicted in the upper part of the following figure. This composition is transformed into closed compositions as given in the lower part of the figure.

Here the agent 3 is intended to *equate* the three places it is connected to, so that its presence as in the second picture above is the same thing as collapsing three handles into one, which is the situation in the first picture. This suggests that 3 induces non-trivial symmetries. In the following we will formalise this construction. In the spirit of Section 5, we shall often be guided by geometric insights gained by the name-less presentation, while the exact description and reasoning will be done using the name-based presentation.

6.2. *Two abstract algebras*

We first define a general notion of 'process algebra with open connections'. We incorporate the wires that we have already encountered in the essential π-calculus. As we shall discuss

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

soon, the incorporation of wires does not lose us any generality, since any process algebra with open connection can be conservatively extended with wires.

Definition 6.1. (Basic open P-algebra) A basic open P-algebra, which we will often simply call an *o-algebra*, is a rooted process set with some operations and a compatible relation called *dynamics*, which is given as follows. First we define the set of terms by

$$P \ ::= \ \alpha_i^{n_i}(a_1 \dots a_{n_i}) \ \mid \ P|Q \ \mid \ (\nu a)P \ \mid \ [a = b] \ \mid \ [a] \ \mid \ 0$$

where we assume the set $\{\alpha_i^{n_i}\}$ of *atoms* with arity n_i, $a_i \neq a_j$ for $i \neq j$. Terms of form $\alpha(\tilde{a})$ as well as $[a = b]$ and $[a]$ are called *atomic processes*, which are together denoted by $\alpha(\tilde{a}), \alpha'(\tilde{b}), \dots$. The treatment of $(\nu a)P$ and α-equality is the same as in Example 2.11. Terms are always understood modulo $\equiv_\alpha$. Next we introduce the relation $\equiv$ over terms, which is a congruence based on the rules and axioms of Example 2.11 (i)...(v), possibly with additional axioms of the form $\alpha_i(\tilde{a}_i) \equiv \alpha_i(\sigma(\tilde{a}_i))$ where σ is a non-trivial permutation over $\tilde{a}_i$ (additional axioms are for generating symmetries on atoms). The *dynamics* is a binary compatible relation closed under the two operations given above, understood to be defined on the quotient set, which should be generated by rules each of the form $\prod_i \alpha_i(\tilde{a}) \longrightarrow R$, where $\prod_i \alpha_i(\tilde{a}_i)$ is the parallel composition of a finite number of atomic processes by $|$, with conditions:

(i) $\mathscr{FN}(\prod_i \alpha_i(\tilde{a}_i)) \supset \mathscr{FN}(R)$.
(ii) $\prod_i \alpha_i(\tilde{a}_i)$ is *connected*, that is, for any non-trivial partition I_1, I_2 of I, there exists $i \in I_1$ and $j \in I_2$ such that $\mathscr{FN}(\alpha_i(\tilde{a}_i)) \cap \mathscr{FN}(\alpha_j(\tilde{a}_j)) \neq \varnothing$.
(iii) $\prod_i \alpha_i(\tilde{a}_i)$ at least includes two atomic processes.

An o-algebra is often denoted by $A_{\mathscr{P}}$ where $\mathscr{P}$ is the set of *processes*, that is, terms modulo $\equiv$. $P, Q, R, \dots$ denote either processes or terms, depending on contexts. A *wireless o-algebra* is an o-algebra that may not have atoms $[a = b]$ and $[a]$ and that are without the equations in (iv) and (v) of Examples 2.11, with the same notion of dynamics. To emphasise, we often use the term a *wired o-algebra* to denote a standard o-algebra.

It is easy to see that any wireless o-algebra can be conservatively extended to a wired one, in the sense that there is a base monomorphism going from the original algebra to the extended one and that preserves and reflects all operations and dynamics. From another perspective, wired o-algebras are a special case of wireless ones, if we extend the latter by closing the present class under all possible quotienting. Another notable point is that, if a formalism is equipped with non-injective substitutions, this can be precisely captured by having wires, as in Example 2.11.

Examples 6.2.

(i) Immediately, the essential π-calculus in Example 2.11 is a (wired) o-algebra.
(ii) π-calculi without summation are (wireless) o-algebras, considering prefixed agents as atoms in the former. The standard reduction relation gives dynamics. In the case of the combinatory versions of π-calculi studied in Honda and Yoshida (1994a), Honda and Yoshida (1994b), Yoshida (1998), and Raja and Shymasundar (1994), their atomic agents precisely correspond to those in the present sense. We can further

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

conservatively extend them to wired o-algebras, with the substitution represented as in Example 2.11.

(iii) For calculi with summations, we can view each sum as an atomic process (thus neglecting + as an operation), then the resulting structure is an o-algebra. Notice that in this case the number of atoms is infinite. CCS, π-calculi with summation, and ACP are thus o-algebras, taking τ actions as their dynamics. In addition, various parallel programming languages often have, or can be reformulated to be, such structures.

Some basic properties of wires in open connection are worth noting.

Proposition 6.3.

(i) $[a = b]|[b = c] \;=\; [a = c]|[b = c] \;=\; [a = b]|[a = c]$.

(ii) If $a \in \mathscr{FN}(P)$ and $b \notin \mathscr{FN}(P)$, then $(va)(P|[a = b]) = P[\binom{ab}{ba}]$.

(iii) Let P_1, P_2, R be processes in o-algebra with wires, and suppose $P_1|P_2 \longrightarrow R$. Suppose $a \in \mathscr{FN}(P_1)\backslash\mathscr{FN}(P_2)$ and $b \in \mathscr{FN}(P_2)\backslash\mathscr{FN}(P_1)$, and let c be fresh. Then we have

$$P_1|P_2[\binom{ba}{ab}] \;\longrightarrow\; (vb)(R \mid [a = b]).$$

Proof. (i) and (ii) are easy. Using (ii), we first note that

$$P_1|P_2[\binom{ba}{ab}] \;\equiv\; (vb)(P_1 \mid P_2 \mid [a = b]).$$

Then we apply the compatibility of $\longrightarrow$ to get (iii). □

We next introduce a class of algebras based on closed connection.

Definition 6.4. (Basic closed P-algebra) A *basic closed P-algebra*, which we will often just call a c-algebra, is given by the following data. Terms are given by the grammar

$$P \;::=\; \alpha_i^n(a_1 \ldots a_n) \mid P \circ Q \mid 3(a_1a_2a_3) \mid 2(a_1a_2) \mid 1(a) \mid 0$$

where we assume we are given the set $\{\alpha_i^{n_i}\}$ of *atoms* (n_i of $\alpha_i^{n_i}$ is called its *arity*). In each of $\alpha_i^n(a_1 \ldots a_n)$, $3(a_1a_2a_3)$, $2(a_1a_2)$ and $1(a)$, we assume $a_i \neq a_j$ if $i \neq j$. The operation $\circ$ induces the binding as in Example 5.6. Processes of the form $\alpha(\tilde{a})$, including $1(a), 2(ab)$ and $3(abc)$, are called *atomic processes*. We quotient the terms by the congruence generated by Example 5.6 (i) and (ii), together with:

(i) $1(a) \circ 1(a) = 0$, $1(a) \circ 3(abc) = 3(bee') \circ 3(ee'c) = 2(bc)$, and $3(abc) \circ 3(cde) = 3(adc) \circ 3(cbe)$.

(ii) $P \circ 3(abc) = P[\sigma] \circ 3(abc)$ where σ is a permutation on $\{a, b, c\} \cap \mathscr{FN}(P)$.

