General Structural Operational Semantics through Categorical Logic

(Extended Abstract)

Sam Staton
Computer Laboratory, University of Cambridge

Abstract

Certain principles are fundamental to operational semantics, regardless of the languages or idioms involved. Such principles include rule-based definitions and proof techniques for congruence results. We formulate these principles in the general context of categorical logic. From this general formulation we recover precise results for particular language idioms by interpreting the logic in particular categories. For instance, results for first-order calculi, such as CCS, arise from considering the general results in the category of sets. Results for languages involving substitution and name generation, such as the π-calculus, arise from considering the general results in categories of sheaves and group actions. As an extended example, we develop a tyft/tyxt-like rule format for open bisimulation in the π-calculus.

1. Introduction

Questions about structural operational semantics. In this paper we are concerned with the semantics of programming languages in the structural style introduced by Plotkin [38]. One can reason in this style by induction on the language syntax, in terms of the steps that programs may make. Thus a structural operational semantics is given by a transition system specification involving a rule-based inductive definition.

When working in this style, it is often infeasible to commence reasoning about particular programs before basic lemmas have been established about the whole language. One must prove, firstly, that the transition system specification indeed gives rise to a transition system, and, secondly, that compositional reasoning is permitted. The second lemma usually involves a proof that bisimilarity, a natural equivalence on programs, is a congruence.

One must reprove these basic results for every language that is encountered. For many languages, though, the definition principles and proof outlines have a lot in common. Thus the following questions arise. Practically speaking, *is there some general machinery that could save work when specifying semantics and proving basic lemmas?* More philosophically, *is there a precise sense in which the proofs 'have a lot in common'?* Perhaps structural operational semantics is too permissive. *Is there a general concept of 'good' structural operational semantics*?

We now summarize two approaches to answering these questions. Rule formats are concrete, but, we argue, not general enough. Functorial Operational Semantics is general, but we argue that it is obscured by abstraction. In the present paper we bring the two approaches together, providing rule formats in a general categorical setting.

Concrete answers: rule formats. In a first attempt to answer the questions above, de Simone [11] introduced a rule format: a list of syntactic conditions about rules, stipulating which variables and operators may appear in the various parts of the rule. Any language specified according to this rule format will have, among other things, the property that bisimilarity is a congruence. Further work on more elaborate rule formats has continued since then (see surveys [3; 36]). In this paper we focus on the tyft/tyxt format [20].

We argue that the questions above are not entirely answered by this research programme. Most rule formats account for languages like CCS [33] and various extensions thereof, but many languages are not simple enough to fit into these formats. Consider the π-calculus [34] — a relatively simple language by some standards. The syntax involves variable binding, as in the λ-calculus, so there are concerns about α-equivalence and capture-avoiding substitution. There are various definitions of bisimulation, some involving substitution and quantification of fresh names.

What kind of structure is a proof for a π-calculus transition? Which kinds of bisimilarity are congruences? The formalization of these kinds of concerns has been a matter of several recent investigations [e.g. 7; 15; 16; 21; 32; 43], including rule formats for the π-calculus [e.g. 9; 12; 45].

Putting these complications aside, though, the specifications of CCS and the π-calculus are not dissimilar, and the proofs of congruence of bisimilarity for the two languages do have a lot in common. To properly answer the above questions, and to have a chance at tackling increasingly sophisticated languages, we must identify the commonality.

Abstract answers: Functorial Operational Semantics. Functorial Operational Semantics [44] provides answers to the questions at an abstract level. Good operational semantics is seen categorically as a distributive law of a monad over a comonad, or equivalently as a lifting of a monad to a category of coalgebras. The initial work [44] illustrated the general framework with congruence results for CCS-like languages, but subsequent work demonstrated the ideas in various different settings (see [28] for a survey).

A potential criticism of much of this work is that it is too abstract. One must really squint hard to view a distributive law as a collection of rules, and to view the naturality conditions of a distributive law as the conditions imposed by a concrete rule format. For every new language idiom, these naturality constraints have to be translated into a new rule format, and this is often an intricate and difficult task.

Methodology. We advocate the following general methodology for answering the questions. First, the results and techniques must be understood at a concrete, syntactic level, for a particular language, or language idiom. In this paper, we consider the tyft/tyxt rule format.

The second step is to find a categorical model theory within which this concrete work can be understood. For CCS-like languages, this model theory would involve sets, free algebras, and transition systems.

The third step is to reconsider the concrete results in the setting of the categorical model theory. In this development, it becomes clear that some aspects of the model theory are not necessary for the results. Here, we find that we are able to formulate results about the tyft/tyxt format in any ΠW-pretopos. Such categories arise as models of first-order logic, dependent type theory, and well-founded induction [35; 18; 5]. From a logical perspective, then, we develop a meta-theory for tyft/tyxt specifications in a constructive logic.

Finally, once the concrete results are understood in general, categorical terms, the connections with Functorial Operational Semantics can be clarified.

To evaluate the generality of the work, we change the model theory, and work in a different ΠW-pretopos from the category of sets. By considering the categorical results in a category of 'nominal substitutions', we arrive at results that are suitable for the idioms of the π-calculus. Considering the categorical results in a more elaborate presheaf category gives an analysis of a more elaborate notion of bisimulation. In the course of this evaluation we discover further generalizations at the categorical level, for instance working with monoidal rather than cartesian structure.

A matter for current and future work is the study of increasingly sophisticated language idioms, such as encryption in applied π-calculi [e.g. 1; 2; 8]. One must first find appropriately sophisticated model categories, and the categorical results of this paper can then be considered in them.

Synopsis and summary. We recall the properties of ΠW-pretoposes and W-types in Sec. 2. In this setting, we introduce a categorical version of the tyft/tyxt rule format in Sec. 3.

The remainder of the paper, Secs. 4, 5, and 6, can be read in any order. In Sec. 4, inspired by Functorial Operational Semantics, we consider the congruence and conservativity results for tyft/tyxt in a more general setting.

The illustrations in Sec. 3 involve constructions from CCS, in the category of sets. In Secs. 5 and 6 we apply the categorical rule format to more sophisticated calculi by working in more elaborate categories. In Sec. 5 we study the categorical format in a category of nominal substitutions, achieving congruence results for a simplification of open bisimilarity [39]. In Sec. 6, we work with (genuine) open bisimilarity by studying the format in a category of presheaves. The resulting format is essentially that of [45].

