A Congruence Rule Format for Name-Passing Process Calculi from Mathematical Structural Operational Semantics

(Extended Abstract)

Marcelo Fiore* and Sam Staton†
Computer Laboratory, University of Cambridge

Abstract

We introduce a mathematical structural operational semantics that yields a congruence result for bisimilarity and is suitable for investigating rule formats for name-passing systems. Indeed, we instantiate this general abstract model theory in a framework of nominal sets and extract from it a GSOS-like rule format for name-passing process calculi for which the associated notion of behavioural equivalence — given by a form of open bisimilarity — is a congruence.

Introduction

A significant strand of research in semantics concerns defining and establishing properties of formats for operational rules [2]. By moving away from a particular syntax and semantics one can simultaneously study a whole class of calculi, becoming instead concerned with the intrinsic nature of the kinds of system that the formats allow.

In this vein, the present work provides an analysis of the congruence properties of bisimilarity for name-passing systems, with the π-calculus [13] being the paradigmatic example. Specifically we ask: What is a name-passing process calculus, and when is its behavioural equivalence a congruence? As we proceed to explain, we tackle these questions from a model-theoretic perspective, merging a series of strands in semantics research.

Background. A starting point for models of the π-calculus is the work of Fiore, Moggi and Sangiorgi [6] and of Stark [18], where a domain equation is solved in a functor category $\mathcal{D}^{\mathbb{I}}$, for a category of domains $\mathcal{D}$. The crucial ingredient of this semantic universe is that it is parameterised by finite sets of names (intuitively those available to a process at any one time) subject to the mode of variation given by injective renamings, so that the model embodies the well-known invariance property of bisimilarity under injective renaming.

An abstract treatment of the GSOS rule format and of bisimilarity congruences was later provided by the mathematical operational semantics of Turi and Plotkin [20]. Given a model $(\mathcal{M}, \Sigma, B)$ as follows

(1)

where $\mathcal{M}$ is a category with binary products, Σ is a signature endofunctor with free term monad T and B is a behaviour endofunctor, they introduced natural transformations of the form

$$\Sigma\big((-) \times B(-)\big) \Longrightarrow BT(-) : \mathcal{M} \to \mathcal{M} \qquad (2)$$

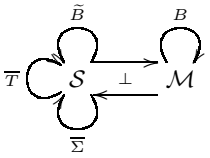
as an abstract notion of operational rule, explaining how such rules induce liftings of the monad T to the category of B-coalgebras by structural recursion and giving conditions under which this yields a compositional semantics. In particular, when the monad T is freely generated by an algebraic signature Σ on the universe of sets, they established that a family of rules in the GSOS format [4] for a first-order process calculus corresponds to a natural transformation as in (2) where $B(-) = \mathcal{P}_{\mathrm{f}}(-)^A$, for $\mathcal{P}_{\mathrm{f}}$ the covariant finite powerset functor and A a set of actions.

The first step needed to put name-passing systems within this framework was to give an algebraic treatment of binding operators. This, amongst other things, was achieved by Fiore, Plotkin and Turi [7] by shifting from the universe of sets to the universe $\mathbf{Set}^{\mathbb{F}}$ of sets parameterised by finite sets

*Research partially supported by an EPSRC Advanced Research Fellowship.
†Research partially supported by EPSRC grant GR/T22049/01.

of variables (intuitively those that may be free in a term) subject to the mode of variation given by arbitrary variable renamings, so that the model embodies the well-known properties of substitution.

Thus in this model-theoretic study of name-passing process calculi two different, but closely related, natural semantic universes arise: a category $\mathcal{M} = \mathbf{Set}^{\mathbb{I}}$ for behaviour, that supports a behaviour endofunctor B, and another category $\mathcal{S} = \mathbf{Set}^{\mathbb{F}}$ for syntax, that supports algebraic signatures Σ for binding operators inducing free term monads T on $\mathcal{S}$. The aforementioned framework of mathematical operational semantics is thus not directly applicable. Importantly, however, as Fiore and Turi [9] realised, the two universes are related by an adjunction, leading to the following situation



(3)

where $\widetilde{B}$ is obtained by shifting B from $\mathcal{M}$ to $\mathcal{S}$ by pre and post composition with the left and right adjoints respectively. Intuitively, $\widetilde{B}$ is a version of B for behaviour embodying the extra structure imposed by the category $\mathcal{S}$, which in the case under consideration amounts to closure under arbitrary renamings. It follows that abstract rules for the model $(\mathcal{S}, \overline{\Sigma}, \widetilde{B})$, given by natural transformations

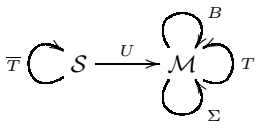
$$\overline{\Sigma}((-) \times \widetilde{B}(-)) \Longrightarrow \widetilde{B}\overline{T}(-) : \mathcal{S} \to \mathcal{S} \ , \qquad (4)$$

induce a semantics for which $\widetilde{B}$-bisimulation is a congruence.

The notion of $\widetilde{B}$-bisimulation that arises amounts to what we call *wide-open* bisimulation; *viz.*, the version of open bisimulation that does not take account of distinctions [16, Sec. 3], and so is less discriminating than open bisimilarity [16, Sec. 7]. Wide-open bisimilarity has been considered by various authors, under different names; *e.g.* [14, 5, 9, 8].

Contributions. In the present work, we continue this line of investigation.

In aiming to extract a concrete rule format from the abstract rules (4) one faces the problem of devising syntax for the shifted-behaviour endofunctor—a difficult task due to the nature of the right adjoint. Rather than following this direction here, our first step is to instead develop a model theory that lies in between that of (1–2) and (3–4). Indeed, we consider models of the following kind



(5)

where $\overline{T}$ is a lifting of T, and develop a mathematical operational semantics for abstract rules of the form

$$\Sigma\big(U(-) \times BU(-)\big) \Longrightarrow BTU(-) : \mathcal{S} \to \mathcal{M} \qquad (6)$$

The way to think about this framework is as a model of syntax and behaviour in which the syntax, and consequently the rules, carry extra structure. In the context of this paper, the extra structure amounts to an operation of name substitution which is essential for modelling name communication.

The framework (5–6) clearly extends (1–2). Moreover, (5) fits within (3) whenever $\overline{\Sigma}$ and $\overline{T}$ restrict from $\mathcal{S}$ to $\mathcal{M}$ and, in this case, if U preserves binary products, the abstract rules (6) can actually be regarded as a subclass of the abstract rules (4), all of which happens in our example.

Our next step is to move away from semantic universes based on functor (more specifically, presheaf) categories (in which stages are explicitly indexed) and instead work in more convenient (sheaf) subcategories. Indeed, we take (5) with $\mathcal{M}$ the universe $\mathbf{Nom}$ of nominal sets and $\mathcal{S}$ a universe $\mathbf{NomSub}$ of nominal substitutions over it. Nominal sets have been championed by Gabbay, Pitts and others (*e.g.* [10]) as a convenient setting in which to handle various aspects of syntax with variable binding. Nominal substitutions are a novel contribution of this paper; they provide a natural extension of nominal sets, supporting the operation of name substitution. Thus, we explore the abstract GSOS rule format (6) from a practical, concrete point of view for the case $\mathcal{S} = \mathbf{NomSub}$ and $\mathcal{M} = \mathbf{Nom}$. In this context, signature endofunctors will arise from algebraic binding signatures. As for behaviour, a key observation is that for relations that are closed under name substitutions, the usual notions of early, late, and ground (as in [17]) bisimulation coincide. Hence we are able to work with ground bisimulation, which admits a reasonably simple behavioural model.

In order to give a concrete representation to abstract rules of the form (6), we introduce a syntactic notion of rule structure. This can be seen as the same concrete notion of rule that has been used, though in a slightly informal fashion, to define systems such as the π-calculus (*e.g.* as in [13]). Subsequently, we provide conditions on rule structures leading to a GSOS-like format that gives rise to abstract rules. For name-passing process calculi defined using rule structures satisfying our conditions, wide-open bisimilarity is a congruence.

Related work. Our work is novel in that we extract a concrete rule format from a model theory for name-passing. However, several authors have tackled congruence formats for name passing from an operational point of view. Weber and Bloom [21] introduced a rule format for name passing where they take a restriction operator and a structural congruence as primitive. Bernstein [3] has encoded the π-calculus rules within her framework, but the bisimilarity there is unusual in that it does not contain α-equivalence. More

recently, and relevant, Ziegler, Miller and Palamidessi [22] have reformulated the tyft/tyxt format within a formal system (fold-nabla [12]) with a special quantifier for new names, introduced a notion of congruence in that setting, and established a congruence result for open bisimilarity.

Future research. In this paper we give an interpretation of rule structures in terms of abstract rules in the model, but it might be worthwhile to interpret rule structures in a formal system, such as nominal logic [15] and/or the logic fold-nabla [12].