(iii) $P \circ 2(ab) = P[\binom{ab}{ba}]$ where $b \notin \mathscr{FN}(P)$,

as well as the additional axioms for atomic processes as in Definition 6.1. The *dynamics* is a binary relation on the quotient processes generated by rules of the form $\prod_i \alpha_i(\tilde{a}_i) \longrightarrow R$, where $\prod_i \alpha_i(\tilde{a}_i)$ is the $\circ$-composition of a finite number of atomic processes, for which we assume the same conditions as given in (i) (ii) (iii) of Definition 6.1. We use the term *wireless c-algebra* to mean the algebra that may not have all of $1(a)$, $2(ab)$ and $3(abc)$,

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

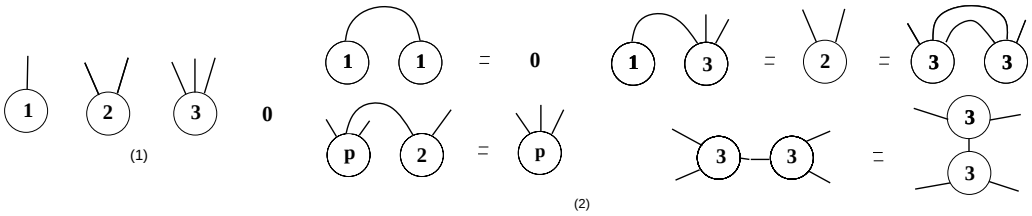

Fig. 10. c-algebra

and, accordingly, is without Equations (i). . . (iii) above. Again, to emphasise, we use the expression *wired c-algebras* to denote standard c-algebras.

The key idea in the above construction is, as we noted at the beginning, to have an agent that 'equates three locations', denoted 3(*abc*). 2(*ab*) equates two locations and acts as a usual wire: 1(*a*), on the other hand, equates none and functions as the name restrictor in the present setting. We shall see later how these constructs can faithfully embed the operations of o-algebras in a coherent way. Figure 10 (1) gives the name-less presentation of four basic atoms of a c-algebra, assuming the full symmetries in the case of 3 and 2. The equations in (i) are given in the name-free form in Figure 10 (2), where we omit the evident correspondences.

As in the case of o-algebras, we can always conservatively extend a wireless c-algebra into a wired one. We observe, however, that having the agent 3(*abc*) means the original possibly confluent reduction may become non-confluent, so that the property of dynamics as a whole may change by its incorporation (even though the original agents have exactly the same dynamics). Proof nets, interaction nets, and varied net-based presentations of λ-calculi give c-algebras when augmented with 3 and 1. It is also notable that any set-theoretic Ω-algebra can be presented using a partial counterpart of c-algebra.

We are now ready to state our main result of the present section. Below, and in the rest of the paper, we use the expression *base isomorphisms* to denote isomorphisms at the level of rooted process sets, to avoid confusion with the ones at the algebraic level.

Theorem 6.5. There is a bijection between the class of o-algebras and the class of c-algebras such that there exists a base isomorphism between each pair of related algebras that, moreover, preserves and reflects dynamics.

The theorem says there is a faithful embedding of one class of algebras in another in both directions. Since the embeddings accompany base isomorphisms, we have that the free names and symmetries of a source process and its translation coincide. Because signatures are different, we cannot have the isomorphism at the level of algebras: with this inevitable limitation, the above theorem states the strongest possible embedding result that faithfully preserves all possible properties of processes, including the locality of dynamics. Also note that the two embeddings are easily extended to the case in which the source algebras are taken as their wireless counterpart, because such structures can always be conservatively extended to wired ones, as we have already noted.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

6.3. *The embedding and its correctness*

In the rest of this section, we validate Theorem 6.5 by introducing the necessary constructions one by one. We detail the proof for the translation from o-algebras to c-algebras, which is more difficult, and briefly discuss the case of the translation in the other direction. Both the geometric insights from the nameless presentation and the precise algebraic manipulation possible in the name-based presentation are indispensable. The notion of symmetries plays a key role for the establishment of the result.

We start from a basic lemma.

Lemma 6.6. Given a c-algebra, write $P|'Q$ for: $P\{c_1 \dots c_n/a_1 \dots a_n\} \circ \prod_i 3(c_i a_i b_i) \circ Q\{b_1 \dots b_n/a_1 \dots a_n\}$, where $\{a_1, \dots, a_n\} = \mathcal{FN}(P) \cap \mathcal{FN}(Q)$ and b_i, c_i are all fresh and distinct (thus $P|'Q$ is determined uniquely up to $\equiv_\alpha$). Also write $(\nu' a)P$ for P if $a \notin \mathcal{FN}(P)$, else $1(a) \circ P$. Then we have:

(i) $\mathcal{FN}(P|'Q) = \mathcal{FN}(P) \cup \mathcal{FN}(Q)$ and $\mathcal{FN}((\nu' a)P) = \mathcal{FN}(P) \setminus \{a\}$.
(ii) $P|'0 \equiv P$, $P|'Q \equiv Q|'P$, and $(P|'Q)|'R \equiv P|'(Q|'R)$.
(iii) $P \circ Q \equiv (\prod 1(a_i)) \circ (P|'Q)$ where $\{a_1 \dots a_n\} = \mathcal{FN}(P) \cap \mathcal{FN}(Q)$.
(iv) $(\nu' ab)P \equiv (\nu' ba)P$ and $(\nu' a)P|'Q \equiv (\nu' a)(P|'Q)$ if $a \notin \mathcal{FN}(Q)$.
(v) $2(ab)|'2(ab) \equiv 2(ab)$.
(vi) $2(ab)|'2(bc) \equiv 2(ac)|'2(bc) \equiv 2(bc)|'2(ab) \equiv 3(abc)$.
(vii) $2(ab)|'P \equiv 2(ab)|'P[\binom{ab}{ba}]$.
(viii) $1(a)|'P \equiv P$ when $a \in \mathcal{FN}(P)$.

Proof. See Appendix B. □

Notice that the operation $|'$ in the above Lemma gives the translation of open composition into closed composition using 3, which we discussed in Section 6.1. Using this derived operation, we can now introduce the embedding.

Definition 6.7. Given an o-algebra $A_{\mathscr{P}}$, let its set of atoms be $\{\alpha_i^{n_i}\}$ and its dynamics $\longrightarrow$. Then we form an algebra $A'_{\mathscr{P}'}$ by the following rules:

(i) Terms are formed from $\{\alpha_i\}$ together with $1, 2, 3$. We write $P|'Q$ for the expression we defined in Lemma 6.6. Then we define the function $[\![P]\!]$ from terms of $A_{\mathscr{P}}$ to those of $A'_{\mathscr{P}'}$, both taken modulo $\equiv_\alpha$, as follows:
   — $[\![0]\!] = 0$, $[\![\alpha_i(\tilde{a})]\!] = \alpha_i(\tilde{a})$ where $\alpha_i(\tilde{a})$ is not of form $[a = b]$ or $[a]$, $[\![[a = b]]\!] = 2(ab)$ and $[\![[a]]\!] = 1(a)$.
   — $[\![P|Q]\!] = [\![P]\!]|'[\![Q]\!]$. If $a \in \mathcal{FN}(P)$, $[\![(\nu a)P]\!] = 1(a) \circ [\![P]\!]$. Otherwise, $[\![(\nu a)P]\!] = [\![P]\!]$.
(ii) The equations are those of the c-algebras. The resulting quotient is written $\mathscr{P}'$.
(iii) Dynamics is generated by, whenever $\prod \alpha_i \longrightarrow P$ is a rule in $A_{\mathscr{P}}$, we set $[\![\prod \alpha_i]\!] \longrightarrow [\![P]\!]$.

Proposition 6.8.

(i) $[\![P[\sigma]]\!] \equiv_\alpha [\![P]\!][\sigma]$.
(ii) $\mathcal{FN}(P) = \mathcal{FN}([\![P]\!])$.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

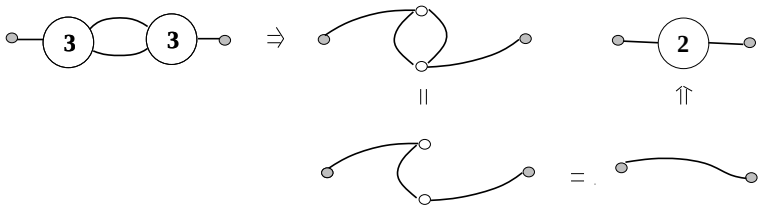

Fig. 11. equational correspondence

(iii) $A'_{\mathscr{P}'}$ of Definition 6.7 (i) is a c-algebra.