Acknowledgements. Many thanks to Marcelo Fiore for his guidance. It was helpful to be able to discuss Prop. 6 with Thorsten Altenkirch, Nicola Gambino and Thomas Streicher, and to discuss [45] with its authors. Andy Pitts first suggested investigating the tyft/tyxt format. Thanks also to the anonymous reviewers. Financial support has been provided by EPSRC grants GR/T22049/01 and EP/E042414/1.

2. Categorical logic

ΠW-pretoposes are categories that arise as of models of first-order logic, dependent type theory, and also well-founded induction, through W-types. This is the right structure for investigating rule-based inductive definitions, as is shown in Sec. 3. In this section, we survey the rudiments of ΠW-pretoposes. A standard text for categorical logic is [23, Part D].

2.1. Locally cartesian closed pretoposes

A pretopos is a regular category with effective equivalence relations and stable disjoint coproducts. A locally cartesian closed pretopos, then, is both a Heyting pretopos, and hence a model for intuitionistic first order logic, and also a locally cartesian closed (lcc) category, and hence a model for dependent type theory [see e.g. 6; 29; 37].

Note that every topos, including the category of sets, is a lcc pretopos. There is no harm in restricting attention to toposes, and indeed the pretoposes that we consider in this paper are all toposes (assuming a classical treatment of the underlying theory of sets). In the categories of nominal substitutions (Sec. 5) and presheaves (Sec. 6), though, the powerobjects are clumsy to describe explicitly, and it is easiest to work directly with relations instead.

Notation. For a morphism $f : X \to Y$ in a lcc category, we have adjoint functors $\amalg_f \dashv \Delta_f \dashv \Pi_f : \mathcal{C}/_X \to \mathcal{C}/_Y$, where $\Delta_f : \mathcal{C}/_Y \to \mathcal{C}/_X$ is pullback along f. In an lcc pretopos, we also have adjoint functors $\exists_f \dashv f^* \dashv \forall_f : \mathrm{Sub}_\mathcal{C}(X) \to \mathrm{Sub}_\mathcal{C}(Y)$, when restricting to the subobjects.

Technicality: internal projectivity. Recall that in a lcc category $\mathcal{C}$, a morphism $f : X \to Y$ is said to be *internally projective* if the product functor preserves epimorphisms. In a lcc pretopos, internal projectivity gives rise to a tighter relationship between the first-order and dependent-type structure. For instance, consider morphisms $X \xrightarrow{f} Y \xrightarrow{g} Z$. If g is internally projective then, writing $Im_X : \mathcal{C}/_X \to \mathrm{Sub}_\mathcal{C}(X)$ for the image functor [e.g. 6],

$$\forall_g \circ \exists_f \circ Im_X \;=\; Im_Z \circ \Pi_g \circ \amalg_f \quad : \mathcal{C}/_X \to \mathrm{Sub}_\mathcal{C}(Z) \quad .$$

Note the following properties of internally projectives in an lcc pretopos [e.g. 24, Sec. IV.5]: they are closed under composition and coproducts; they are stable under pullback; the canonical morphisms $0 \to 1$, $1 \to 1$, and $1+1 \to 1$ are internally projective; if gf is internally projective, then so is f. The 'internal axiom of choice' is the statement that every morphism is internally projective.

2.2. W-types, free algebras, and ΠW-pretoposes

An important aspect of Martin-Löf's dependent type theory is W-types, which describe well-founded induction and recursion [30]. W-types are central to our development, and we now summarize the important ideas in the categorical setting, following [35; 18; 5].

Signatures in categories, and their algebras.

Definition 1 *A signature in a category is a morphism* $\mathrm{ar}_\Sigma : \mathrm{Ar}_\Sigma \to \Sigma$.

In such a signature, Σ is to be thought of as the 'object of operators', and Ar_Σ as 'the object of arities'. An operator is a generalized point of Σ, i.e. an object I together with a morphism $\sigma : I \to \Sigma$. The arity of such an operator found by pulling it back along $\mathrm{ar}_\Sigma : \mathrm{Ar}_\Sigma \to \Sigma$.

For an example from process algebra, we fix a set of actions A, and define a signature in the category $\mathbf{Set}$ of sets, for a variant of CCS [33]. It has operators $\Sigma_{\mathrm{CCS}} = \mathit{nil} : 1 + \mathit{par} : 1 + \mathit{act} : A + \overline{\mathit{act}} : A$, and arities $\mathrm{Ar}_{\Sigma_{\mathrm{CCS}}} = \mathit{nil} : \varnothing + \mathit{par} : 2 + \mathit{act} : A + \overline{\mathit{act}} : A$. (Here we are labelling the components of the coproducts.) The function $\mathrm{ar}_{\Sigma_{\mathrm{CCS}}} : \mathrm{Ar}_{\Sigma_{\mathrm{CCS}}} \to \Sigma_{\mathrm{CCS}}$ is the evident one. This signature provides, for instance, a binary parallel operator *par*, and a unary prefix operator act_a for each $a \in A$.

Any signature $(\mathrm{Ar}_\Sigma \xrightarrow{\mathrm{ar}_\Sigma} \Sigma)$ in a lcc category $\mathcal{C}$ gives rise to an endofunctor on $\mathcal{C}$:

$$\Sigma(-) \;=\; \mathcal{C} \xrightarrow{\Delta_!} \mathcal{C}/_{\mathrm{Ar}_\Sigma} \xrightarrow{\Pi_{\mathrm{ar}_\Sigma}} \mathcal{C}/_\Sigma \xrightarrow{\amalg_!} \mathcal{C} \quad .$$

An algebra for a signature $(\mathrm{Ar}_\Sigma \to \Sigma)$ is an algebra for the corresponding endofunctor. For signatures in $\mathbf{Set}$, such as Σ_{CCS} above, these algebras are same as the algebras for signatures studied in universal algebra. The free Σ_{CCS}-algebra is the set of all CCS terms.

W-types, ΠW-pretoposes, and free monads. A lcc category $\mathcal{C}$ is said to be have *W-types* if there is a free algebra W_Σ for every signature Σ. A lcc pretopos with W-types is called a Π*W-pretopos*. Note that every topos with a natural numbers object is a ΠW-pretopos [35].