Concerning the possibilities on rule formats that this work opens up, it would be interesting to investigate extensions to our concrete format that are complete with respect to the model theory (that is, where every abstract rule arises from a family of concrete ones), and to further provide concrete formats for abstract rules of the form (4).

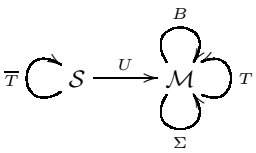
An immediate problem for achieving completeness is that the concrete format that we present here involves only positive premises, whereas Turi and Plotkin [20] have shown that abstract rules (2) over $\mathbf{Set}$ account for rules with both positive and negative premises. We have found that by restricting attention to natural transformations that are suitably monotone one is able to capture the positive nature of the rules. Even for this restricted model, it seems that, for a completeness result, our concrete format needs to be further extended to allow rules equipped with suitable freshness assumptions on names.

It would also be interesting to adapt the model to account for open bisimulation and/or to see how the framework of Klin [11], which accounts for equivalences other than bisimilarity, extends to the name-passing case. More speculatively, we would like to extend the model theory, and the rule format, to be relevant for data-passing systems such as the applied pi calculus [1].

1. Mathematical operational semantics

We develop a mathematical operational semantics for rules that carry extra structure, establishing the congruence of behavioural equivalence.

Mathematical universe. We consider the following universe of discourse



of categories and functors between them, where T is the free monad on the endofunctor Σ and where $\overline{T}$ is a monad lifting of T along U. Recall that a *monad lifting* of a monad S on $\mathcal{C}$ along a functor $F : \mathcal{D} \longrightarrow \mathcal{C}$ is a monad $\overline{S}$ on $\mathcal{D}$ together with a natural isomorphism $\lambda : SF \xrightarrow{\sim} F\overline{S}$ that defines a monad functor $(F, \lambda) : (\mathcal{D}, \overline{S}) \longrightarrow (\mathcal{C}, S)$ in the sense of Street [19]. When $SF = F\overline{S}$ and $\lambda = \mathrm{id}$, the monad lifting is said to be *strict*.

Operational models and bisimulation. A (U, BU)-*dialgebra* (occasionally referred to as a U-*structured* B-*coalgebra*) is an object X of $\mathcal{S}$ equipped with a B-coalgebra structure $UX \to BUX$ on UX in $\mathcal{M}$. These are operational models to be thought as abstract transition systems carrying extra structure. We let (U, BU)-$\mathbf{dialg}$ be the category of (U, BU)-dialgebras and homomorphisms (*viz.*, maps between the underlying objects that are compatible with the coalgebra structure).

A U-*structured* B-*bisimulation (relation)* between two (U, BU)-dialgebras, (X, h) and (Y, k), is a (jointly mono) span $X \leftarrow R \rightarrow Y$ in $\mathcal{S}$ such that there is a B-coalgebra structure $UR \to BUR$ on UR lifting the span to the category of (U, BU)-dialgebras.

We say that a U-structured B-bisimulation between two (U, BU)-dialgebras is *final* if every other U-structured B-bisimulation between them factors through it uniquely. Whenever it exists, the final U-structured B-bisimulation is called U-*structured* B-*bisimilarity*.

Congruence of bisimilarity. For a monad S, to be thought of as describing algebraic structure, an S-*congruence* between two S-algebras (X, x) and (Y, y) is a span $X \leftarrow R \rightarrow Y$ for which there exists an S-algebra structure lifting the span to the category of S-algebras.

The following result describes a setting in which structured bisimilarity is a congruence.

Theorem 1.1 *Let $\overline{S}$ be a monad lifting of a monad S on $\mathcal{S}$ along the forgetful functor $F : (U, BU)$-$\mathbf{dialg} \longrightarrow \mathcal{S}$. Consider (U, BU)-dialgebras (X, h) and (Y, k), and let $SX \leftarrow R \rightarrow SY$ be a span in $\mathcal{S}$.*

If the span $F\overline{S}(X, h) \xleftarrow{\sim} SX \leftarrow R \rightarrow SY \xrightarrow{\sim} F\overline{S}(Y, k)$ is a U-structured B-bisimilarity between $\overline{S}(X, h)$ and $\overline{S}(Y, k)$ then the span $SX \leftarrow R \rightarrow SY$ is an S-congruence, where SX and SY are considered with their respective free algebra structures.

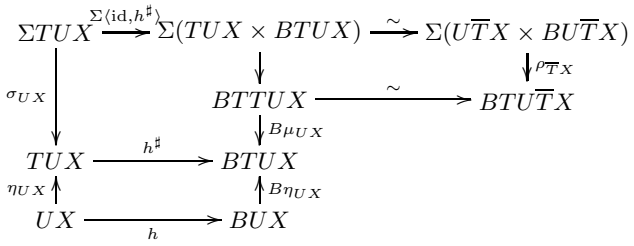
Abstract rules and operational semantics. We assume that $\mathcal{M}$ has binary products, and define an *abstract operational rule* as a natural transformation as follows:

$$\rho : \Sigma\big(U(-) \times BU(-)\big) \Longrightarrow BTU(-) : \mathcal{S} \to \mathcal{M} \quad (7)$$

Note that for $\mathcal{S} = \mathcal{M}$, $U = \mathrm{Id}$, and $T = \overline{T}$ this notion, and the following result, specialise to those of Turi and Plotkin [20].

Theorem 1.2 *For every B-coalgebra $h : UX \to BUX$ there exists a unique B-coalgebra $h^\sharp : TUX \to BTUX$*

such that the diagram



commutes, where σ denotes the free Σ-algebra structure, and η and μ respetively denote the unit and multiplication of the monad T. Further, the mapping defined by

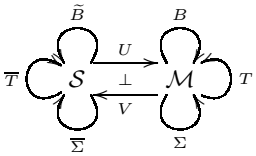
$$\overline{T}^{\rho}(X,h) = (\overline{T}X, U\overline{T}X \xrightarrow{\sim} TUX \xrightarrow{h^\sharp} BTUX \xrightarrow{\sim} BU\overline{T}X)$$

*extends to a strict monad lifting $\overline{T}^{\rho}$ of $\overline{T}$ along the forgetful functor (U, BU)-**dialg** $\to \mathcal{S}$.*

Thus, the monad $\overline{T}^{\rho}$, which is actually lifted from $\overline{T}$ via structural recursion, associates to each operational model $UX \to BUX$ a behavioural interpretation $U\overline{T}X \to BU\overline{T}X$ of the monadic structure as specified by the operational rule ρ.

Corollary 1.3 *Assume that $\mathcal{S}$ has an initial object 0, and that U preserves it. If it exists, the U-structured B-bisimilarity $\overline{T}0 \leftarrow E \to \overline{T}0$ on the operational model $U\overline{T}0 \to BU\overline{T}0$ (arising from the unique map $U0 \to BU0$) is a $\overline{T}$-congruence on $\overline{T}0$. Consequently, the span $T0 \xleftarrow{\sim} U\overline{T}0 \leftarrow UE \to U\overline{T}0 \xrightarrow{\sim} T0$ is a T-congruence on the initial Σ-algebra $T0$.*

Adjoint mathematical universes. The mathematical universes of discourse in the examples at hand (see [9] and Section 2 below) support plenty of further structure. For the purpose of the development here we highlight the following: (i) the category $\mathcal{S}$ has binary products; (ii) the endofunctor Σ on $\mathcal{M}$ arises as the restriction of an endofunctor $\overline{\Sigma}$ on $\mathcal{S}$, and $\overline{T}$ is the free monad on $\overline{\Sigma}$; (iii) the functor U preserves binary products and has a right adjoint, say $V : \mathcal{M} \to \mathcal{S}$ (so that the categories (U, BU)-**dialg** and VBU-**coalg** are isomorphic). Thus we have the following adjoint situation



where $\widetilde{B} = VBU$. Hence, one can consider, as did Fiore and Turi [9], abstract rules of the form

$$\overline{\Sigma}\big((-) \times \widetilde{B}(-)\big) \Longrightarrow \widetilde{B}\overline{T}(-) : \mathcal{S} \to \mathcal{S} \qquad (8)$$

as in the setting of Turi and Plotkin [20].

The following result relates the operational semantics advocated here based on rules of the form (7) to the more general one with rules of the form (8).

Theorem 1.4 *Every ρ as in (7) induces a $\overline{\rho}$ of the form (8) such that the monad liftings $\overline{T}^{\rho}$ and $\overline{T}^{\overline{\rho}}$ of the monad $\overline{T}$ respectively to (U, BU)-**dialg** and to $\widetilde{B}$-**coalg** are isomorphic.*

Indeed, for $X \in \mathcal{S}$, one lets $\overline{\rho}_X$ be the right adjunct of the composite $U\overline{\Sigma}(X \times VBUX) \xrightarrow{\sim} \Sigma(UX \times UVBUX) \xrightarrow{\Sigma(\mathrm{id} \times \varepsilon_{BUX})} \Sigma(UX \times BUX) \xrightarrow{\rho_X} BTUX \xrightarrow{\sim} BU\overline{T}X$, where ε denotes the counit of the adjunction $U \dashv V$.