Proof. (i) is mechanical. (ii) is by Lemma 6.6 (i) and by $1(a) \circ P = \mathscr{FN}(P)\backslash\{a\}$ when $a \in \mathscr{FN}(P)$. For (iii) we only have to check dynamics. But it is easy to see that if $\prod \alpha_i$ conforms to (i)(ii)(iii) of 6.1, then $[\![\prod \alpha_i]\!]$ is again connected and cannot be a singleton, and satisfies the free name condition, which comes from (i) of the present Proposition. □

We now verify the key property of the mapping. In the proof, we use the translation in the reverse direction.

Proposition 6.9. $P \equiv Q$ iff $[\![P]\!] \equiv [\![Q]\!]$.

Proof. The 'only if' case is by rule induction, using Lemma 6.6. For the 'if' direction, we introduce the following translation from $\mathscr{P}'$ to $\mathscr{P}$.

— $0^\star \stackrel{\text{def}}{=} 0$, $\alpha_i(\tilde{a})^\star \stackrel{\text{def}}{=} \alpha_i(\tilde{a})$ (except α_i is among $1, 2, 3$), $3(abc)^\star \stackrel{\text{def}}{=} [a = b][b = c]$, $2(ab)^\star \stackrel{\text{def}}{=} [a = b]$, and $[\![1(a)]\!] \stackrel{\text{def}}{=} [a]$.

— $(P \circ Q)^\star \stackrel{\text{def}}{=} (\nu\mathscr{FN}(P) \cap \mathscr{FN}(Q))(P|Q)$.

We now show:

(i) $(P[\sigma])^\star \equiv_\alpha P^\star[\sigma]$, $\mathscr{FN}(P) = \mathscr{FN}(P^\star)$, and $P \equiv Q \quad \Rightarrow \quad P^\star \equiv Q^\star$

(ii) $[\![P]\!]^\star \equiv P$

from which the statement is immediate. For (i), the first two are easy. For the embedding of $\equiv$, we show a few insightful cases. Below, we will often simply write $P = Q$ for $P \equiv Q$. For associativity assume $\mathscr{FN}(P) \cap \mathscr{FN}(Q) \cap \mathscr{FN}(R) = \varnothing$, and $\tilde{a} = \mathscr{FN}(P) \cap \mathscr{FN}(Q)$ and $\tilde{b} = (\mathscr{FN}(P) \cup \mathscr{FN}(Q)) \cap \mathscr{FN}(R)$. Notice that $\{\tilde{a}\} \cap \{\tilde{b}\} = \varnothing$. Then $(P \circ Q \circ R)^\star \stackrel{\text{def}}{=} (\nu\tilde{b})((\nu\tilde{a})((P|Q)^\star)|R^\star) = (\nu\tilde{a}\tilde{b})(P^\star|Q^\star|R^\star)$, and similarly for the right-hand side. For the equations in Definition 6.4, we have

$$
\begin{aligned}
(1(a) \circ 3(abc))^\star &\stackrel{\text{def}}{=} (\nu a)([a]|[a = b]|[b = c]) = (\nu a)[b = c] = [b = c] \\
(3(aef) \circ 3(efb))^\star &\stackrel{\text{def}}{=} (\nu ef)([a = e]|[e = f]|[e = f]|[f = b]) \\
&= (\nu ef)([a = e]|[e = f]|[f = b]) \\
&= (\nu e)[a = b] \quad = \quad 2(ab)^\star.
\end{aligned}
$$

We now give the name-free account of the second series of equations in Figure 11 (omitting obvious correspondences). Notice how this analyses (or co-analyses) the original equation. Finally, by Proposition 6.3 (i), we have $P \circ 2(ab) = P[\binom{ab}{ba}]$ with $b \notin \mathscr{FN}(P)$, thus we

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

know (i) holds. (ii) is by induction: the base cases are trivial and the inductive step is verified by the following calculation:

$$\begin{aligned}
[\![(P|Q)]\!]^\star &\stackrel{\text{def}}{=} ([\![P]\!]\{\tilde{b}/\tilde{a}\} \circ [\![Q]\!]\{\tilde{c}/\tilde{a}\} \circ \prod 3(b_i c_i a_i))^\star \\
&= (\nu \tilde{b}\tilde{c})(([\![P]\!]\{\tilde{b}/\tilde{a}\})^\star | ([\![Q]\!]\{\tilde{c}/\tilde{a}\})^\star | \prod([b_i = a_i]|[c_i = a_i])) \\
&= [\![P]\!]\{\tilde{b}/\tilde{a}\}^\star \quad = \quad P|Q. \\
[\![(\nu a)P]\!]^\star &\stackrel{\text{def}}{=} 1(a) \circ [\![P]\!]^\star \quad = \quad (\nu a)([a]|[\![P]\!]^\star) \quad = \quad (\nu a)(P).
\end{aligned}$$

(in the latter we have assumed $a \in \mathscr{FN}(P)$ – if not the result is trivial). This concludes the proof. □

Corollary 6.10. *$sym(P) = sym([\![P]\!])$ where P is considered modulo $\equiv$.*

Proof. The statement is equivalent to $P[\sigma] \equiv P$ iff $[\![P[\sigma]]\!] \equiv [\![P]\!]$. □

Proposition 6.11. *$P \longrightarrow Q$ iff $[\![P]\!] \longrightarrow [\![Q]\!]$.*

Proof. Suppose $P \longrightarrow Q$. Then we can write $P \equiv C[P_0]$ and $Q \equiv C[Q_0]$ such that $P_0 \longrightarrow Q_0$ is one of the rules. Thus $[\![P_0]\!] \longrightarrow [\![Q_0]\!]$. Observing that $[\![C[R]]\!]$ can be written $C'[[\![R]\!]]$ regardless of R, we know $[\![C[P_0]]\!] \longrightarrow [\![C[Q_0]]\!]$, but by Proposition 6.9 this implies $[\![P]\!] \longrightarrow [\![Q]\!]$, as required. For the converse, we first show $P \longrightarrow Q$ in the c-algebra implies $P^\star \longrightarrow Q^\star$, using the same reasoning. Moreover, again using Proposition 6.9, we have $[\![\longrightarrow]\!]^\star = \longrightarrow$, which shows $[\![P]\!] \longrightarrow [\![Q]\!]$ implies $P \longrightarrow Q$, and hence we are done. □

Because the above proof also shows that $[\![P]\!] \longrightarrow P'$ implies $P' \equiv [\![Q]\!]$ for some Q, and the proof of Proposition 6.9 presents the inverse translation $(\,\cdot\,)^\star$ and shows in effect that it satisfies the required properties, we have now established Theorem 6.5.

7. Discussion

7.1. *Substitutions*

One notion that often accompanies name-based formalisms, such as π-calculus, is *substitution*, or non-injective renaming. We first notice that semantic or syntactic structures closed under name substitutions are by definition already closed under injective renaming, thus inducing a rooted p-set. We also observe that the status of substitution and that of injective renaming are quite different. First of all, name substitution is in general an irreversible operation, in the sense that there is no inverse operation that cancels the effect of the operation. Related to this is the fact that when we perform a name substitution on a given process, we cannot tell what the symmetries of the resulting process will be just by looking at those of the original process (for example, given $\overline{a}b|ac|\overline{e}f|gh$, the substitution $\{e/g\}$ increases a symmetry, while $\{e/h\}$ does not). These points indicate that the substitution is not as semantically neutral as renaming: it is best thought to be a specific kind of algebraic operation. Related to this is the fact that in many formalisms, substitutions are

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

partial operations, as in typed π-calculi (Milner 1992; Pierce and Sangiorgi 1993; Honda 1996).