In the setting of ΠW-pretoposes, the category of Σ-algebras is monadic over $\mathcal{C}$. We write W_Σ for the corresponding monad, the "free monad on Σ".

Multi-sorted signatures. It is useful to consider algebras for multi-sorted signatures.

Definition 2 *A* sorting *for a signature* $\mathrm{ar}_\Sigma : \mathrm{Ar}_\Sigma \to \Sigma$ *is an object S with a pair of morphisms $s_\Sigma : \Sigma \to S$, $s_{\mathrm{Ar}_\Sigma} : \mathrm{Ar}_\Sigma \to S$. (No diagrams are required to commute.)*

Every S-sorted signature in a ΠW-pretopos $\mathcal{C}$ gives rise to an endofunctor on $\mathcal{C}/_S$:

$$\Sigma(-) \;=\; \mathcal{C}/_S \xrightarrow{\Delta_{s_{\mathrm{Ar}_\Sigma}}} \mathcal{C}/_{\mathrm{Ar}_\Sigma} \xrightarrow{\Pi_{\mathrm{ar}_\Sigma}} \mathcal{C}/_\Sigma \xrightarrow{\amalg_{s_\Sigma}} \mathcal{C}/_S .$$

The category of S-sorted Σ-algebras, viz. algebras for this endofunctor, is monadic over $\mathcal{C}/_S$ [18].

3. Operational semantics with categorical logic

The tyft/tyxt format has been proposed as a format for well-behaved rule-based inductive definitions [20]. In this section we reformulate the tyft/tyxt requirements in an arbitrary ΠW-pretopos, and state the congruence results in this setting.

Concrete version. Before moving to the categorical setting, we recall the tyft/tyxt rule format from [20]. We say a rule is in *pre-tyft format* if it is of the form

$$\frac{t_1 \xrightarrow{l_1} w_1 \quad t_2 \xrightarrow{l_2} w_2 \quad \ldots}{op(v_1, \ldots, v_n) \xrightarrow{l} t}$$

where: *op* is an operator in the signature; the v_is and w_is are all variables, possibly the same; and t and the t_is are all terms, possibly involving free variables. If a rule has a single variable in the left-hand side of the conclusion, in place of an operator, it is in the *pre-tyxt* format.

A pre-tyft/tyxt rule is a *tyft/tyxt* rule if the variables on the left-hand side of the conclusion and the right-hand sides of the premises are all different from each other.

A *(pre-)tyft/tyxt transition system specification* (TSS) is given by an algebraic signature (for the syntax), a set of labels, and a set of (pre-)tyft/tyxt rules.

3.1. tyft/tyxt TSSs in a ΠW-pretopos

We now explain how transition system specifications in the tyft/tyxt format can be understood as structures in a ΠW-pretopos. This can be seen as a reformulation of the introduction to this section in dependent type theory.

Labels as operators. A pre-tyft/tyxt TSS is used to specify a labelled transition system. To be precise, let L be a set of labels, and recall that a labelled transition system is a set X equipped with a relation $(\rightsquigarrow_X) \subseteq X \times L \times X$. Infix notation is common, writing $x \overset{l}{\rightsquigarrow}_X x'$ to mean "x evolves to x', through l".

Here, we will generalize, considering the labels as a collection of operators that may take arbitrary arities. Following the developments of Sec. 2, we have a signature L of labels giving rise to an endofunctor $L(-)$ An L-labelled transition system, then, is an object X together with a relation $(\rightsquigarrow_X) \subseteq X \times L(X)$.

This definition subsumes the conventional one, where labels are to be thought of as being unary: in that situation one might use the notation $x \rightsquigarrow_X l(x')$. On the other hand, a nullary label can be thought of as a predicate on states. (More complex arities are crucial in Sec. 5.)

Categorical pre-tyft/tyxt.

Definition 3 *A* categorical pre-tyft/tyxt TSS *in a* ΠW*-pretopos is given by the following data. Notation is explained below.*

1. *Signatures* $\mathrm{Ar}_\Sigma \to \Sigma$, $\mathrm{Ar}_L \to L$ *for syntax and for labels, respectively.*
2. *An object R of all the rules in the TSS; an object P of all the premises in the TSS; and an object V of all the variables in the TSS.*
3. *A morphism $V \to R$, assigning to each variable the rule in which it appears.*
4. *A morphism $\rho : P \to R$, assigning to each premise the rule for which it is a premise.*
5. *A morphism* $(p : P) \to \mathrm{W}_\Sigma(V[\rho(p)])$*, assigning to each premise the term appearing in its left-hand side. (The free variables in that term must be variables for that rule.)*
6. *A morphism $P \to L$, assigning to each premise the label of its right-hand side.*
7. *A morphism $P \times_L \mathrm{Ar}_L \to V$ over R, assigning a variable to each parameter of the right-hand side of each premise.*
8. *A morphism $R \to (\Sigma + 1)$. If a rule maps into 1, then it is thought of as a tyxt rule; otherwise it is a tyft rule and it is assigned to the operator on the left-hand side of the conclusion.*
9. *A morphism* $(r : R) \to L(\mathrm{W}_\Sigma(V[r]))$*, assigning to each rule the labelled term appearing on the right-hand side of its conclusion.*
10. *A morphism $R \times_{\Sigma+1} (\mathrm{Ar}_\Sigma + 1) \to V$ over R, assigning a variable to each parameter of the left-hand side of each conclusion.*

Notation. In item (7), by 'a morphism over R', is meant a morphism in $\mathcal{C}/_R$, when the domain and codomain are considered as objects of $\mathcal{C}/_R$ as a result of items (4) and (3) respectively. Terminology in item (10) is analogous.

A signature $\mathrm{ar}_\Sigma : \mathrm{Ar}_\Sigma \to \Sigma$ in $\mathcal{C}$ gives rise to a signature $(R \times \mathrm{ar}_\Sigma) : R \times \mathrm{Ar}_\Sigma \to R \times \Sigma$ in $\mathcal{C}/_R$. The free monad $\mathrm{W}_{R\times\Sigma}$ on this signature can be thought of as a localization of W_Σ to $\mathcal{C}/_R$. The notation in item (5) means that a morphism is given in $\mathcal{C}/_R$:

$$\begin{tikzcd} P \arrow[rr] \arrow[dr, "\rho"'] & & \mathrm{W}_{R\times\Sigma}(V) \arrow[dl] \\ & R & \end{tikzcd}$$

The notation in item (9) is analogous.