2. Abstract syntax

We develop a theory of binding signatures suitable for name-passing systems. We investigate models for the signatures and consider how morphisms between model categories relate different kinds of model.

2.1. Signatures and their models

Binding signatures. The syntax of the π-calculus is built from various operators. For instance, the input phrase $c(a).P$, which will be written $\mathsf{inp}(c, \langle a\rangle P)$, has one name parameter c and one term parameter P with name a bound; the output phrase $\bar{c}d.P$ will be written $\mathsf{out}(c, d, P)$, it has two name parameters c, d, and one term parameter P with no names bound. We will also use the restriction phrase $\nu a.P$, written $\mathsf{res}(\langle a\rangle P)$, with one term parameter with a name bound in it; and the parallel phrase $P|Q$, written $\mathsf{par}(P, Q)$, which has two term parameters, neither with any names bound.

The following notion of signature, already used by Fiore and Turi [9], can be seen as an extension of that of Fiore, Plotkin and Turi [7] to allow name parameters, or alternatively as a restricted form of nominal-logic signature as introduced by Pitts [15].

Definition 2.1 *A* binding signature Σ *consists of a finite set* OP_Σ *of operators together with, for each operator* $\mathrm{op} \in \mathrm{OP}_\Sigma$*, a* name-arity $\mathrm{ar_n}(\mathrm{op}) \in \mathbb{N}$ *and a* term-arity $\mathrm{ar_t}(\mathrm{op}) \in \mathbb{N}$*. To each $j \in [1, \mathrm{ar_t}(\mathrm{op})]$ is associated a* binding depth $\mathrm{bdep_{op}}(j) \in \mathbb{N}$.

Note for instance that for the fragment of the π-calculus recalled above we have a signature Σ_π with operators $\mathrm{OP}_{\Sigma_\pi} = \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{par}\}$; arities are assigned as follows.

op	$\mathrm{ar_n}(\mathrm{op})$	$\mathrm{ar_t}(\mathrm{op})$	dep
inp	1	1	$\mathrm{bdep}_{\mathsf{inp}}(1) = 1$
out	2	1	$\mathrm{bdep}_{\mathsf{out}}(1) = 0$
res	0	1	$\mathrm{bdep}_{\mathsf{res}}(1) = 1$
par	0	2	$\mathrm{bdep}_{\mathsf{par}}(j) = 0 \quad (j = 1, 2)$

Signature models. Signatures for name-passing admit interpretation in a variety of categories.

Definition 2.2 *A* model category *(for binding signatures) is a category $\mathcal{C}$ with (i) finite products and coproducts, (ii) a distinguished object $\mathcal{N}_{\mathcal{C}} \in \mathcal{C}$ representing names, and (iii) an endofunctor $[\mathcal{N}]_{\mathcal{C}}$ on $\mathcal{C}$ representing name binding.*

Definition 2.3 *A* model *for a binding signature Σ in a model category $\mathcal{C}$ is an algebra for the endofunctor $\Sigma_{\mathcal{C}}$ on $\mathcal{C}$ given as follows.*

$$\Sigma_{\mathcal{C}}(-) = \coprod_{\mathrm{op} \in \mathrm{OP}_\Sigma} \left({\mathcal{N}_{\mathcal{C}}}^{\mathrm{ar_n}(\mathrm{op})} \times \prod_{j \in \mathrm{ar_t}(\mathrm{op})} {[\mathcal{N}]_{\mathcal{C}}}^{\mathrm{bdep_{op}}(j)}(-) \right)$$

In particular, for the π-calculus fragment introduced above, we have the following endofunctor.

$$\begin{array}{rcccc} \Sigma_{\pi,\mathcal{C}}(-) = & \mathcal{N}_{\mathcal{C}} \times [\mathcal{N}]_{\mathcal{C}}(-) & + & \mathcal{N}_{\mathcal{C}} \times \mathcal{N}_{\mathcal{C}} \times (-) \\ + & [\mathcal{N}]_{\mathcal{C}}(-) & + & (-) \times (-) \end{array}$$

Typically, and this is the case in all our examples, $\mathcal{C}$ is cartesian closed and has colimits of ω-chains that are preserved by $[\mathcal{N}]_{\mathcal{C}}$; thus the free monad $T_{\mathcal{C}}$ on $\Sigma_{\mathcal{C}}$ can be constructed in the usual fashion to provide a free model for the signature. A model for Σ in $\mathcal{C}$ can be equivalently given by an algebra for the monad $T_{\mathcal{C}}$.

Raw syntax. Fixing a set $\mathtt{N}$ of name meta-variables, we can consider the model category $\mathbf{Set}_{\mathtt{N}} = \mathbf{Set}$ with name object $\mathcal{N}_{\mathbf{Set}_{\mathtt{N}}} = \mathtt{N}$ and with abstraction endofunctor $[\mathcal{N}]_{\mathbf{Set}_{\mathtt{N}}} = \mathtt{N} \times (-)$. Then for any set X the set $T_{\mathbf{Set}_{\mathtt{N}}} X$ contains all raw Σ-terms with name variables from $\mathtt{N}$ and term variables from X, where α-equivalent terms are not identitifed.

Notation. Elements $(\mathtt{a}, \mathtt{x})$ of $[\mathcal{N}]_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X}$ will be written suggestively as $\langle \mathtt{a} \rangle \mathtt{x}$, even though no α-equivalence is imposed on them.

2.2. Abstract syntax with variable binding

We give two examples of model categories capturing syntax up-to α-equivalence. The first one is the model of nominal sets of Gabbay and Pitts [10]; the second one is an extension of the former which is a sheaf subcategory of the model of Fiore, Plotkin and Turi [7].

2.2.1. Nominal sets

We briefly review and set notation for the theory of nominal sets.

Nominal sets. Throughout the paper we fix an infinite countable set $\mathcal{N}$ of names. Recall that a left action of the symmetric group $\mathsf{Sym}(\mathcal{N})$ on $\mathcal{N}$ is a set X equipped with a function $\bullet_X : \mathsf{Sym}(\mathcal{N}) \times X \to X$ (written infix) which is such that for any element $x \in X$ we have $\mathrm{id}_{\mathcal{N}} \bullet_X x = x$ and, for any $\sigma, \tau \in \mathsf{Sym}(\mathcal{N})$, that $(\tau\sigma) \bullet_X x = \tau \bullet_X (\sigma \bullet_X x)$.

A finite set of names $C \subseteq_{\mathrm{f}} \mathcal{N}$ is said to support an element x of a $\mathsf{Sym}(\mathcal{N})$-action $(X, \bullet_X)$ if every permutation $\sigma \in \mathsf{Sym}(\mathcal{N})$ that fixes every element of C also fixes x. A *nominal set* is a $\mathsf{Sym}(\mathcal{N})$-action in which every element has finite support. We let $\mathbf{Nom}$ be the category of nominal sets and equivariant functions, *i.e.* functions compatible with the actions.

Notation. For names $a, b \in \mathcal{N}$ be write $[a \leftrightarrow b]$ for the permutation on $\mathcal{N}$ that swaps a and b and fixes the other names. The expression "$a\#x$", that stands for "a is fresh for x", means that there is a support of x that does not contain a.

Constructions. The category $\mathbf{Nom}$ has colimits and finite limits, and the functor $|-| : \mathbf{Nom} \to \mathbf{Set}$ that forgets the $\mathsf{Sym}(\mathcal{N})$-action structure preserves them. So, in particular, sums and products are inherited from $\mathbf{Set}$.

The set $\mathcal{N}$ of names has nominal-set structure; the action is given by evaluation, *i.e.* $\sigma \bullet_{\mathcal{N}} a = \sigma(a)$. For any nominal set X we have the nominal set $[\mathcal{N}]X$ of the names abstracted in X. The carrier set is the quotient

$$[\mathcal{N}]X = (\mathcal{N} \times X)/{\sim_{[\mathcal{N}]X}}$$

where $(a, x) \sim_{[\mathcal{N}]X} (a', x')$ if for any $b \in \mathcal{N}$ such that $b\#x$ and $b\#x'$ we have $[b \leftrightarrow a] \bullet_X x = [b \leftrightarrow a'] \bullet_X x'$. We write $\langle a \rangle x$ for the equivalence class $(a, x)_{[\mathcal{N}]X}$. The $\mathsf{Sym}(\mathcal{N})$-action of $[\mathcal{N}]X$ is inherited from that of the product.

These constructions make $\mathbf{Nom}$ into a model category.