There are a few ways to incorporate the non-injective renaming in the name-free theory. One of the ways, which is pioneered in the treatment of the π-calculus in action structures (Milner 1993) and which we have used in preceding sections, is to incorporate wires, or links, into the formalism, see Example 2.11 in Section 2.3. A drawback of this formulation is that the original syntactic formalism, such as the π-calculus, is in general based on direct substitution rather than such linking agents: we need to reformulate the notion just as we did in Example 2.11. At the same time, even at the term level, the reformulation does preserve the original algebra and dynamics modulo added equations, so we do not lose any essential information. A merit of this construction is that, because the approach represents substitutions just as standard process-theoretic operations (in the sense of the present theories), it can be easily extended to a more refined setting, a representative case being when processes are typed and thus we cannot collapse certain handles, *cf.* Milner (1992), Pierce and Sangiorgi (1993) and Honda (1996). It would also be interesting to see how the same scheme may apply to functional formalisms with name substitutions, such as Pitts–Stark's *v*-calculus (Stark 1995).

7.2. *Comparisons and further issues*

Our work can be considered as offering a basic building block for diverse process theories. As such, it relates to various proposals for the general semantic framework of concurrent and interactive computing. We give some comparative discussions below.

(i) As we mentioned in the Introduction, Milner's Action Structures, initially presented in Milner (1993) and studied by him and by other researchers, *cf.* Mifsud (1996), Mifsud *et al.* (1996) and Gardner (1995), gives a semantic framework for general concurrent computation using a class of strict symmetric monoidal categories. Here arrows denote both name-based and name-free processes (called actions), the tensor gives their name-based 'open' composition (in the sense of Section 5), while arrow composition gives 'closed' parallel composition. The dynamics is treated by interaction rules inducing a preorder in each homset, which is given a refined treatment by specifying 'control', which determines the timing of activation of some processes. The resulting structures are called *control structures*, and, through their study, many topics have arisen that are closely related to the present work. In particular, a notion of interface of processes (called *surface*) was studied in Mifsud *et al.* (1996) and Mifsud (1996), and further generalised by Milner (Milner 1995b). An important aspect of the study in Milner (1995b) is that, in the generalised framework, names can be composite entities (like an address in the street), as long as certain algebraic conditions are satisfied. It would be interesting to see whether the structure of an interface thus formulated has a corresponding nameless representation as in Section 4. Another closely related study in this context is Gardner's work on a name-free reformulation of certain syntactic concrete structures called action calculi (Gardner 1995), which reduces the open connection of tensors into the closed connection of arrow composition. She used a version of wires that are closely related to those we used in Section 6. Also,

in Milner (1994), a graphical presentation of the π-calculus called *π-nets* is given with the motivation of elucidating geometric aspects of the π-calculus. Among other constructions, a 'torpedo' in π-nets corresponds to an n-ary sharing agent in terms of our algebraic framework in Section 6, combined with information on data flow.

(ii) Abramsky *et al.* (1995) presents a framework of typed semantics for concurrent processes called interaction categories, where the representation of processes is based on nameless presentation, setting each process to be an arrow in a certain category. Using synchronisation trees as the representation of processes, categorical structures of various process specifications are studied, including trace-based ones and more refined ones with liveness conditions. The abstraction of name-based processes as arrows in interaction categories is to some extent parallel to the translation in Section 6, even though the structures of nameless processes as such are not articulated. It would be interesting to know how some notions elucidated in the present theory, such as symmetries, can have significance in the fine categorical structures studied in their framework. In Abramsky (1996), Abramsky also studied a class of categories based on algebraic abstraction of Girard's geometry of interaction (Girard 1988; Girard 1989; Girard 1995b), in which arrows are again essentially a nameless presentation of interacting processes, particularly in those instances that are studied in so-called game-based semantics (Abramsky *et al.* 1994; Hyland and Ong 1994) (see also our discussions after Example 5.5). It is worth clarifying how various categorical constructions can be re-interpreted in the framework of the 'algebra of processes' in the sense of Section 4. Related to this, Danos and Regnier (Danos and Regnier 1993; Danos and Regnier 1993) have studied algebras of information flow for the dynamics of λ-calculus seen as interacting processes in a graphical format. Their work uses injective partial maps as carriers of their algebra, even though the meaning of their partial maps differs from ours (since theirs represent data on the abstract information flow associated with each interface rather than correspondence between interfaces). In spite of this difference, we may reformulate their constructions using the framework of Section 4, which would offer a useful analysis of a basic setting of their constructions.

(iii) Berry and Boudol (Berry and Boudol 1990) introduced *chemical abstract machines* based on the metaphor of chemical solutions, to capture various concurrent computations. The essential idea is to regard processes as forming a multiset, or equivalently a commutative monoid with a unit, which is often utilised in an equational form given in Milner (1990). While the present theory captures the case when the processes may not form a multiset, their constructions give the fundamental machinery in turning many graphical formalisms into the name-based format, as we saw in Section 6.

(iv) Meseguer (Meseguer 1992) has studied *conditional concurrent term rewriting systems*, which extend the standard algebraic semantics to non-deterministic computation. Meseguer (1992) shows that diverse examples, such as Turing Machines, λ-calculi and actors, can be treated in the framework. In particular, in the context of the embedding of actors, Meseguer (1992) discusses how names are treated in his framework, including the generation of new names. It would be possible to shed another light on

the role of names in computation through such a representation. At the same time, notions such as 'wires' may not be easily treatable in his framework.

Finally, in the footnote to Section 2.2, we referred to a work by Fraenkel and Mostowsky early in this century that gives a construction of a set-theoretic universe starting from atoms and their permutation (*cf*., for example, Kunen (1980, Chapter IV, Exercise 24) or Jech (1977, Section 5)), communicated to us by Andrew Pitts at the proof reading stage. Their sets with permutation are basic instances of rooted p-sets, and their set-theoretic operations conform to those of rooted p-sets given in Section 2. The construction indicates how a general set-theoretic universe can be formed purely starting from atoms – or, in the present context, names – and their permutation. Pitts and Gabbay (1999) used this construction for an analysis of binding, regarding the binding as a renaming-closed operation (as we did in Section 2 of the present paper). As we discussed in the Introduction, sets with renaming and their algebra naturally arise from the usage of names and identifiers in theories of computing, most notably in theories of concurrent processes where names are used to represent points of interaction, which is a vital element of interacting processes. Combined with the alternative geometric theory developed in Section 4, we hope that our constructions and results in the present work can be used as cornerstones of the further study of the syntax and semantics of interacting processes, and, more broadly, of general computing systems.

Appendix A. Basic properties of RPS and RPS_{rel}

A.1. RPS *is a topos*

In this Appendix we give a basic categorical characterisation of RPS and RPS_{rel} by showing that it is a topos and that RPS_{rel} is its corresponding relational universe. While the development is rather technical, it is important to know that the present construction induces a topos, since it means that the universe is automatically equipped with all set-theoretic operations following the standard constructions. It is notable that all our concrete operations in Section 2 conform to standard categorical constructions (which, in particular, ensure the uniqueness of these constructions up to isomorphisms, as well as that they all work with each other consistently). The technical development is also closely related to our treatment of symmetries in Section 3.

We start from an alternative characterisation of named process sets. Given a topological group G, a *continuous G-set* is a set X equipped with a continuous G-action on X: that is, assuming the discrete topology on X, we want the action $S_{\mathcal{N}} \times X \to X$ to be continuous.

Proposition 1. View $S_{\mathcal{N}}$ as a topological group induced by the subspace topology of $\prod_{a \in \mathcal{N}} \mathcal{N}_a$ (the latter given the usual product topology) and let X be a continuous $S_{\mathcal{N}}$-set. Then we have:

(i) If X is a continuous $S_{\mathcal{N}}$-set, any $x \in X$ has the minimum finite support, where a *support of* x is $A \subset \mathcal{N}$ such that $\forall a \in A.\sigma(a) = a$ implies $\sigma \cdot x = x$. Then X is a named process set, taking the minimum support as $\mathcal{FN}(x)$.