Categorical tyft/tyxt. A categorical pre-tyft/tyxt TSS is a *categorical tyft/tyxt TSS* if the morphism

$$(R \times_{\Sigma+1} (\mathrm{Ar}_\Sigma + 1)) + (P \times_L \mathrm{Ar}_L) \to V$$

is a complemented mono. That is, there is an object $\bar{V}$ of "variables that don't appear in the left-hand side of the conclusion or the right-hand side of premises", and

$$V \cong \bar{V} + (R \times_{\Sigma+1} (\mathrm{Ar}_\Sigma + 1)) + (P \times_L \mathrm{Ar}_L).$$

Well-founded and pure tyft/tyxt. Two additional properties of tyft/tyxt rules are important. We reformulate notions from [20].

A categorical pre-tyft/tyxt TSS is *well-founded* if there is a morphism $V \to \mathbb{N}$, assigning a 'depth' to each variable, which is such that all the variables on the right-hand side of a premise are of higher depth than all the variables on its left-hand side.

A well-founded tyft/tyxt TSS is said to be *pure* if all the variables that appear in the rules appear either on the left-hand side of a conclusion, or on the right-hand side of a premise. Formally, a pure categorical TSS is one for which the morphism $(R \times_{\Sigma+1} (\mathrm{Ar}_\Sigma + 1)) + (P \times_L \mathrm{Ar}_L) \to V$ is an isomorphism.

3.2. The meaning of TSSs, as proof trees

One way to describe the meaning of a TSS is to formally define its proof trees. We will consider proof trees for a categorical pre-tyft/tyxt TSS as syntax for a multi-sorted signature. The operators in this signature are the nodes of the proof tree: they are instances of rules, i.e., rules paired with valuations of the variables. The parameters taken by a rule instance, considered as an operator, are the premises of the rule.

The sorting discipline ensures that the premises assigned to a rule indeed prove the appropriate transitions. The sorts for the signature are pairs $\mathrm{W}_\Sigma \times L(\mathrm{W}_\Sigma)$, i.e. they are possible transitions. The sort of a rule instance, considered as an operator, is the transition that it proves. The sorts of its parameters, i.e. the premises of the rule, are the transitions that those premises must prove.

Formally, the signature PT for proof trees is the projection $\coprod_{r:R}((\mathrm{W}_\Sigma)^{V[r]} \times P[r]) \to \coprod_{r:R}(\mathrm{W}_\Sigma)^{V[r]}$. The sorting for the 'arities' is given by the composite

$$\begin{aligned}&\coprod_{r:R}\Big((\mathrm{W}_\Sigma)^{V[r]} \times P[r]\Big)\\ &\quad\xrightarrow{(5,6,7)} \coprod_{r:R}\Big((\mathrm{W}_\Sigma)^{V[r]} \times \mathrm{W}_\Sigma(V[r]) \times L(V[r])\Big)\\ &\quad\xrightarrow{\text{str, eval}} \mathrm{W}_\Sigma \times L(\mathrm{W}_\Sigma).\end{aligned}$$

The first morphism is derived from data (5,6,7). The second one uses the evaluation of the exponential together with the strengths of W_Σ and L [18, Prop. 6, 20]. The sorting for the operators is derived from data (8,9,10) in an analogous manner.

Because the ambient category has W-types, this multi-sorted signature has a free algebra, W_{PT}, which is the object of proof trees. The sorting for the free algebra gives a map $\mathrm{W}_{\mathrm{PT}} \to \mathrm{W}_\Sigma \times L(\mathrm{W}_\Sigma)$, and the image of this map is the labelled transition relation induced by the TSS.

3.3. Illustrations from CCS

We considered a signature for the syntax of CCS in Sec. 2.2. We now explain how the TSS for CCS from [33] can be seen as a categorical tyft TSS in the sense of Def. 3.

Consider, for example, the communication rule in CCS.

$$\frac{v \xrightarrow{a} v' \quad w \xrightarrow{\bar{a}} w'}{v|w \xrightarrow{\tau} v'|w'}$$

This should be viewed as a family of rules, one for each action $a \in A$. Removing the syntactic sugar, we can rewrite the rules as follows.

$$\textit{rule-com}_a : \quad \frac{v_a \rightsquigarrow a : v'_a \quad w_a \rightsquigarrow \bar{a} : w'_a}{\textit{par}(v_a, w_a) \rightsquigarrow \tau : \textit{par}(v'_a, w'_a)} \qquad (1)$$

It is now a matter of transcribing these rules into the style of Def. 3. There is one rule, *rule-com*$_a$, for each action a, so $R \cong A$. There are two premises for each rule: $P \cong 2 \times R$. There are four variables, v_a, v'_a, w_a, w'_a in each rule *rule-com*$_a$, so $V \cong 4 \times R$. The remaining data for Def. 3 is extracted straightforwardly from (1). For example, for item (5) we need a function $P \to L$. This function assigns the label a to the premise $(1, \textit{rule-com}_a)$, and the label $\bar{a}$ to the premise $(2, \textit{rule-com}_a)$.

Proof trees. An example proof of a transition for CCS is the following.

$$\frac{\dfrac{}{a.\bar{a}.0 \xrightarrow{a} \bar{a}.0} \quad \dfrac{}{\bar{a}.a.0 \xrightarrow{\bar{a}} a.0}}{a.\bar{a}.0 \,|\, \bar{a}.a.0 \xrightarrow{\tau} \bar{a}.0 \,|\, a.0} \,(\textit{rule-com}_a)$$

This arises as an element of the set of proof trees, W_{PT}. The outermost 'operator' is the root of the tree; it is given by the rule name *rule-com*$_a$ together with the function

$$v_a \mapsto a.\bar{a}.0 \qquad w_a \mapsto \bar{a}.a.0 \qquad v'_a \mapsto \bar{a}.0 \qquad w'_a \mapsto a.0$$

assigning terms to the variables of the rule. This operator takes two parameters, describing the left and right branches of the tree.

3.4 Congruence of bisimilarity

In this subsection we state the congruence result, and a conservativity result, for categorical tyft/tyxt. To begin, we define notions of congruence and bisimilarity in this setting.