Syntax up-to α-equivalence. For any binding signature Σ and nominal set X, the nominal set $T_{\mathbf{Nom}} X$ contains all Σ-terms with name variables from $\mathcal{N}$ and term variables from X; here, α-equivalent terms are identified in accordance with the intrinsic notion of α-equivalence in X. Indeed, a $T_{\mathbf{Nom}}$-algebra is a model of the nominal-logic signature underlying Σ, in the sense of Pitts [15].

2.2.2. Nominal substitutions

While nominal sets support actions of bijective renamings useful for modelling α-equivalence, to give semantics to name communication it is necessary to further consider arbitrary name substitutions.

Definition 2.4 *A* nominal substitution *is a nominal set X together with an equivariant function $\mathsf{sub}_X : \mathcal{N} \times [\mathcal{N}]X \to X$ that satisfies the following four axioms, where, for clarity, we write $[b/a]x$ for $\mathsf{sub}_X(b, \langle a \rangle x)$.*

1. *Identity:* $[a/a]x = x$.
2. *Weakening:* $[b/a]x = x$, *whenever* $a\#x$.
3. *Contraction:* $[c/b][b/a]x = [c/b][c/a]x$.
4. *Permutation:* $[d/b][c/a]x = [c/a][d/b]x$, *whenever* $c \neq b \neq a \neq d$.

(Note that by definition of $[\mathcal{N}]X$ we have that $[b/a]x = [b/z][a \leftrightarrow z]x$, whenever $z\#x$.)

A homomorphism of nominal substitutions is an equivariant function between the underlying nominal sets that respects the nominal substitution. Thus we have a category $\mathbf{NomSub}$ of nominal substitutions. This category is complete and cocomplete, and the forgetful functor $\mathbf{NomSub} \to \mathbf{Nom}$ preserves limits and colimits. The object $\mathcal{N}$ of names in $\mathbf{Nom}$ has a unique nominal-substitution structure, while for any nominal substitution X the nominal set $[\mathcal{N}]X$ has a nominal-substitution structure such that for $x \in X$ and $a, b, c \in \mathcal{N}$ with $a \neq c \neq b$ we have $[b/a]\langle c\rangle x = \langle c\rangle[b/a]x$. In this way the endofunctor $[\mathcal{N}]$ on $\mathbf{Nom}$ is lifted along the forgetful functor $\mathbf{NomSub} \to \mathbf{Nom}$.

Thus $\mathbf{NomSub}$ is a model category and we are also able to consider syntax there. In fact, the syntax thus obtained is as in the model category $\mathbf{Nom}$, but equipped with a name-substitution action (see the following subsection).

Explicit name substitutions. For every signature Σ there is an extended signature Σ^{sub} which is the same as Σ but has one extra operator: the substitution operator sub, that takes one name parameter and one term parameter with a name bound in it. For any model category $\mathcal{C}$, we write $\Sigma^{\mathsf{sub}}_{\mathcal{C}}$ for the endofunctor on $\mathcal{C}$ generated by the signature Σ^{sub}. We further write $T^{\mathsf{sub}}_{\mathcal{C}}$ for the free monad on $\Sigma^{\mathsf{sub}}_{\mathcal{C}}$, when it exists.

In the particular case of the model category $\mathbf{NomSub}$, we have a natural transformation $\Sigma^{\mathsf{sub}}_{\mathbf{NomSub}} \to \Sigma_{\mathbf{NomSub}}$ which evaluates the explicit substitutions using the notion internal to nominal substitutions. This induces a monad functor $T_{\mathbf{NomSub}} \to T^{\mathsf{sub}}_{\mathbf{NomSub}}$, and hence, in particular, morphisms $T^{\mathsf{sub}}_{\mathbf{NomSub}}X \to T_{\mathbf{NomSub}}X$ natural for X in $\mathbf{NomSub}$.

2.3. Morphisms of model categories

By considering morphisms between model categories we are able to relate the different models of syntax.

Definition 2.5 *A morphism $(F, f, \phi) : \mathcal{D} \to \mathcal{C}$ between model categories is given by a product-preserving functor $F : \mathcal{D} \to \mathcal{C}$ together with a morphism $f : \mathcal{N}_{\mathcal{C}} \to F(\mathcal{N}_{\mathcal{D}})$ in $\mathcal{C}$ and a natural transformation $\phi : [\mathcal{N}]_{\mathcal{C}}\, F \to F\, [\mathcal{N}]_{\mathcal{D}}$.*

For any binding signature Σ, such a morphism induces a natural transformation $\Sigma_{\mathcal{C}}F \to F\Sigma_{\mathcal{D}}$. If the free monads on $\Sigma_{\mathcal{C}}$ and $\Sigma_{\mathcal{D}}$ exist then they induce, by structural recursion, a monad functor $(\mathcal{D}, T_{\mathcal{D}}) \to (\mathcal{C}, T_{\mathcal{C}})$.

We now use this framework to explain how raw syntax is quotiented by α-equivalence.

Instantiating raw syntax. We fix a set $\mathtt{N}$ of name meta-variables and consider an instantiation function $I : \mathtt{N} \to \mathcal{N}$ sending name meta-variables to names. We have a morphism of model categories $\mathbf{NomSub} \to \mathbf{Set}_{\mathtt{N}}$ given by the forgetful functor $|-| : \mathbf{NomSub} \to \mathbf{Set}$ together with the function $I : \mathtt{N} \to |\mathcal{N}|$ and the natural transformation $\mathtt{N} \times |-| \to |\mathcal{N}| \times |-| \twoheadrightarrow |[\mathcal{N}](-)|$. Thus for any binding signature we have an induced monad functor $T_{\mathbf{NomSub}} \to T_{\mathbf{Set}_{\mathtt{N}}}$, and so for any nominal substitution X we have a function $T_{\mathbf{Set}_{\mathtt{N}}}|X| \to |T_{\mathbf{NomSub}}X|$ that in fact converts terms of raw syntax into terms of abstract syntax up-to α-equivalence.

Lifting syntax from $\mathbf{Nom}$ to $\mathbf{NomSub}$. For any morphism $(F, f, \phi) : \mathcal{D} \to \mathcal{C}$ of model categories, if the functor F preserves coproducts and f and ϕ are isomorphisms then the induced natural transformation $\Sigma_{\mathcal{C}}F \to F\Sigma_{\mathcal{D}}$ is an isomorphism. If, in addition, the free monads on $\Sigma_{\mathcal{C}}$ and $\Sigma_{\mathcal{D}}$ exist and F has a right adjoint then the further induced natural transformation $T_{\mathcal{C}}F \to FT_{\mathcal{D}}$ is also an isomorphism. That is, $T_{\mathcal{D}}$ is a monad lifting of $T_{\mathcal{C}}$ along F.

We can apply this to relate models in $\mathbf{Nom}$ with models in $\mathbf{NomSub}$. Indeed, a morphism $\mathbf{NomSub} \to \mathbf{Nom}$ of model categories is given by the forgetful functor $|-| : \mathbf{NomSub} \to \mathbf{Nom}$ together with the identities $\mathcal{N}_{\mathbf{Nom}} = |\mathcal{N}_{\mathbf{NomSub}}|$ and $[\mathcal{N}]_{\mathbf{Nom}}|-| = |[\mathcal{N}]_{\mathbf{NomSub}}(-)|$. Moreover, $|-|$ has a right adjoint and so for any binding signature Σ we have isomorphisms $\Sigma_{\mathbf{Nom}}|X| \xrightarrow{\sim} |\Sigma_{\mathbf{NomSub}}X|$ and $T_{\mathbf{Nom}}|X| \xrightarrow{\sim} |T_{\mathbf{NomSub}}X|$ natural for X in $\mathbf{NomSub}$.

3. Behavioural models of name-passing

We introduce coalgebraic models for both ground and wide-open bisimulation.

3.1. Ground bisimulation

We study ground bisimulation, *viz.* bisimulation for which the only relevant input transitions are those involving fresh data.

Deterministic ground behaviour. We introduce an endofunctor on $\mathbf{Nom}$ for deterministic ground behaviour:

$$\begin{array}{rcll} L_{\mathrm{g}}(-) = & & \mathcal{N} \times [\mathcal{N}](-) & \text{(input)} \\ & + & \mathcal{N} \times \mathcal{N} \times (-) & \text{(output)} \\ & + & \mathcal{N} \times [\mathcal{N}](-) & \text{(bound output)} \\ & + & (-) & \text{(silent action)} \end{array}$$

Thus an L_{g}-coalgebra is a nominal set of states together with an equivariant function assigning to each state either: an input (in) behaviour (*i.e.* a channel name and a resumption state with one name bound); an output (out) behaviour, with the output data paired rather than bound; or a bound output (bout) or silent (tau) behaviour.