(ii) Any rooted process set is a continuous $S_{\mathcal{N}}$-set with respect to the induced group action.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

Proof. The characterisation by the minimum support property is well known (see, for example, Stark (1995)), so we only show the outline here. The key observation is that a base for open sets containing $id_{\mathcal{N}}$ in $S_{\mathcal{N}}$ can be given as the collection of sets each of form $O_A \stackrel{\text{def}}{=} \{\sigma \mid \forall a \in A.\sigma(a) = a\}$ for a finite subset of A, which is easy using the construction of the product topology. Now continuity is the same thing as the isotropy group of each x being open (notice the inverse image of x by the action consists of the isotropy group and its conjugates, so if the isotropy group is open, then this image is also: conversely if the image is open, then take its intersection with $S_{\mathcal{N}} \times \{x\}$, which is clearly open). But the isotropy group surely contains $id_{\mathcal{N}}$, and, by the closure under multiplication, if O_A and O_B are in the isotropy group, then so is $O_{A\cap B}$, thus showing (i). Conversely if P has a finite support A, then the isotropy group is nothing but O_A, which is by definition open. □

By a well-known result in topos theory, *cf.* Mac Lane and Moerdijk (1992, III-9), we immediately get the following corollary.

Corollary 1. RPS is a topos.

By the corollary we have (a categorical analogue of) all basic set-like constructions in RPS. Another significance of the corollary is that it leads to the standard construction of RPS_{rel} from RPS. The following is from Freyd and Scedrov (1990) and McLarty (1992). Let C be a topos. Then we can define a 'relation' in C as a monic pair. If $f_1 : R \to A, f_2 : R \to B$ and $g_1 : S \to B, g_2 : S \to C$ are two such 'relations' (so R and S objectify relations), their *composition* is given by:

(1) first take the pullback of f_2 and g_1, say $e_1 : T_0 \to R$ and $e_2 : T_0 \to S$,
(2) next let h be the universal arrow $h = \langle f_1 \circ e_1, f_2 \circ e_2\rangle : T_0 \to A \times C$, and
(3) epi-mono decompose h, say as $q : T_0 \to T$ and $i : T \to A \times C$, the latter being the smallest subobject of h and then q is automatically epi.

Then $\pi_1 \circ i : T \to A$ and $\pi_2 \circ i : T \to C$ give the required composition. In topos (and more generally in regular categories) this operation is associative and diagonals give identities: hence we have a 'category of relations' from the topos, on whose homsets we can define relational operations based on the structures in C. Following *op.cit.*, we write such a universe $Rel(C)$. Conversely, given such a category E of relations as above with a suitable set of operations such as join and meet as well as inverse, an arrow R can be considered to be a 'map' if it satisfies $R \circ R^\bullet \subset \mathbf{id}$ and $R^\bullet \circ R \supset \mathbf{id}$ where $R^\bullet$ is the operation corresponding to the relational inverse. Such 'maps' compose, so we obtain a category $Map(R)$. We can now state the following proposition.

Proposition 2. $Map(\mathsf{RPS}_{rel}) \simeq \mathsf{RPS}$ and $Rel(\mathsf{RPS}) \simeq \mathsf{RPS}_{rel}$, where $\simeq$ denotes categorical equivalence.

The verification of the second part is easy by forming the relational universe from RPS by the standard method outlined above, which in turn gives the first part by the general result in (Freyd and Scedrov 1990). Thus we can concentrate on RPS for our algebraic study of these two universes.

Finally, observe that PS is also a topos by the equivalence between PS and RPS. We

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

can then apply the same construction to PS, obtaining PS_{rel} in the end, so that we now know PS_{rel} gives the relational counterpart of PS.

A.2. *Concrete constructions in* RPS *and* PS

Here we show that the concrete constructions of products and quotients in Section 2 do give corresponding categorical notions in RPS.

Proposition 3.

(i) $\mathscr{P} \times \mathscr{Q}$ (is a process set and) is a product of $\mathscr{P}$ and $\mathscr{Q}$ in RPS.

(ii) Given a compatible relation $\mathscr{R}$ seen as a rooted p-set as in Definition 2.9 (ii), there is a monic pair $\mathscr{F} : \mathscr{R} \to \mathscr{P}$ and $\mathscr{G} : \mathscr{R} \to \mathscr{Q}$ in RPS such that $\mathscr{F}^{-1}\mathscr{G} = \mathscr{R}$. Conversely, any monic pair gives rise to a compatible relation in this way, up to isomorphism.

(iii) Suppose $\sim$ is a congruence on $\mathscr{P}$. Then if $F, G :\sim\!\!\to \mathscr{P}$ is a monic pair tabulating $\sim$, there is a unique epimorphism $H : \mathscr{P} \to \mathscr{P}/\sim$ for which F, G are projections in a pullback of H along itself.

Proof. (i) is easy. For (ii), noticing monics are injective homomorphisms in RPS, we know there is a monic from $\mathscr{R}$ to $\mathscr{P} \times \mathscr{Q}$. Using this, we construct a monic pair that does satisfy the properties. Finally, for (iii), without loss of generality we can take the canonical projection from $\mathscr{R}$ (via its inclusion into $\mathscr{P} \times \mathscr{P}$ as above) as the monic pair. Then H *qua* function is given as the natural map induced by $\mathscr{R}$. Thus $H : Q \mapsto [P]_\sim$ iff $Q \sim P$, but this means, by Proposition 2.8, that $\mathscr{FN}([P]_\sim)$ should be given as $\bigcap\{Q \mid Q \sim P\}$, and similarly for $[P]_\sim[\sigma]$. Thus if we show $\mathscr{P}/\sim$ is a rooted process set, we are done. Let $A = \bigcap\{\mathscr{FN}(P')|P' \in [P]_{\mathscr{R}}\}$. Suppose $\sigma_1 \restriction A = \sigma_2 \restriction A$. Then we get $P[\sigma_1] = P \prod_{1\leqslant i\leqslant n}[\binom{\sigma_2(x_i)\sigma_1(b_i)}{\sigma_1(b_i)\sigma_2(x_i)}] \circ [\sigma_2] \circ \prod_{1\leqslant i\leqslant n}[\binom{b_i x_i}{x_i b_i}]$ where $\{b_i\} = \mathscr{FN}(P)\setminus A$ and $\{x_i\}$ are taken fresh such that $\sigma_2(x_i) \notin \sigma_1(\mathscr{FN}(P))$. If, now, $\{b_i\}$ is *not* empty, there is a name b_1 that is *not* in $\mathscr{FN}(P')$ for some $P' \in [P]_{\mathscr{R}}$. By compatibility and because we have $P'[\binom{b_i x_i}{x_i b_i}] = P'$ for such P', we have

$$\begin{array}{rcl} P[\rho_1] \;\; \mathscr{R} \;\; P' \prod_{1\leqslant i\leqslant n}[\binom{\sigma_2(x_i)\sigma_1(b_i)}{\sigma_1(b_i)\sigma_2(x_i)}] \circ [\sigma_2] \circ \prod_{1\leqslant i\leqslant n}[\binom{b_i x_i}{x_i b_i}] \\ = \;\; P' \prod_{2\leqslant i\leqslant n}[\binom{\sigma_2(x_i)\sigma_1(b_i)}{\sigma_1(b_i)\sigma_2(x_i)}] \circ [\sigma_2] \circ \prod_{2\leqslant i\leqslant n}[\binom{b_i x_i}{x_i b_i}]. \end{array}$$

Repeating the procedure for each b_i and using transitivity of $\mathscr{R}$, we finally get that $P[\sigma_1]\mathscr{R}P[\sigma_2]$, that is, $[P]_{\mathscr{R}}[\sigma_1] = [P]_{\mathscr{R}}[\sigma_2]$, as a consequence of $\sigma_1 \restriction A = \sigma_2 \restriction A$. Therefore, if for any $a \in \bigcap\{\mathscr{FN}(P')|P' \in [P]_{\mathscr{R}}\}$, $\sigma(a) = a$ holds, then the above discussions show $[P]_{\mathscr{R}}[\sigma] = [P]_{\mathscr{R}}[id_{\mathscr{N}}] = [P]_{\mathscr{R}}$. Regarding the third equation, it holds since

$$\mathscr{FN}([P]_{\mathscr{R}}[\sigma]) = \bigcap\{\sigma(\mathscr{FN}(P'))|P' \in [P]_{\mathscr{R}}\} = \sigma(\mathscr{FN}([P]_{\mathscr{R}})),$$

and hence we are done. □

The above result easily leads to the observation that the concrete constructions on (non-rooted) p-sets given in Definition 4.12 give their categorical counterparts in PS. By Theorem 5.1, it is enough to show that translations result in isomorphic structures. We only sketch one example (for the case of products) of the translation in the following. Given $\mathscr{P}$ and $\mathscr{Q}$, $[\![\mathscr{P}]\!] \times [\![\mathscr{Q}]\!]$ includes all possible 'connections' between name-based processes,

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

while rooted processes in $[\![\mathscr{P} \times \mathscr{Q}]\!]$ are precisely all possible connections among name-less processes whose handles are labelled by names. Since the fact that they are isomorphic to each other is evident, we are done.