Congruence.

Definition 4 *For any signature* Σ*, a relation* $R \subseteq \mathrm{W}_\Sigma \times \mathrm{W}_\Sigma$ *is a* congruence *if it satisfies the following formula in the internal logic of the pretopos.*

$$\begin{aligned}&\forall \sigma : \Sigma,\ \vec{t}, \vec{t'} : (\mathrm{W}_\Sigma)^{\mathrm{Ar}_\Sigma[\sigma]}.\\ &\qquad \forall n : \mathrm{Ar}_\Sigma[\sigma].\ t_n \, R \, t'_n \implies \sigma(\vec{t}) \, R \, \sigma(\vec{t'})\end{aligned}$$

(We are perhaps breaking with tradition by not requiring congruences to be equivalence relations.)

Bisimulation. We define bisimulation for the general case of L-labelled transition systems, by rephrasing the usual notion in the present setting.

Definition 5 *Consider two L-labelled transition systems, $(X, \rightsquigarrow_X)$ and $(Y, \rightsquigarrow_Y)$. A relation $R \subseteq X \times Y$ is a* simulation *if it satisfies the formula*

$$\forall x : X,\ y : Y,\ l : L,\ \vec{x} : X^{\mathrm{Ar}_L[l]}.\ x \rightsquigarrow_X l(\vec{x}) \wedge x\, R\, y$$
$$\implies \exists \vec{y} : Y^{\mathrm{Ar}_L[l]}.\ y \rightsquigarrow_Y l(\vec{y}) \wedge \forall n : \mathrm{Ar}_L[l].\, x_n\, R\, y_n$$

in the internal logic. A simulation is a bisimulation *if the opposite $R^\circ \subseteq Y \times X$ is also a simulation.*

We say that an L-labelled transition system $(X, \rightsquigarrow_X)$ is *projectively branching* if the projection map $(\rightsquigarrow_X) \to X$ is internally projective.

Proposition 6 *Let L be a signature in a ΠW-pretopos. Between every pair of projectively branching L-labelled transition systems, there is a greatest simulation (called* similarity*), and a greatest bisimulation (called* bisimilarity*).*

The requirement of internal projectivity is not needed if the ambient ΠW-pretopos is a topos, as is the case in all our examples. In this setting, the result can be proved using Tarski's fixed point theorem.

Congruence theorem.

Theorem 7 *Consider a pure well-founded categorical tyft/tyxt TSS with an internally projective premises map, $P \to R$. For the initial labelled transition system that is induced, the similarity and bisimilarity are both congruences.*

We demonstrate why the internal-projectivity requirement is necessary in Sec. 5.2. It is implicit in [20], because the axiom of choice is assumed.

Combining TSSs and conservativity. Categorical TSSs can be combined in a straightforward way, by taking coproducts of the corresponding objects and morphisms. We thus have a generalization of an elementary conservativity property of tyft TSSs [20, Thm. 7.6; 3, Sec. 4].

Theorem 8 *If a pure, well-founded tyft/tyxt TSS with an internally projective premise map ($P \to R$) is combined with a tyft TSS with no common operators, then every term in the first TSS is bisimilar to the same term considered in the combined TSS.*

4. Good semantics and monad morphisms

In Sec. 3, we reformulated the tyft/tyxt congruence format in the setting of ΠW-pretoposes. We now investigate operational semantics and congruence results in a even more general setting. We state general definitions and results in the setting of categories of monads.

A category of monads. We follow [42] in defining the category $\mathbf{Mnd}$ of monads as follows. The objects are pairs $(\mathcal{C}, T)$ of a category $\mathcal{C}$ and a monad T; a morphism $(F, \phi) : (\mathcal{C}, T) \to (\mathcal{D}, S)$ is a functor $F : \mathcal{C} \to \mathcal{D}$ together with a natural transformation $\phi : SF \to FT$ that respects the monad unit and multiplication.

A *lifting* of a monad S on a category $\mathcal{D}$ along a functor $F : \mathcal{C} \to \mathcal{D}$ is given by a monad T on $\mathcal{C}$ together with an isomorphism ϕ, making (F, ϕ) a monad morphism $(\mathcal{C}, T) \to (\mathcal{D}, S)$.

Monad liftings and operational semantics. Let $\mathcal{S}ystems$ be a category, thought of as a category of models, and let $\mathcal{S}tates$ be another category, whose objects are thought of as state spaces. Suppose that there is a 'forgetful' functor $\mathcal{S}ystems \to \mathcal{S}tates$, taking a model to its state space. In this section we develop the following thesis: *a good semantics defines a lifting of a monad on $\mathcal{S}tates$ along the forgetful functor $\mathcal{S}ystems \to \mathcal{S}tates$.*

(The Functorial Operational Semantics of [44] is recovered when $\mathcal{S}ystems$ is a category of coalgebras for a weak-pullback-preserving comonad.)

4.1. Monad liftings induced by tyft/tyxt TSSs

We begin by showing that every categorical pre-tyft/tyxt TSS (Def. 3) for a signature Σ induces a lifting of the monad W_Σ to a category of labelled transition systems.

Categories of transition systems. Recall that an L-labelled transition system, for a signature L in a ΠW-pretopos $\mathcal{C}$, is a pair $(X, \rightsquigarrow_X)$ of an object X and a relation $(\rightsquigarrow_X) \subseteq X \times L(X)$. A morphism of L-labelled transition systems, $(X, \rightsquigarrow_X) \to (Y, \rightsquigarrow_Y)$, is a morphism $X \to Y$ in $\mathcal{C}$ for which there exists a morphism $(\rightsquigarrow_X) \to (\rightsquigarrow_Y)$ making the evident diagram commute. We write L-$\mathbf{LTS}$ for the resulting category.