Ground behaviour. We introduce non-determinism into the model by interpreting the theory of (finite-)join semi-lattices in the category $\mathbf{Nom}$. For any nominal set X, we write $\mathcal{P}_{\mathrm{f}}X$ for the carrier of the free join semi-lattice on

X. This free construction is a monad lifting of the free join semi-lattice monad on $\mathbf{Set}$ (whose underlying functor is the covariant finite powerset) along the forgetful functor $\mathbf{Nom} \to \mathbf{Set}$.

We thus define an endofunctor B_{g} on $\mathbf{Nom}$ for (non-deterministic) *ground behaviour*:

$$B_{\mathrm{g}} = \mathcal{P}_{\mathrm{f}} L_{\mathrm{g}} \quad .$$

In the remainder of this subsection we put this theory in a more concrete perspective.

Nominal transition systems. We start by introducing a notion of nominal transition system. First, for each nominal set N we have a nominal set of labels over N:

$$Lab(N) \;=\; N \times N \;+\; N \times N \;+\; N \times N \;+\; 1 \quad .$$

The components of this sum correspond respectively to input action (written $c(z)$), output action ($\bar{c}d$), bound output action ($\bar{c}(z)$), and silent action (τ). For any label $l \in Lab(N)$ the binding names $\mathsf{bn}(l)$ and free names $\mathsf{fn}(l)$ are defined as usual: $\mathsf{bn}(c(z)) = \mathsf{bn}(\bar{c}(z)) = \{z\}$ and $\mathsf{bn}(\bar{c}d) = \mathsf{bn}(\tau) = \emptyset$; whilst $\mathsf{fn}(c(z)) = \mathsf{fn}(\bar{c}(z)) = \{c\}$, $\mathsf{fn}(\bar{c}d) = \{c, d\}$, and $\mathsf{fn}(\tau) = \emptyset$.

Definition 3.1 *A* ground transition system *is a nominal set X together with an equivariant relation $\longrightarrow \subseteq X \times Lab(\mathcal{N}) \times X$ for which binding names are always fresh; i.e., if $x \xrightarrow{\;l\;} x'$ then $\mathsf{bn}(l)\#x$.*

Every B_{g}-coalgebra (X, h) induces a ground transition system with carrier the underlying nominal set X and with transition relation $\longrightarrow_h \subseteq X \times Lab(\mathcal{N}) \times X$ given as follows.

If $\mathsf{in}(c, \langle z \rangle x') \in h(x)$ and $z\#x$ then $x \xrightarrow{c(z)}_h x'$.

If $\mathsf{out}(c, d, x') \in h(x)$ then $x \xrightarrow{\bar{c}d}_h x'$.

If $\mathsf{bout}(c, \langle z \rangle x') \in h(x)$ and $z\#x$ then $x \xrightarrow{\bar{c}(z)}_h x'$.

If $\mathsf{tau}(x') \in h(x)$ then $x \xrightarrow{\tau}_h x'$.

Note that this transition relation is finite-branching, up to renaming of bound names.

Ground bisimulation. We consider ground bisimulations in the sense of Sangiorgi [17, Def. 2.1] that are closed under bijective renamings.

Definition 3.2 *A* ground bisimulation *between two ground transition systems $(X, \longrightarrow)$ and $(X', \longrightarrow')$ is an equivariant relation $R \subseteq X \times X'$ such that the following holds: for any $(x, x') \in R$ and any label $l \in Lab(\mathcal{N})$ such that $\mathsf{bn}(l)\#(x, x')$, (i) if $x \xrightarrow{\;l\;} y$ then there is $y' \in X'$ such that $x' \xrightarrow{\;l\;}' y'$ and $(y, y') \in R$, and (ii) if $x' \xrightarrow{\;l\;}' y'$ then there is $y \in X$ such that $x \xrightarrow{\;l\;} y$ and $(y, y') \in R$.*

Proposition 3.3 *An equivariant relation between the carriers of two B_{g}-coalgebras is a B_{g}-bisimulation if and only if it is a ground bisimulation between the induced ground transition systems.*

3.2. Wide-open bisimulation

Operational models. We consider the operational models given by structured B_{g}-coalgebras with respect to the forgetful functor $|-| : \mathbf{NomSub} \to \mathbf{Nom}$; that is, B_{g}-coalgebras together with a name-substitution action on their carrier.

Interestingly, this notion of operational model has essentially appeared before, though in a different guise. Indeed, to give a structured B_{g}-coalgebra is to give an $\mathcal{N}$-LTS in the sense of Cattani and Sewell [5, Def. 3.4] that additionally satisfies an image finiteness condition, and for which the carrier presheaf preserves pullbacks of injections. We explored correspondences of this kind in [8].

Wide-open bisimulation. The associated notion of behavioural equivalence, *viz.* structured B_{g}-bisimulation, has a simple description as a form of open bisimulation.

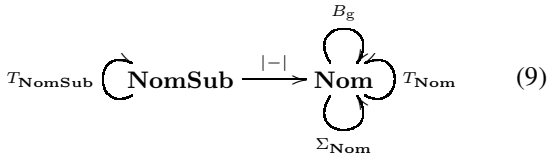
Definition 3.4 *Let X and Y be nominal substitutions. A* wide-open bisimulation *between ground transition systems on the nominal sets $|X|$ and $|Y|$ is a ground bisimulation $R \subseteq |X| \times |Y|$ that is substitution closed in the sense that, for any $a, b \in \mathcal{N}$, $x \in X$, $y \in Y$, if $x\,R\,y$ then $[b/a]x\;R\;[b/a]y$.*

Proposition 3.5 *The notions of B_{g}-bisimulation relation between structured B_{g}-coalgebras (X, h) and (Y, k) and of wide-open bisimulation between the induced ground transition systems $(|X|, \longrightarrow_h)$ and $(|Y|, \longrightarrow_k)$ coincide.*

Furthermore, wide-open bisimilarity is characterised as the structured B_{g}-bisimilarity.

4. Operational rules for name-passing

We proceed to give concrete rules for name-passing systems that have been extracted from abstract ones. Recall then from Sections 2 and 3 that for every binding signature Σ we obtain a mathematical universe as follows



(9)

(which is in fact an adjoint mathematical universe).

Following the mathematical operational semantics of Section 1, an abstract rule is a natural transformation of the form:

$$\Sigma_{\mathbf{Nom}}\big(\,|-| \times B_{\mathrm{g}}\,|-|\,\big) \Longrightarrow B_{\mathrm{g}} T_{\mathbf{Nom}} |-| \qquad (10)$$

between functors $\mathbf{NomSub} \to \mathbf{Nom}$.

(We stress that rules merely arising from the model $(\mathbf{Nom}, \Sigma_{\mathbf{Nom}}, B_{\mathrm{g}})$ are not expressive enough, crucially to support name communication.)

4.1. Concrete rules

We introduce rule structures as a formalisation of the usual, concrete notion of rule.

Informal presentation. Systems such as the π-calculus involve rules of the form

$$\frac{\mathtt{x}_1 \xrightarrow{l_1} \mathtt{y}_1 \quad \mathtt{x}_2 \xrightarrow{l_2} \mathtt{y}_2 \quad \dots}{\mathrm{op}(\mathtt{c}, \langle \mathtt{a} \rangle \mathtt{x}, \dots) \xrightarrow{l} t} \quad \begin{array}{l} \text{— } \textit{premises} \\ \text{— } \textit{conclusion} \end{array}$$

where op is an operator of a binding signature; $\mathtt{a}$, $\mathtt{c}$ are name meta-variables; $\mathtt{x}, \mathtt{x}_j, \mathtt{y}_j$ are term meta-variables; l, l_j are labels (for input, output, bound output, and silent actions) involving name meta-variables; and where t is a compound term built out of operators from the signature together with name and term meta-variables.

For first examples consider the π-calculus rules for input and scope closure.

$$\textit{(input)}\ \frac{}{\mathsf{inp}(\mathtt{c}, \langle \mathtt{a} \rangle \mathtt{x}) \xrightarrow{\mathtt{c}(\mathtt{a})} \mathtt{x}}$$

$$\textit{(close)}\ \frac{\mathtt{x} \xrightarrow{\bar{\mathtt{c}}(\mathtt{a})} \mathtt{y} \qquad \mathtt{x}' \xrightarrow{\mathtt{c}(\mathtt{a})} \mathtt{y}'}{\mathsf{par}(\mathtt{x}, \mathtt{x}') \xrightarrow{\tau} \mathsf{res}(\langle \mathtt{a} \rangle \mathsf{par}(\mathtt{y}, \mathtt{y}'))}$$

While the terms of the language are considered up-to α-equivalence, it makes little sense to consider α-equivalence in the rules themselves. Indeed, rules are templates where no actual binding takes place. In the terminology of Section 2: rules are built of raw syntax.