Appendix B. Proofs

B.1. *Proof of Proposition 3.7*

The 'only if' direction is easy. We will prove the 'if' direction. We first suppose X' and Y' are transitive G-sets, and for some $x' \in X'$ and $y' \in Y'$, we have $G_x = g \cdot G_y \cdot g^{-1}$. It is standard that if $y'' = gy' \in Y'$, then $G_{y''} = g \cdot G_y \cdot g^{-1} = G_x$, so that, for any g, we have $G_{gx'} = G_{gy''}$ by Definition 3.5 (iii). Now construct a function F such that $F(gx') = g(Fx')$ and $F(x') = y''$ for any g. This is well-defined, because if $g_1x' = g_2x'$, then $g_1x' = (g_2g_1^{-1}g_1)x'$, so $g_2g_1^{-1} \in G_{g_1x'} = G_{g_1y''}$, and therefore $F(g_1x') = g_1y'' = (g_2g_1^{-1}g_1)(y'') = g_2y''$. So F defines a map, and indeed a morphism. But if $F(g_1x') = F(g_2x')$, then $g_2g_1^{-1} \in G_{g_1y''} = G_{g_1x'}$, so $g_1x' = (g_2g_1^{-1}g_1)(x') = g_2x'$, and therefore F is bijective, and hence an isomorphism. Returning to the Proposition above, since each orbit is transitive, the bijective correspondence in the statement of the above proposition induces isomorphisms between respective orbits, which together form an isomorphism as a whole, and hence we are done.

B.2. *Proof of Proposition 5.9*

For (i), let $\mathscr{F}$ be a homomorphism. It is obvious that $\mathscr{F}^\circ$ sends one process to a unique process, and that it is defined everywhere on the domain. We show each component of $\mathscr{F}^\circ$ is symmetry preserving. In doing so, the surjectivity in each correspondence as well as the uniqueness of correspondences modulo $\sim$ will also be established. By definition and also by Proposition 2.8, if $F^\circ : \underline{sym(P)} \mapsto_f \underline{sym(Q)}$ (here $\underline{sym(P)}$ denotes a translated non-rooted process making the underlying symmetries explicit, and similarly in the following), we can write, for fixed Q and $\pi : \mathscr{FN}(Q) \to \mathscr{N}$ such that $F(P) = Q[\pi]$, $f = \tau \circ \pi^{-1} \circ \rho$ where $\rho \in sym(P)$ and $\tau \in sym(Q)$. We now show that we do not need to mention ρ above, and can write $f = \tau \circ \pi^{-1}$ instead. Because of Proposition 2.8 again, we have $\pi = \rho \circ \pi \circ (\pi^{-1} \circ \rho^{-1} \circ \pi)$. But $F(P[\rho^{-1}]) = F(P)$, and hence $Q[\pi][\rho^\bullet] = Q[\pi]$, therefore $Q[\rho^{-1} \circ \pi] = Q[\pi][\pi^{-1} \circ \rho^{-1}] = Q$, so $\rho^{-1} \circ \pi$ is a symmetry of Q. Let the inverse of this symmetry (which is by definition also a symmetry) be τ'. Now using the above equation, we have $\tau \circ \pi^{-1} \circ \rho = \tau \circ \tau' \circ \pi^{-1}$, and hence we have the result, as required. This also shows uniqueness modulo $\sim$, and surjectivity is now obvious, and hence we are done.

For (ii), suppose $\mathfrak{F}$ is a p-map. Let $\mathfrak{F} : p \mapsto_{f'} q$. Then by symmetry preservation of f', we can write $f' = \rho_i \circ f$ for a fixed f, with $\rho_i \in \mathscr{S}(q)$. We now show $\mathfrak{F}^\star$ does define a function. Since we already know it is compatible, this shows $\mathfrak{F}^\star$ is a homomorphism. Regarding $\mathfrak{F}^\star$ as a map on equivalence classes, for each injection $\Psi_0 : \mathscr{H}(p) \to \mathscr{N}$ we have its image by $\mathfrak{F}^\star$, $\{\langle \Psi_i, q\rangle\}$, such that $\mathfrak{F}^\star : \langle \Psi_0, p\rangle \mapsto \langle \Psi_i, q\rangle$ and $\Psi_i^{-1} \circ \Psi_0 = \rho_i \circ f$. Using surjectivity, we can then write $\Psi_i = \Psi_0 \circ f^{-1} \circ \rho_i^{-1}$, which shows $\langle \Psi_i, q\rangle \approx \langle \Psi_j, q\rangle$. Moreover, if, for $\tau \in \mathscr{S}(p)$ we have $\mathfrak{F}^\star : \langle \Psi_0 \circ \tau, p\rangle \mapsto \langle \Psi, q\rangle$, then by symmetry preservation we can permute τ to the symmetry of q, and hence we are done.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

Finally, for (iii), observe that, from (i) above, whenever $F^\circ : \underline{sym(P)} \mapsto_f \underline{sym(Q)}$ we have $F : P \mapsto Q[f^\bullet]$. First $F_2^\circ \circ F_1^\circ \subset (F_2 \circ F_1)^\circ$ is easy because, if $F_1^\circ : \underline{sym(P)} \mapsto_f \underline{sym(Q)}$ $F_2^\circ : \underline{sym(Q)} \mapsto_g \underline{sym(R)}$ and $F_2^\circ \circ F_1^\circ : \underline{sym(P)} \mapsto_{g\circ f} \underline{sym(R)}$, then $F_1 : P \mapsto Q[f^\bullet]$ and $F_2 : Q \mapsto R[g^\bullet]$, which implies $F_2 \circ F_1 : P \mapsto R[g^\bullet][f^\bullet]$, and therefore $F_2 \circ F_1 : P \mapsto R[g^\bullet][f^\bullet]$, that is, $(F_2 \circ F_1)^\circ : \underline{sym(P)} \mapsto_{g\circ f} \underline{sym(Q)}$, as required. To show that $F_2^\circ \circ F_1^\circ \supset (F_2 \circ F_1)^\circ$, if $(F_2 \circ F_1)^\circ : \underline{sym(P)} \mapsto_h \underline{sym(Q)}$, then $F_2 \circ F_1 : P \mapsto R[h^{-1}]$, and hence, for some Q', $F_1 : P \mapsto Q'(= Q'[f][f^{-1}])$, which means $F_1^\circ : \underline{sym(P)} \mapsto_f \underline{sym(Q'[f])}$, as well as $F_2 : Q' \mapsto R[h^{-1}]$, that is $F_2 : Q'[f] \mapsto R[h^{-1}][f]$, which means $F_2^\circ : sym(Q'[f]) \mapsto_{h\circ f^{-1}} sym(R)$, but because of Proposition 2.8 we know that $h = g \circ f^{-1} \circ f = g$, as required. This shows $(\,\cdot\,)^\circ$ commutes with the functional composition. To show the same for $(\,\cdot\,)^\star$, the direction $\mathfrak{F}_2^\star \circ \mathfrak{F}_1^\star \supset (\mathfrak{F}_2 \circ \mathfrak{F}_1)^\star$ is easy by using the presentation we gave in (ii) above. We show the converse. Suppose $\mathfrak{F}_1^\star : \langle \Psi_1, p\rangle_\approx \mapsto \langle \Psi'_2, q\rangle_\approx$, $\mathfrak{F}_2^\star : \langle \Psi'_2, q\rangle_\approx \mapsto \langle \Psi'_3, r\rangle_\approx$, and therefore $\mathfrak{F}_2^\star \circ \mathfrak{F}_1^\star : \langle \Psi_1, p\rangle_\approx \mapsto \langle \Psi'_3, r\rangle_\approx$. Then for some $\Psi_2 \approx \Psi'_2$ and $\Psi'_3 \approx \Psi_3$, we have $F_1 : p \mapsto_{\Psi_2^{-1}\circ\Psi_1} q$ and $F_2 : q \mapsto_{\Psi_3^{-1}\circ\Psi_2\circ\rho} q$ where $\rho \in sym(q)$. By symmetry preservation, we can write $F_2 : q \mapsto_{\rho'\circ\Psi_3^{-1}\circ\Psi_2} q$, and hence by surjectivity, we know $F_2 \circ F_1 : p \mapsto_{\rho'\circ\Psi_3^{-1}\circ\Psi_1} r$, as required.