Monad liftings. Let Σ and L be a signatures on a ΠW-pretopos $\mathcal{C}$. A categorical pre-tyft/tyxt TSS for Σ and L gives rise to a lifting $\overline{\mathrm{W}_\Sigma}$ of the free monad W_Σ along the forgetful functor L-$\mathbf{LTS} \to \mathcal{C}$, as we now explain. The lifted monad takes an L-labelled transition system, $(X, \rightsquigarrow_X)$ to a new transition system with carrier $\mathrm{W}_\Sigma(X)$ and whose relation is found by augmenting the ambient TSS. We add X as constant operators of the signature, and we introduce a rule with no premises for each element of $(\rightsquigarrow_X)$. Formally, then, the signature $\mathrm{PT}_{(X,\rightsquigarrow_X)}$ for proof trees for this augmented TSS is as follows.

$$0 + \coprod_{r:R} (\mathrm{W}_\Sigma(X))^{V[r]} \times P[r]) \to (\rightsquigarrow_X) + \coprod_{r:R} ((\mathrm{W}_\Sigma(X))^{V[r]})$$

The sorts for this signature are $\mathrm{W}_\Sigma(X) \times L(\mathrm{W}_\Sigma(X))$, and they are assigned analogously to Sec. 3.2. We thus form an object $\mathrm{W}_{\mathrm{PT}_{(X,\rightsquigarrow_X)}}$ of proof trees for this augmented TSS. The image of the sorting projection $\mathrm{W}_{\mathrm{PT}_{(X,\rightsquigarrow_X)}} \to \mathrm{W}_\Sigma(X) \times L(\mathrm{W}_\Sigma(X))$ gives the transition system component of $\overline{\mathrm{W}_\Sigma}(X, \rightsquigarrow_X)$.

The forgetful functor L-$\mathbf{LTS} \to \mathcal{C}$ is faithful, and so the action of the lifted monad on objects, thus defined, is enough data to define the entire lifted monad. All that remains is to verify that the functorial action, the unit and the multiplication of W_Σ on $\mathcal{C}$ lift to a functorial action, unit and multiplication for $\overline{\mathrm{W}_\Sigma}$ on L-$\mathbf{LTS}$. This follows directly from the universal properties.

Example. TSS for CCS. Our simple example is the TSS for CCS. In the tradition of first-order logic, we can understand the TSS as a first-order theory $\mathbb{T}_{\mathrm{CCS}}$ with a relation symbol for each label and an operator symbol for every operator in the signature Σ_{CCS}. The axioms of this theory are the rules in the specification.

In model theoretic terminology, a *structure* for $\mathbb{T}_{\mathrm{CCS}}$ is a labelled transition system together with a Σ_{CCS}-algebra. A structure is a *model* if the interpretations of the rules hold.

The tyft/tyxt TSS for CCS gives rise to a lifting of the monad $\mathrm{W}_{\Sigma_{\mathrm{ccs}}}$ along the forgetful functor L-$\mathbf{LTS} \to \mathbf{Set}$. Every algebra for the lifted monad contains a structure for the theory $\mathbb{T}_{\mathrm{CCS}}$: the carrier of the algebra is a labelled transition system, and the forgetful functor induced by the lifting morphism associates every algebra with a Σ_{CCS}-algebra. In fact, the algebras for this lifted monad are precisely the models of $\mathbb{T}_{\mathrm{CCS}}$. Moreover, morphisms of algebras are the same thing as morphisms of models. The theory $\mathbb{T}_{\mathrm{CCS}}$ has a free model, the usual semantics of CCS, which is the free algebra on the empty transition system, as in Sec. 3.3.

4.2. General congruence results

We now move back to the general setting of categories of monads. We define notions of congruence and bisimilarity, and establish congruence results in this generality, with illustrations building on Sec. 3.

Congruence.

Definition 9 *Let T be a monad (on an arbitrary category), and let (X, x) and (Y, y) be T-algebras. We say that a span $X \leftarrow R \to Y$ is a* T-congruence *if there is a T-algebra structure $T(R) \to R$ such that the morphisms in the span define algebra homomorphisms.*

For instance, let Σ be a signature in a ΠW-pretopos. The requirement of congruence for a relation on W_Σ is that of Def. 4.

Bisimulation from open maps. To define bisimulation in a general setting, we consider the following situation, generalizing the work of [4; 26]. In a category $\mathcal{S}ystems$, we consider a class of 'open' morphisms. There are various properties that might be expected of this class. We do not require anything of the morphisms for now, but the following axioms (from [25; 24]) will be referred to in this development.

- A1. Isomorphisms are open, and open maps are closed under composition.
- A3 (Descent). In a pullback square

$$\begin{tikzcd} Y' \arrow[r] \arrow[d, "f'"'] & Y \arrow[d, "f"] \\ X' \arrow[r, two heads, "g"'] & Y \end{tikzcd}$$

  if g is epi and f' is open, then f is also open.
- A5. An I-indexed coproduct of open maps is again open.

Definition 10 *Consider a functor $U : \mathcal{S}ystems \to \mathcal{S}tates$, and a class of 'open' morphisms in $\mathcal{S}ystems$. A* bisimulation *between two 'models', $M, N \in \mathcal{S}ystems$, is a span $U(M) \leftarrow R \to U(N)$ in $\mathcal{S}tates$ for which there is a span $M \leftarrow \bullet \to N$ of open morphisms in $\mathcal{S}ystems$, whose U-image is R. Simulation is defined analogously.*

Open morphisms of transition systems. The general definition of bisimulation specializes to the definition for L-labelled transition systems (Def. 5), as follows. We say that a morphism $f : (X, \rightsquigarrow_X) \to (Y, \rightsquigarrow_Y)$ between L-labelled transition systems is *open* if the canonical morphism (dotted) is epic.

$$\begin{tikzcd} (\rightsquigarrow_X) \arrow[dr, dotted] \arrow[drr, bend left] \arrow[ddr, bend right] & & \\ & \Delta_f(\rightsquigarrow_Y) \arrow[r] \arrow[d] & (\rightsquigarrow_Y) \arrow[d] \\ & X \arrow[r, "f"] & Y \end{tikzcd}$$

Given two L-labelled transition systems, $(X, \rightsquigarrow_X)$, $(Y, \rightsquigarrow_Y)$, and a relation between X and Y, the conditions for (bi)simulation of Def. 5 are equivalent to the conditions of Def. 10, with this notion of openness. Moreover, every (bi)simulation span, in the sense of Def. 10, factors through a (bi)simulation relation, and so the final (bi)similarity is the greatest (bi)simulation relation, (bi)similarity.

When $\mathcal{C}$ is a topos, the open morphisms can be seen as coalgebra homomorphisms in the sense of [4]. When $\mathcal{C} = \mathbf{Set}$ and $L = A \times (-)$, the open morphisms are exactly the open maps of transition systems as defined in [26]. We have the following result.