Note also that rules may involve renamings in the right hand side of the conclusion; consider for instance the π-calculus rule for communication.

$$\textit{(com)}\ \frac{\mathtt{x} \xrightarrow{\bar{\mathtt{c}}\mathtt{d}} \mathtt{y} \qquad \mathtt{x}' \xrightarrow{\mathtt{c}(\mathtt{a})} \mathtt{y}'}{\mathsf{par}(\mathtt{x}, \mathtt{x}') \xrightarrow{\tau} \mathsf{par}(\mathtt{y}, [\mathtt{d}/\mathtt{a}]\mathtt{y}')}$$

For this reason explicit substitutions (as in Section 2.2.2) are needed.

Formal rule structures. Rule structures for a binding signature Σ are defined as follows.

Definition 4.1 *Let* $\mathtt{N}$ *and* $\mathtt{X}$ *be finite sets, and write* $Lab(\mathtt{N})$ *for the set of labels over* $\mathtt{N}$.

1. *A* rule structure *over* $(\mathtt{N}, \mathtt{X})$ *is a finite set of premises over* $(\mathtt{N}, \mathtt{X})$ *together with a conclusion over* $(\mathtt{N}, \mathtt{X})$.
2. *A* premise *over* $(\mathtt{N}, \mathtt{X})$ *is a triple in* $\mathtt{X} \times Lab(\mathtt{N}) \times \mathtt{X}$*; the components are referred to as the source, the label, and the target of the premise.*
3. *A* conclusion *over* $(\mathtt{N}, \mathtt{X})$ *is a triple in*

$$\Sigma_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X} \times Lab(\mathtt{N}) \times T^{\mathsf{sub}}_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X}$$

The components are again referred to as the source, the label, and the target of the conclusion.

Interpretation of rules. In Section 4.3 we explain how a rule structure induces an operational semantics. For the time being, we suggest that a rule be used to define a transition system by induction in the usual way, subject to two conventions about how the rule can be instantiated: (i) Names, when instantiated, must be as distinct as the corresponding name meta-variables are in the rule. (ii) Binding data on the conclusion label must be fresh for the conclusion source.

These two conventions are to be thought of as side conditions that are implicit in every rule. For instance, consider the π-calculus rules for mismatch and parallel transition.

$$\textit{(mm)}\ \frac{\mathtt{x} \xrightarrow{\tau} \mathtt{y}}{\mathsf{mm}(\mathtt{c}, \mathtt{d}, \mathtt{x}) \xrightarrow{\tau} \mathtt{y}} \qquad \textit{(par)}\ \frac{\mathtt{x} \xrightarrow{\bar{\mathtt{c}}(\mathtt{a})} \mathtt{y}}{\mathsf{par}(\mathtt{x}, \mathtt{x}') \xrightarrow{\bar{\mathtt{c}}(\mathtt{a})} \mathsf{par}(\mathtt{y}, \mathtt{x}')}$$

Convention (i) eliminates the need for the usual side condition $\mathtt{c} \neq \mathtt{d}$ on the rule *(mm)*; convention (ii) covers the usual side condition $\mathtt{a} \notin \mathsf{fn}(\mathtt{x}')$ in rule *(par)*.

On the other hand, because of these conventions, some duplication in the rules may be necessary. For instance, to attain the usual output behaviour it is necessary to split the usual rule in two:

$$\textit{(out-eq)}\ \frac{}{\mathsf{out}(\mathtt{c}, \mathtt{c}, \mathtt{x}) \xrightarrow{\bar{\mathtt{c}}\mathtt{c}} \mathtt{x}} \qquad \textit{(out-neq)}\ \frac{}{\mathsf{out}(\mathtt{c}, \mathtt{d}, \mathtt{x}) \xrightarrow{\bar{\mathtt{c}}\mathtt{d}} \mathtt{x}}$$

One can envisage a notion of rule structure with explicit side conditions from which a finite family of rule structures in the form of Definition 4.1 can be derived, but we will not dwell on that here.

4.2. Rule format for name-passing

We introduce conditions on rule structures explicitly designed to capture concrete rules that give rise to abstract rules, and hence to guarantee that wide-open bisimilarity is a congruence for the induced transition systems.

Throughout this section we fix a rule structure $\mathtt{R}$ over $(\mathtt{N}, \mathtt{X})$, with premise set Prem and conclusion with source

$$\underline{\mathtt{op}}\left((\underline{\mathtt{c}}_i)_{i \in [1, \mathrm{ar_n}(\underline{\mathtt{op}})]}, \left(\langle \underline{\mathtt{a}}^j_k \rangle_{k \in [1, \mathrm{bdep}_{\underline{\mathtt{op}}}(j)]} \underline{\mathtt{x}}_j \right)_{j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]} \right),$$

label $\underline{\mathtt{l}}$, and target $\underline{\mathtt{tar}}$. (We distinguish entities appearing in the conclusion by underlining them.)

GSOS-like conditions:

1. Every term meta-variable appears exactly once in the conclusion source and the premise targets.
2. The source of every premise appears in the conclusion source.

Conditions relating to name binding:

3. For each term meta-variable in the conclusion source, the binding names are distinct.
$$\forall j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})],\ k, k' \in [1, \mathrm{bdep}_{\underline{\mathtt{op}}}(j)].$$
$$\underline{\mathtt{a}}^j_k = \underline{\mathtt{a}}^j_{k'} \implies k = k'$$
4. No free names bind in the conclusion source.
$$\forall j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})],\ k \in [1, \mathrm{bdep}_{\underline{\mathtt{op}}}(j)].$$
$$\underline{\mathtt{a}}^j_k \notin \mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem})$$
5. For each premise, binding names in the label are fresh for the source.
$$\forall (\mathtt{x}, \mathtt{l}, \mathtt{y}) \in \mathrm{Prem}.\ \mathtt{bn}(\mathtt{l}) \cap \mathtt{FN}(\mathtt{x}) = \emptyset$$
6. Free names of the conclusion label are free.
$$\mathtt{fn}(\underline{\mathtt{l}}) \subseteq \mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem})$$
7. Bound names of the conclusion label are fresh.[1]
$$\mathtt{bn}(\underline{\mathtt{l}}) \cap \mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) = \emptyset$$
8. No names become unbound in the induced transition.
$$\mathtt{FN}(\underline{\mathtt{tar}}) \subseteq \mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) \cup \mathtt{bn}(\underline{\mathtt{l}})$$
9. The conclusion target is well-formed.
$$\mathtt{WF}(\underline{\mathtt{tar}})$$

Figure 1. Conditions on rule structures.

Conditions on rule structures. In Figure 1 we present conditions that we expect to hold of rule structures. Conditions (1–2) are the conditions of the GSOS format [4] considered in this context. Conditions (3–9) relate to the freshness of the names that appear in binding position. To specify these conditions formally it is necessary to formalise the notions of bound and free names that are implicit in rule structures.

Associating names to variables. From here on we assume that Conditions (1–2) hold of $\mathtt{R}$. We then assign to each term meta-variable $\mathtt{x} \in \mathtt{X}$ the set $\mathtt{BN}(\mathtt{x}) \subseteq \mathtt{N}$ of name meta-variables that are binding in $\mathtt{x}$. For instance, in the *(input)* rule above, $\mathtt{BN}(\mathtt{x}) = \{\mathtt{a}\}$, and in the *(par)* rule, $\mathtt{BN}(\mathtt{x}) = \mathtt{BN}(\mathtt{x}') = \emptyset$, while $\mathtt{BN}(\mathtt{y}) = \{\mathtt{a}\}$.

To define $\mathtt{BN}$ we use the fact that since Conditions (1–2) are satisfied we have a bijection

$$\mathtt{X} \cong [1, \mathrm{ar_t}(\underline{\mathtt{op}})] + \coprod_{j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]} \left\{ (\mathtt{x}, \mathtt{l}, \mathtt{y}) \in \mathrm{Prem} \;\middle|\; \mathtt{x} = \underline{\mathtt{x}}_j \right\}$$

whose inverse maps $j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]$ to $\underline{\mathtt{x}}_j$, and $\mathsf{inj}_j(\mathtt{x}, \mathtt{l}, \mathtt{y})$ to $\mathtt{y}$. Now:

- For $j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]$ we let
$$\mathtt{BN}(\underline{\mathtt{x}}_j) = \left\{ \underline{\mathtt{a}}^j_k \;\middle|\; k \in [1, \mathrm{bdep}_{\underline{\mathtt{op}}}(j)] \right\} \quad .$$
- For $(\mathtt{x}, \mathtt{l}, \mathtt{y}) \in \mathrm{Prem}$ we let
$$\mathtt{BN}(\mathtt{y}) = \mathtt{BN}(\mathtt{x}) \cup \mathtt{bn}(\mathtt{l}) \quad .$$

Finally, we write $\mathtt{BN}(\underline{\mathtt{src}}, \mathrm{Prem}) \subseteq \mathtt{N}$ for the set

$$\mathtt{BN}(\underline{\mathtt{src}}, \mathrm{Prem}) = \bigcup_{\mathtt{x} \in \mathtt{X}} \mathtt{BN}(\mathtt{x})$$

of all name meta-variables that appear in binding position in the conclusion source or the premise labels.