B.3. *Proof of Lemma 6.6*

(i) The proof is immediate.

(ii) For the associativity, define $|'(P, Q, R)$ as

$$P\{\tilde{a}'/\tilde{a}\} \circ Q\{\tilde{b}'/\tilde{b}\} \circ R\{\tilde{c}'/\tilde{c}\} \circ \prod 4(a'_i b'_i c'_i e_i) \circ \prod 3(a'_j b'_j f_j) \circ \prod 3(a'_k c'_k g_k) \circ \prod 3(b'_l c'_l h_l)$$

where $4(abcd) = 3(abe) \circ 3(ecd)$ with e fresh, and $\tilde{a} \ldots \tilde{h}$ given by
$\{\tilde{a}\} = \mathscr{FN}(P) \cap (\mathscr{FN}(Q) \cup \mathscr{FN}(R))$,
$\{\tilde{b}\} = \mathscr{FN}(Q) \cap (\mathscr{FN}(P) \cup \mathscr{FN}(R))$,
$\{\tilde{c}\} = \mathscr{FN}(R) \cap (\mathscr{FN}(P) \cup \mathscr{FN}(P))$,
$\{\tilde{e}\} = \mathscr{FN}(P) \cap \mathscr{FN}(Q) \cap \mathscr{FN}(R)$,
$\{\tilde{f}\} = (\mathscr{FN}(P) \cap \mathscr{FN}(Q)) \backslash \mathscr{FN}(R)$,
$\{\tilde{g}\} = (\mathscr{FN}(P) \cap \mathscr{FN}(R)) \backslash \mathscr{FN}(Q)$,
$\{\tilde{h}\} = (\mathscr{FN}(Q) \cap \mathscr{FN}(R)) \backslash \mathscr{FN}(P)$,
while $a'_i, b'_i, c'_i, \ldots$ etc. are selected according to the above correspondence, for example, $a'_i = e_i\{\tilde{a}/\tilde{a}'\}$, and so on. Notice that any three pairs of these agents have no common names, so we can dispense with parentheses. It is routine to show that both sides of the given equation are equal to $|'(P, Q, R)$.

(iii) We show a simple case where $\mathscr{FN}(P) = \mathscr{FN}(Q) = \{a\}$. The general case is the same.

$$\begin{aligned} 1(a) \circ (P|'Q) &\stackrel{\text{def}}{=} 1(a) \circ P\{b/a\} \circ Q\{c/a\} \circ 3(abc) \\ &= 2(bc) \circ P\{b/a\} \circ Q\{c/a\} \\ &= P\{c/a\} \circ Q\{c/a\} \quad \equiv_\alpha \quad P \circ Q. \end{aligned}$$

(iv) The proof is similar to (iii).

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press

(v) $2(ab)|'2(ab) \stackrel{\text{def}}{=} 2(a'b') \circ 2(a''b'') \circ 3(aa'a'') \circ 3(bb'b'') = 3(ab'b'') \circ 3(b'b''b) = 2(ab)$.

(vi) $2(ab)|'2(bc) \stackrel{\text{def}}{=} 2(ae) \circ 3(ebf) \circ 2(fc) = 3(abf) \circ 2(fc) = 3(abc)$.

(vii) If $a, b \notin \mathscr{FN}(P)$, the result is trivial. If both are in $\mathscr{FN}(P)$,

$$\begin{aligned} 2(ab)|'P &\stackrel{\text{def}}{=} 2(a'b') \circ P\{a''b''/ab\} \circ 3(a'aa'') \circ 3(b'bb'') \\ &= P\{a''b''/ab\} \circ 3(b'aa'') \circ 3(b'bb'') \\ &= P\{a''b''/ab\} \circ 3(b'a''b'') \circ 3(b'ab) \\ &= P[\binom{ab}{ba}] \circ 3(a'aa'') \circ 3(b'bb'') \quad = \quad 2(ab)|'P[\binom{ab}{ba}]. \end{aligned}$$

The case where only either of a, b is in $\mathscr{FN}(P)$ is simpler.

(viii) If $a \in \mathscr{FN}(P)$, $1(a)|'P = 1(a') \circ P\{a''/a\} \circ 3(a'aa'') = P\{a''/a\} \circ 2(aa'') = P$, as required. Note that if $a \notin \mathscr{FN}(P)$, we cannot get rid of $2(aa'')$.

Acknowledgements

The author would like to thank Samson Abramsky, Peter Aczel, Vincent Danos, Philippa Gardner, Cliff Jones, Alex Mifsud, Robin Milner, Andrew Pitts and Nobuko Yoshida for beneficial discussions, and N. Raja for his hospitality in Bombay. He is also grateful to an anonymous referee and Nobuko Yoshida for criticisms of earlier versions of the paper.

References

Abramsky, S. (1996) Retracing Some Paths in Process Algebras. CONCUR'96. *Springer-Verlag Lecture Notes in Computer Science* **1119** 1–17.

Abramsky, S., Gay, S. and Nagarajan, R. (1995) Interaction Categories and Foundations of Typed Concurrent Computing. *Deductive Program Design: Proceedings of the 1994 Marktoberdorf International Summer School*, Springer-Verlag.

Abramsky, S., Jagadeesan, R. and Malacaria, P. (1994) Full Abstraction for PCF. To appear in *Infomation and Computation.*

Asperti, A., Danos, V., Laneve, V. and Regnier, L. (1994) Paths in the lambda-calculus. In: *LICS'94*, IEEE Computer Society Press, 426–436.

Asperti, A. and Longo, L. (1991) *Categories, Types and Categories, Types and Structures: an introduction to Category Theory for the working computer scientist,* M.I.T. Press.

Aczel, P. (1990) Replacement Systems and the axiomatisation of Situation Theory. In: Situation Theory and Applications. *CSLI Lecture Notes* **22** Stanford University.

Aczel, P. (1991) Term Declaration Logic and Generalised Composita. In: *LICS '91*, IEEE Computer Society Press.

Bawden, A. (1986) *Connection Graphs, Lisp and Function*, ACM Press 258–265.

Baeton, J. and Wejland, W. (1990) *Process Algebra*, Cambridge Tracts in Theoretical Computer Science, CUP.

Berry, G. and Boudol, G. (1990) The Chemical Abstract Machine. *Theoretical Computer Science* **96** 217–248.

Boreale, M. and Sangiorgi, D. (1996) Some Congruence Properties for π-calculus Bisimilarities, Revised version of Tec. Rep. RR-2870, INRIA – Sophia Antipolis.

Danos, V. and Regnier, L. (1993) Local and asynchronous beta-reduction (an analysis of Girard's execution formula). In: *LICS'93*, IEEE Computer Society Press.

Danos, V. and Regnier, L. (1996) Reversible and Irreversible Computations. *ENCS* **3** Elsevier.

Freyd, P. and Scedrov, A. (1990) *Categories, Allegories*, North-Holland.