Proposition 11 *Let $\mathrm{Ar}_L \to L$ be a signature in a ΠW-pretopos $\mathcal{C}$ with universal I-indexed coproducts, for some set I. The category L-$\mathbf{LTS}$ has pullbacks and universal I-indexed coproducts, and the open maps in L-$\mathbf{LTS}$ satisfy Axioms A1, A3 and A5 above.*

Congruence theorem. We can now state the congruence theorem in full generality.

Theorem 12 *Let $(\mathcal{S}ystems, T) \to (\mathcal{S}tates, S)$ be a monad lifting, and consider a class of 'open' morphisms in $\mathcal{S}ystems$ that satisfies Axiom A1. Suppose that T preserves open morphisms. Let $(X, x : T(X) \to X)$ and $(Y, y : T(Y) \to Y)$ be T-algebras.*

If x is open, then the final simulation between (X, x) and (Y, y), if it exists, is an S-congruence on the underlying S-algebras.

If x and y are both open, then the final bisimulation between (X, x) and (Y, y), if it exists, is an S-congruence.

In particular, if the multiplication of T is open, then the final bisimulation is a congruence for the free T-algebra.

Relevance of tyft/tyxt. The congruence theorem for categorical tyft/tyxt (Thm. 7) can be understood in the context of Thm. 12, through the following result.

Theorem 13 *Consider the monad lifting induced by a well-founded categorical tyft/tyxt TSS.*

1. *The multiplication of the lifted monad is componentwise open.*
2. *If the TSS is pure and $(P \to R)$ is internally projective then the functorial action of the lifted monad preserves open maps.*

In [14; 20] it is argued that the purity and well-foundedness conditions can be eliminated from the congruence theorem, because the initial labelled transition system induced by any tyft/tyxt TSS can also be induced by a pure well-founded tyft/tyxt TSS. However, these kinds of results do not fit well into the general framework of this section, because two different TSSs may have different categories of models, even though their initial models coincide.

4.3. Limits of monads and conservativity

Combining TSSs can be seen as taking limits of monads. The category $\mathbf{Mnd}$ does not have all limits, but those limits that it has are preserved by the construction of categories of algebras, $\mathbf{Mnd} \to \mathbf{CAT}$.

Suppose, for example, that we want to add a left-merge operation to our fragment of CCS [e.g. 3, Sec. 5.4.5]. One way to do this is to consider the TSS for the merge operation alone: it has one operator and one rule. This TSS gives rise to a monad $T_{\text{L-MERGE}}$ on the category $\mathbf{LTS}$ of transition systems. The following pullback exists in $\mathbf{Mnd}$.

$$\begin{tikzcd}
(\mathbf{LTS}, T_{\text{CCS}} + T_{\text{L-MERGE}}) \arrow[r] \arrow[d] & (\mathbf{LTS}, T_{\text{L-MERGE}}) \arrow[d] \\
(\mathbf{LTS}, T_{\text{CCS}}) \arrow[r] & (\mathbf{LTS}, \mathrm{id})
\end{tikzcd}$$

(We write $(T_{\text{CCS}} + T_{\text{L-MERGE}})$ because some authors [e.g. 27; 22] refer to this monad as the *sum* of T_{CCS} and $T_{\text{L-MERGE}}$.) Algebras for $(T_{\text{CCS}} + T_{\text{L-MERGE}})$ are models of the combination of the theory for CCS with the theory of left-merge.

We have the following general conservativity result.

Theorem 14 *Let $\mathcal{S}ystems$ be a locally presentable category with universal ω-sums, and with a class of open morphisms satisfying axioms A1, A3, A5. Let T_1, T_2 be finitary monads on $\mathcal{S}ystems$. If T_1 preserves open morphisms and the unit of T_2 is componentwise open, then the natural injection $T_1 \to (T_1 + T_2)$ is componentwise open.*

Here, we have assumed that $\mathcal{S}ystems$ is locally presentable and that the monads are finitary because the result can then be proved using an inductive characterization of the sum $(T_1 + T_2)$, following [27].

Relevance of tyft/tyxt. The conservativity result for tyft/tyxt (Thm. 8) can be understood in the context of Thm. 14, through Thm. 13 and the following result.

Proposition 15 *If all the rules in a categorical pre-tyft/tyxt TSS are in tyft format, i.e. if $R \to (\Sigma + 1)$ factors through $\Sigma \to (\Sigma + 1)$, then the unit of the lifted monad is componentwise open.*

5. Advanced examples I: name-passing calculi

We now apply the developments of the previous sections to the π-calculus [34; 40], a simple calculus for describing the communication of channel names along named channels. The semantics of the π-calculus involves variable binding and substitution, and so we investigate the categorical tyft/tyxt format in the setting of nominal sets [17] and nominal substitutions [12].

5.1. Nominal sets, nominal substitutions and wide-open bisimulation

We recall some definitions from the theory of nominal sets [e.g. 17]. For the remainder of this paper we fix an infinite set $\mathcal{N}$ of names. We write $\mathsf{Sym}(\mathcal{N})$ for the group of permutations on $\mathcal{N}$.

Recall that a *permutation set* is a set X equipped with a function $\cdot_X : \mathsf{Sym}(\mathcal{N}) \times X \to X$ that respects the identity and group action appropriately. *Equivariant functions* between permutation sets are functions between the underlying sets that preserve the permutation action structure.

A finite set of names is said to *support* an element of a permutation set if every permutation that fixes those names also fixes that element. If an element $x \in X$ has a finite support, then it also has a minimal one, $\mathrm{supp}(x)$, which is the intersection of all other supports.

A *nominal set* is a permutation set in which every element has a finite support. The category $\mathbf{Nom}$ of nominal sets and equivariant functions has a lot of structure: it is a Grothendieck topos. The set $\mathcal{N}$ of names, equipped with the obvious permutation action, is a nominal set. For every nominal set X there is a nominal set $[\mathcal{N}]X$, the quotient of $\mathcal{N} \times X$ by the equivalence relation ('α-equivalence'):

$$(a, x) \sim_\alpha (b, y) \iff \\ \exists c \in \mathcal{N}.\ c \notin \mathrm{supp}(x, y) \text{ and } (a\ c) \cdot x = (b\ c) \cdot y \quad .$$

We write $\langle a \rangle x$ for the equivalence class containing (a, x).