We now associate to each variable $\mathtt{x} \in \mathtt{X}$ a set $\mathtt{FN}(\mathtt{x}) \subseteq \mathtt{N}$, which approximates (from the point of view of the rule) the names which appear free when the variable $\mathtt{x}$ is instantiated. To do this we first define the set $\mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) \subseteq \mathtt{N}$ that approximates the names that will be free in the conclusion source when it is instantiated.

$$\mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) = \left\{ \underline{\mathtt{c}}_i \;\middle|\; i \in [1, \mathrm{ar_n}(\underline{\mathtt{op}})] \right\} \cup \bigcup_{(\mathtt{x},\mathtt{l},\mathtt{y}) \in \mathrm{Prem}} \mathtt{fn}(\mathtt{l}) \setminus \mathtt{BN}(\mathtt{x})$$

Finally, for any $\mathtt{x} \in \mathtt{X}$, we let

$$\mathtt{FN}(\mathtt{x}) = \mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) \cup \mathtt{BN}(\mathtt{x}) \quad .$$

The function $\mathtt{FN}$ extends to compound terms with explicit substitutions. For $\mathtt{t} \in T^{\mathsf{sub}}_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X}$ with

$$\mathtt{t} = \mathtt{op} \left((\mathtt{c}_i)_{i \in [1, \mathrm{ar_n}(\mathtt{op})]},\ \left(\langle \mathtt{a}^j_k \rangle_{k \in [1, \mathrm{bdep}_{\mathtt{op}}(j)]} \mathtt{t}_j \right)_{j \in [1, \mathrm{ar_t}(\mathtt{op})]} \right)$$

we define $\mathtt{FN}(\mathtt{t}) \subseteq \mathtt{N}$ by

$$\mathtt{FN}(\mathtt{t}) = \{\mathtt{c}_i \mid i \in [1, \mathrm{ar_n}(\mathtt{op})]\} \cup \bigcup \left\{ \mathtt{FN}(\mathtt{t}_j) \setminus \left\{ \mathtt{a}_k^j \mid k \in [1, \mathrm{bdep}_{\mathtt{op}}(j)] \right\} \;\middle|\; j \in [1, \mathrm{ar_t}(\mathtt{op})] \right\}$$

As an example, consider the *(close)* rule above: $\mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) = \{\mathtt{c}\}$, while $\mathtt{BN}(\mathtt{y}) = \mathtt{BN}(\mathtt{y}') = \{\mathtt{a}\}$, and so $\mathtt{FN}(\mathtt{y}) = \mathtt{FN}(\mathtt{y}') = \{\mathtt{a}, \mathtt{c}\}$. However, $\mathtt{FN}(\underline{\mathtt{tar}}) = \{\mathtt{c}\}$.

For the *(com)* rule above, we have $\mathtt{FN}(\underline{\mathtt{src}}, \mathrm{Prem}) = \mathtt{FN}(\mathtt{y}) = \{\mathtt{c}, \mathtt{d}\}$, while $\mathtt{BN}(\mathtt{y}') = \{\mathtt{a}\}$ and so $\mathtt{FN}(\mathtt{y}') = \{\mathtt{a}, \mathtt{c}, \mathtt{d}\}$. However, $\mathtt{FN}(\underline{\mathtt{tar}}) = \{\mathtt{c}, \mathtt{d}\}$.

Well-formed conclusion targets. Condition (9) asserts that the predicate WF holds of the conclusion target. Informally, this predicate requires that a binding variable is not used to bind in one term in the conclusion source and in a different term in the conclusion target. For instance, consider a strange operator taking two term parameters, the first one with a binder, and the following rule structure.

$$\textit{(strange)}\ \frac{}{\mathsf{strange}(\langle \mathtt{a} \rangle \mathtt{x}, \mathtt{x}') \xrightarrow{\tau} \mathsf{res}(\langle \mathtt{a} \rangle \mathsf{par}(\mathtt{x}, \mathtt{x}'))}$$

Here the scope of the binder a in the conclusion target encompasses both x and x′, but was previously only binding in x; thus the conclusion target is not well-formed.

Formally, the predicate WF is defined by induction on the structure of the set $T^{\mathsf{sub}}_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X}$, as follows.

- For $\mathtt{x} \in \mathtt{X}$, we always let $\mathtt{WF}(\mathtt{x})$.
- For $\mathtt{t} = \mathtt{op}\left((\mathtt{c}_i)_{i \in [1, \mathrm{ar_n}(\mathtt{op})]}, \left(\langle \mathtt{a}_k^j \rangle_{k \in [1, \mathrm{bdep}_{\mathtt{op}}(j)]} \mathtt{t}_j \right)_{j \in [1, \mathrm{ar_t}(\mathtt{op})]} \right)$, we let $\mathtt{WF}(\mathtt{t})$ if: for all $j \in [1, \mathrm{ar_t}(\mathtt{op})]$ we have $\mathtt{WF}(\mathtt{t}_j)$ and, furthermore, for all $k \in [1, \mathrm{bdep}_{\mathtt{op}}(j)]$, if $\mathtt{a}_k^j \in \mathtt{BN}(\underline{\mathtt{src}}, \mathrm{Prem})$ then for all x appearing in $\mathtt{t}_j$ we have $\mathtt{a}_k^j \in \mathtt{FN}(\mathtt{x})$.

Necessity of conditions. If one of Conditions (1–6) or (8–9) is violated then wide-open bisimilarity need not be a congruence for the induced transition system.[1] The reasons suggested by Bloom *et. al.* [4, App. A] justify Conditions (1–2). Conditions (3–6) and (8–9) are important because they disallow testing of name freshness. For instance, consider the construct if-fresh, which takes one name parameter and one term parameter with a binder, with semantics given by the rule structure

$$\textit{(if-fresh)}\ \frac{\mathtt{x} \xrightarrow{\tau} \mathtt{y}}{\mathsf{if\text{-}fresh}(\mathtt{c}, \langle \mathtt{c} \rangle \mathtt{x}) \xrightarrow{\tau} \mathtt{y}}$$

which violates Condition (4). If the if-fresh construct was allowed, the semantics in the nominal framework would be that if $a \# \langle b \rangle P$ then $\mathsf{if\text{-}fresh}(a, \langle b \rangle P)$ performs all the τ transitions of $[a \leftrightarrow b]P$, because in that case $\langle b \rangle P = \langle a \rangle [a \leftrightarrow b] P$. Thus the context $\mathsf{if\text{-}fresh}(a, \langle b \rangle \mathsf{tau}(-))$ would distinguish the π-calculus term nil from the bisimilar term $\mathsf{mm}(a, b, \mathsf{nil})$.

4.3. From concrete to abstract rules

Our aim in this section is to derive from a rule structure an abstract rule, *i.e.* a natural transformation of the form (10). To do this we consider all the possible instantiations of the rule structure.

Instantiations. For a nominal substitution X, an *instantiation* I of the rule structure R is a pair of functions $(I_{\mathrm{n}} : \mathtt{N} \to |\mathcal{N}|, I_{\mathrm{t}} : \mathtt{X} \to |X|)$ such that I_{n} is injective.

Archetypal parameter. To each instantiation I we assign an *archetypal parameter*

$$I(\underline{\mathtt{src}}, \mathrm{Prem}) \in \Sigma_{\mathbf{Nom}} \left(|X| \times B_{\mathrm{g}} |X| \right)$$

This is to be thought of as a simultaneous instantiation of both the conclusion source and of the premises.