Gardner, P. (1995) A name free account of action calculi. In: Proc. MFPS'95. *ENCS 1*, Elsevier.

Girard, J.-Y. (1987) Linear Logic, *Theoretical Computer Science* **50** 1–102.

Girard, J.-Y. (1988) Geometry of Interaction I: interpretation of System F. In: *Logic Colloquium '88*, North-Holland.

Girard, J.-Y. (1989) Geometry of Interaction II: deadlock free algorithms. In: Proceedings of COLOG-88. *Springer-Verlag Lecture Notes in Computer Science* **417**.

Girard, J.-Y. (1995b) Geometry of Interaction III: Accomodating the Additives. In: Girard, J.-Y., Lafont, Y. and Regnier, L. (eds.) Advances in linear logic. *London Mathematical Society Lecture Notes Series*, Cambridge University Press **222** 329–389.

Girard, J.-Y. (1996) Proof Nets: Parallel Syntax for Proof-Theory. Typescript, 28pp.

Girard, J.-Y., Lafont, Y. and Regnier, L. (eds.) (1995) Advances in linear logic. *London Mathematical Society Lecture Notes Series*, Cambridge University Press **222**.

Gonthier, G., Abadi, M. and Levy, J.-J. (1992) The Geometry of Optimal Lambda Reduction. In: *POPL'92*, ACM press 15–26.

Hennesy, M. (1989) *Algebraic Theory of Processes*, MIT Press.

Hoare, C. A. R. (1985) *Communicating Sequential Processes*, Prentice Hall.

Honda, K. (1995a) Notes on P-Algebra (1): Process Structure. Proc. TPPP'94. *Springer-Verlag Lecture Notes in Computer Science* **907**. Also as UMCS-95-12-2, Department of Computer Science, University of Manchester.

Honda, K. (1995b) Notes on P-Algebra (2): Group Presentation of Process Structure. UMCS-95-12-3, Department of Computer Science, University of Manchester.

Honda, K. (1996) Composing Processes. *POPL'96*, ACM Press 344–357.

Honda, K. and Yoshida, N. (1993) On Reduction-Based Process Semantics. *Theoretical Computer Science* **151** 437–486.

Honda, K. and Yoshida, N. (1994a) Combinatory Representation of Mobile Processes. In: *POPL'94*, ACM press 348–360.

Honda, K. and Yoshida, N. (1994b) Replication in Concurrent Combinators, In: TACS'94. *Springer-Verlag Lecture Notes in Computer Science* **789** 786–805.

Honda, K. and Yoshida, N. (1997) Game-theoretic Analysis of Call-by-value Computation. In: Proc. of ICALP'97. *Springer-Verlag Lecture Notes in Computer Science* **1256** 225–236. (Full version to appear in *Theoretical Computer Science*.)

Hyland, M. and Ong, L. (1994) On Full Abstraction for PCF": I, II and III. 130 pages. ftp-able at `theory.doc.ic.ac.uk/papers/Ong`. (To appear in *Information and Computation*.)

Jech (1977) About the axiom of choice. In: *Handbook of Mathematical Logic,* North-Holland 345–370.

Kunen, K. (1980) *Set Theory: An Introduction to Independence Proofs,* North-Holland.

Kahn, G. and MacQueen, D. (1977) Coroutines and networks of parallel processes. *Information Processing* **77** 993-998.

Lafont, Y. (1990) Interaction Nets. In: *POPL'90*, ACM press 95–108.

Lafont, Y. (1995) From Interaction Nets to Proof Nets. In: Girard, J.-Y., Lafont, Y. and Regnier, L. (eds.) Advances in linear logic. *London Mathematical Society Lecture Notes Series*, Cambridge University Press **222** 225–248.

Lafont, Y. (1996) Interaction Combinators. *Infomation and Computation* **137** 69–101.

Lambek, J. and Scott, P. J (1986) *Introduction to higher order categorical logic*, Cambridge University Press.
Lamping, J. (1990) An Algorithm for Optimal Lambda Calculus Reduction. In: *POPL'90*, ACM Press 16–30.
Lang, S. (1993) *Algebra*, third edition, Addison Wesley.
Mac Lane, S. and Moerdijk, I. (1992) *Sheaves in Geometry and Logic*, Springer-Verlag.
McLarty, C. (1992) *Elementary Categories, Elementary Toposes*, Oxford University Press.
Meseguer, J. (1992) Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science* **92**.
Mifsud, A. (1996) *Control Structures*, Ph. D. Thesis, University of Edinburgh.
Mifsud, A., Milner, R. and Power, J. (1996) Control Structures. In: *LICS 95*, IEEE Computer Society.
Milner, R. (1979) Flowgraphs and Flow Algebras. *Journal of Computing Machinery* 794–818.
Milner, R. (1980) A Calculus of Communicating Systems. *Springer-Verlag Lecture Notes in Computer Science* **76**.
Milner, R. (1989) *Communication and Concurrency*, Prentice Hall.
Milner, R. (1990) Functions as Processes. *Mathematical Structures in Computer Science* **2** (2) 119–146.
Milner, R. (1992) Polyadic π-Calculus: a tutorial. *Logic and Algebra of Specification* 203–246.
Milner, R. (1993) Action Structures. Research Report LFCS-92-249, Computer Science Department, Edinburgh University.
Milner, R. (1994) Pi-nets: a graphical form of pi-calculus. In: Proc. of ESOP'94, Programming. *Springer-Verlag Lecture Notes in Computer Science* **788** 26–42.
Milner, R. (1995a) *Calculi for Interaction*. Acta Informatica **33** (8) 707–737.
Milner, R. (1995b) Control Structures II: Naming Monoids (draft).
Milner, R., Parrow, J. G. and Walker, D. J. (1989) A Calculus of Mobile Processes. *Information and Computation* **100** (1) 1–77.
Milner, R. and Sangiorgi, D. (1992) Barbed Bisimulation. In: Proc. of ICALP'92. *Springer-Verlag Lecture Notes in Computer Science* **623** 685–695.
Nerode, A. (1959) Composita, Equations, and Freely Generated Algebras. *Transactions of the American Mathematical Society* 139–151.
Palamidessi, C. (1997) Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. In: *POPL'97*, ACM press 256–265.
Parrow, Y. (1993) Interaction Diagrams. In: Decades of Concurrency. *Springer-Verlag Lecture Notes in Computer Science* **803** 477–508.
Pierce, B. C. and Sangiorgi. D. (1993) Typing and subtyping for mobile processes. *LICS'93* 187–215. (The full version appeared in *Mathematical Structures in Computer Science* (1996) **6** (5) 409–454.)
Pitts, A. and Gabbay, D. (1999) A New Approach to Abstract Syntax Involving Binding. *LICS'99*, IEEE.
Raja, N. and Shyamasundar, R. K. (1994) Combinatory Formulations of Concurrent Languages. *TOPLAS*, ACM Press **19** (6) 899–915.
Sangiorgi, D. (1992) *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*, Ph. D. Thesis, Department of Computer Science, University of Edinburgh.
Stark, I. (1995) *Names and Higher-Order Functions*, Ph. D. Thesis, Department of Computer Science, Cambridge University.
Walker, D. (1994) Objects in the π-calculus. *Information and Computation* **116** (2) 253–271.
Winskel, G. (1984) Synchronization Trees. *Theoretical Computer Science* **34** 33–82.
Winskel, G. and Nielsen, M. (1995) Models for Concurrency. In: *Handbook of Logic in Computer Science* **4** Oxford University Press 1–148.

Yoshida, N. (1994) Graph Notation for Concurrent Combinators. In: TPPP'94. *Springer-Verlag Lecture Notes in Computer Science* **907** 393–412.

Yoshida, N. (1998) Minimality and Separation Results on Asynchronous Mobile Processes: Representability Theorems by Concurrent Combinators. Proc. CONCUR'98. *Springer-Verlag Lecture Notes in Computer Science* **1466** 131–146.

https://doi.org/10.1017/S0960129599002947 Published online by Cambridge University Press