First, for each $j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]$, we instantiate the premises with source $\underline{\mathtt{x}}_j$, by defining $I(\mathrm{Prem}[j]) \in B_{\mathrm{g}} |X|$.

$$\begin{aligned} & I(\mathrm{Prem}[j]) \\ &= \left\{ \mathsf{in}(I_{\mathrm{n}}(\mathtt{c}), \langle I_{\mathrm{n}}(\mathtt{a}) \rangle I_{\mathrm{t}}(\mathtt{y})) \mid (\underline{\mathtt{x}}_j, \mathtt{c}(\mathtt{a}), \mathtt{y}) \in \mathrm{Prem} \right\} \\ &\quad \cup \left\{ \mathsf{out}(I_{\mathrm{n}}(\mathtt{c}), I_{\mathrm{n}}(\mathtt{d}), I_{\mathrm{t}}(\mathtt{y})) \mid (\underline{\mathtt{x}}_j, \overline{\mathtt{c}}\mathtt{d}, \mathtt{y}) \in \mathrm{Prem} \right\} \\ &\quad \cup \left\{ \mathsf{bout}(I_{\mathrm{n}}(\mathtt{c}), \langle I_{\mathrm{n}}(\mathtt{a}) \rangle I_{\mathrm{t}}(\mathtt{y})) \mid (\underline{\mathtt{x}}_j, \overline{\mathtt{c}}(\mathtt{a}), \mathtt{y}) \in \mathrm{Prem} \right\} \\ &\quad \cup \left\{ \mathsf{tau}(I_{\mathrm{t}}(\mathtt{y})) \mid (\underline{\mathtt{x}}_j, \tau, \mathtt{y}) \in \mathrm{Prem} \right\} \end{aligned}$$

Now the archetypal parameter $I(\underline{\mathtt{src}}, \mathrm{Prem})$ is given by

$$\underline{\mathtt{op}}\left((I_{\mathrm{n}}(\underline{\mathtt{c}}_i))_{i \in [1, \mathrm{ar_n}(\underline{\mathtt{op}})]}, \left(\langle I_{\mathrm{n}}(\underline{\mathtt{a}}_k^j) \rangle_{k \in [1, \mathrm{bdep}_{\underline{\mathtt{op}}}(j)]} \left(I_{\mathrm{t}}(\underline{\mathtt{x}}_j), I(\mathrm{Prem}[j]) \right) \right)_{j \in [1, \mathrm{ar_t}(\underline{\mathtt{op}})]} \right)$$

Archetypal result. To each instantiation I we assign an *archetypal result* $I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) \in L_{\mathrm{g}} T_{\mathbf{Nom}} |X|$. This is to be thought of as a simultaneous instantiation of both the conclusion label and of the conclusion target.

First, we consider how to instantiate the conclusion target. By instantiating raw syntax into abstract syntax (as in Section 2.3) and then evaluating explicit substitutions (as in Section 2.2.2), we have a function

$$T^{\mathsf{sub}}_{\mathbf{Set}_{\mathtt{N}}} \mathtt{X} \to \left| T^{\mathsf{sub}}_{\mathbf{NomSub}} X \right| \to |T_{\mathbf{NomSub}} X| \xrightarrow{\sim} T_{\mathbf{Nom}} |X|$$

and we let $I(\underline{\mathtt{tar}}) \in T_{\mathbf{Nom}} |X|$ be the image of this function on the conclusion target $\underline{\mathtt{tar}}$.

[1] Condition (7), though sensible, is not strictly necessary for our main result (Theorem 4.4); rules violating it do not induce any transitions.

Now the archetypal result is dependent on the kind of conclusion label, as follows:

$$\text{for } \underline{\mathtt{l}} = \mathtt{c}(\mathtt{a}),\ I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) = \mathsf{in}(I_{\mathrm{n}}(\mathtt{c}), \langle I_{\mathrm{n}}(\mathtt{a})\rangle I(\underline{\mathtt{tar}}));$$
$$\text{for } \underline{\mathtt{l}} = \bar{\mathtt{c}}\mathtt{d},\ I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) = \mathsf{out}(I_{\mathrm{n}}(\mathtt{c}), I_{\mathrm{n}}(\mathtt{d}), I(\underline{\mathtt{tar}}));$$
$$\text{for } \underline{\mathtt{l}} = \bar{\mathtt{c}}(\mathtt{a}),\ I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) = \mathsf{bout}(I_{\mathrm{n}}(\mathtt{c}), \langle I_{\mathrm{n}}(\mathtt{a})\rangle I(\underline{\mathtt{tar}}));$$
$$\text{for } \underline{\mathtt{l}} = \tau,\ I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) = \mathsf{tau}(I(\underline{\mathtt{tar}})).$$

Abstract rules. The archetypal parameter of an instantiation represents the smallest parameter that should be considered with that instantiation. The same instantiation, however, is also adequate for overspecified parameters; *i.e.* those that more than fulfill the premises. Formally, thus, we say that an instantiation I is *adequate* for a parameter

$$s = \underline{\mathtt{op}}\left(\begin{array}{l}(I_{\mathrm{n}}(\underline{\mathtt{c}}_i))_{i\in[1,\mathrm{ar_n}(\underline{\mathtt{op}})]}\,,\\ \left(\langle I_{\mathrm{n}}(\underline{\mathtt{a}}^j_k)\rangle_{k\in[1,\mathrm{bdep}_{\underline{\mathtt{op}}}(j)]}\left(I_{\mathrm{t}}(\underline{\mathtt{x}}_j), \beta_j\right)\right)_{j\in[1,\mathrm{ar_t}(\underline{\mathtt{op}})]}\end{array}\right)$$

in $\Sigma_{\mathbf{Nom}}\left(|X| \times B_{\mathrm{g}}\,|X|\right)$ if $I(\mathrm{Prem}[j]) \subseteq \beta_j$ for all $j \in \mathrm{ar_t}(\underline{\mathtt{op}})$ and $I_{\mathrm{n}}(\mathsf{bn}(\underline{\mathtt{l}}))\#s$.

Thus for every parameter $s \in \Sigma_{\mathbf{Nom}}\left(|X| \times B_{\mathrm{g}}\,|X|\right)$ we have a set $[\![\mathtt{R}]\!]_X(s) \subseteq L_{\mathrm{g}}T_{\mathbf{Nom}}\,|X|$ of possible results:

$$[\![\mathtt{R}]\!]_X(s) = \{I(\underline{\mathtt{l}}, \underline{\mathtt{tar}}) \mid I \text{ is an adequate instantiation for } s\,\}$$

The collection of adequate instantiations is typically not finite. However, we have the following result.

Lemma 4.2 *The set $[\![\mathtt{R}]\!]_X(s)$ is finite.*

Thus the mapping $s \mapsto [\![\mathtt{R}]\!]_X(s)$ yields a function as follows:

$$[\![\mathtt{R}]\!]_X : \Sigma_{\mathbf{Nom}}\left(|X| \times B_{\mathrm{g}}\,|X|\right) \to B_{\mathrm{g}}T_{\mathbf{Nom}}\,|X| \quad .$$

Theorem 4.3 *The family $\{[\![\mathtt{R}]\!]_X\}_{X\in\mathbf{NomSub}}$ is a natural family of equivariant functions.*

The collection of natural transformations (10) pointwise inherits a join semi-lattice structure from B_{g}, allowing the extension of the theory to finite sets of rules.

Theorem 4.4 *For a name-passing system defined by a finite set of rule structures for a binding signature Σ, each satisfying Conditions (1–9), wide-open bisimilarity on the ground transition system induced by the syntactic operational model $T_{\mathbf{Nom}}\emptyset \to B_{\mathrm{g}}T_{\mathbf{Nom}}\emptyset$ is a congruence.*

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. POPL'01*, 2001.

[2] L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra*, chapter 1.3, pages 197–292. Elsevier, 1999.

[3] K. L. Bernstein. A congruence theorem for structural operational semantics of higher-order languages. In *Proc. LICS'98*, pages 153–164, 1998.

[4] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[5] G. L. Cattani and P. Sewell. Models for name-passing processes: interleaving and causal. *Inform. and Comput.*, 190(2):136–178, 2004.

[6] M. P. Fiore, E. Moggi, and D. Sangiorgi. A fully abstract model for the π-calculus. *Inform. and Comput.*, 179(1):76–117, 2002.

[7] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proc. LICS'99*, pages 193–202, 1999.

[8] M. P. Fiore and S. Staton. Comparing operational models of name-passing process calculi. *Inform. and Comput.*, 204(4):524–560, 2006.

[9] M. P. Fiore and D. Turi. Semantics of name and value passing. In *Proc. LICS'01*, pages 93–104, 2001.

[10] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. *Formal Aspect. Comput.*, 13(3–5):341–363, 2002.

[11] B. Klin. From bialgebraic semantics to congruence formats. In *Proc. SOS'04*, ENTCS, pages 3–37, 2005.

[12] D. Miller and A. Tiu. A proof theory for generic judgments. *ACM Trans. Comput. Logic*, 6(4):749–783, 2005.

[13] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (II). *Inform. and Comput.*, 100(1):41–77, 1992.

[14] J. Parrow and B. Victor. The Update calculus. In *Proc. AMAST'97*, volume 1349 of *LNCS*, 1997.

[15] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Inform. and Comput.*, 186(2):165–193, 2003.

[16] D. Sangiorgi. A theory of bisimulation for the pi-calculus. *Acta Inform.*, 33(1):69–97, 1996.

[17] D. Sangiorgi. Lazy functions and mobile processes. In *Essays in honour of Robin Milner*. MIT Press, 2000.

[18] I. Stark. A fully abstract domain model for the π-calculus. In *Proc. LICS'96*, pages 36–42, 1996.

[19] R. Street. The formal theory of monads. *J. Pure Appl. Algebra*, 2:149–168, 1972.

[20] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291, 1997.

[21] S. Weber and B. Bloom. Metatheory of the π-calculus. Technical Report TR96-1564, Cornell University, 1996.

[22] A. Ziegler, D. Miller, and C. Palamidessi. A congruence format for name-passing calculi. In *Proc. SOS'05*, 2005.