An example of a nominal set is the set of terms of the π-calculus. The permutation action permutes the free names of terms.

Nominal substitutions. A *nominal substitution* [12; 41, Sec. 7.3] is a nominal set X equipped with an equivariant function $\mathrm{sub}_X : \mathcal{N} \times [\mathcal{N}]X \to X$ subject to four conditions. For clarity, we write $[^b/_a]x$ for $\mathrm{sub}_X(b, \langle a \rangle x)$.

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*: $[^d/_b][^c/_a]x = [^c/_a][^d/_b]x$, when $c \neq b \neq a \neq d$.

A *homomorphism* between nominal substitutions is an equivariant function that respects the substitution structure. We write $\mathbf{NSub}$ for the category of nominal substitutions and homomorphisms between them. The category of nominal substitutions is a Grothendieck topos. Indeed, it is a sheaf subcategory of the category $\mathbf{Set}^{\mathbb{F}}$, studied by [13] (see [41, Sec. 7.3.1]). The forgetful functor $\mathbf{NSub} \to \mathbf{Nom}$ preserves products, and we can lift the additional structure of $\mathbf{Nom}$ along it. The nominal set of names has an evident nominal substitution structure. For any nominal substitution X, the exponential $X^{\mathcal{N}}$ has underlying nominal set $[\mathcal{N}]X$. As a result of axioms 1–4, the equivariant function $\mathrm{sub}_X : \mathcal{N} \times [\mathcal{N}]X \to X$ is, in fact, a homomorphism of nominal substitutions; indeed, it is the evaluation map for the exponential.

Syntax of the π-calculus. The operators of the π-calculus form the nominal substitution

$$\Sigma_\pi = \textit{nil}{:}1 + \textit{par}{:}1 + \textit{res}{:}1 + \textit{inp}{:}\mathcal{N} + \textit{out}{:}(\mathcal{N} \times \mathcal{N})$$

and the arities form the nominal substitution

$$\mathrm{Ar}_{\Sigma_\pi} = \textit{nil}{:}0 + \textit{par}{:}2 + \textit{res}{:}\mathcal{N} \\ + \textit{inp}{:}(\mathcal{N} \times \mathcal{N}) + \textit{out}{:}(\mathcal{N} \times \mathcal{N}).$$

For instance, the arity of the *res* operator, for restriction, is $\mathcal{N}$. By the isomorphism $X^{\mathcal{N}} \cong [\mathcal{N}]X$, we see that *res* is a binding operator. There is one *inp* operator for every name, representing input on that named channel. Fixing a name a, the arity of $\textit{inp}_a$, is $\mathcal{N}$, for it takes one parameter, with a name bound in it, specifying what to do once a name is input. The free algebra W_{Σ_π} is the set of π-calculus terms up-to α-equivalence and subject to the evident substitution action.

Labels of the π-calculus. The labels of the π-calculus form the nominal substitution

$$L_\pi = \mathsf{tau}{:}1 + \mathsf{inp}{:}(\mathcal{N} \times \mathcal{N}) + \mathsf{out}{:}(\mathcal{N} \times \mathcal{N}) + \mathsf{bout}{:}\mathcal{N}$$

and the arities form the nominal substitution

$$\mathrm{Ar}_{L_\pi} = \mathsf{tau}{:}1 + \mathsf{inp}{:}(\mathcal{N} \times \mathcal{N}) \\ + \mathsf{out}{:}(\mathcal{N} \times \mathcal{N}) + \mathsf{bout}{:}(\mathcal{N} \times \mathcal{N}).$$

Thus there is a unary silent label, and, for every pair of names, there is a unary input label and a unary output label. For every channel name, there is a bound output label, bout_a, describing the output of new names. It has arity $\mathcal{N}$, for it is a binding label.

Here we are considering the labels for the early semantics, but the late semantics is handled in much the same way.

Labelled transition systems. An L_π-labelled transition system is a nominal substitution X of states with a nominal substitution $(\rightsquigarrow_X) \subseteq X \times L_\pi(X)$. Relations in $\mathbf{NSub}$ are relations in $\mathbf{Set}$ that are substitution-closed, in the sense that

$$x \rightsquigarrow_X l : \vec{x} \implies [^b/_a]x \rightsquigarrow_X [^b/_a](l : \vec{x}) \quad . \tag{2}$$

A first example of a labelled transition system is the one with carrier W_{Σ_π}, and with relation $(\rightsquigarrow_\pi) \subseteq X \times L_\pi(X)$ derived from the standard semantics [e.g. 40, Def 1.3.2]:

$$\begin{array}{lcl} p \rightsquigarrow_\pi \mathsf{tau} : q & \iff & p \xrightarrow{\tau} q \\ p \rightsquigarrow_\pi \mathsf{inp}_{a,b} : q & \iff & p \xrightarrow{ab} q \\ p \rightsquigarrow_\pi \mathsf{out}_{a,b} : q & \iff & p \xrightarrow{\bar{a}b} q \\ p \rightsquigarrow_\pi \mathsf{bout}_a : \langle b \rangle q & \iff & p \xrightarrow{\bar{a}(b)} q. \end{array}$$

The substitution closure condition, (2), is a standard theorem of the π-calculus [40, Lem. 1.4.8].

Wide-open bisimulation. An L_π-bisimulation relation on this transition system, in the sense of Def. 5, is an early bisimulation [40, Def. 2.2.1] that is a relation in $\mathbf{NSub}$, i.e., is substitution-closed. The greatest L_π-bisimulation is thus the greatest substitution-closed early bisimulation. We refer to substitution-closed early bisimulations as *wide-open bisimulations*. (The term 'open' here has nothing to do with Sec. 4.2.)

5.2. A tyft TSS for name-passing

We now proceed to demonstrate how the specification of the π-calculus, as originally given in [34], can be seen as a categorical tyft TSS in the category of nominal substitutions. The categorical tyft format in this category guarantees that wide-open bisimilarity is a congruence.

Definition of the π-calculus: input. Consider the input rule of the π-calculus.

$$\frac{}{a(c)v \xrightarrow{ab} [^b/_c]v} \tag{3a}$$

We assert that, in terms of conventional logic, this rule, as it stands, does not have a clear meaning. We will consider it as a family of rules, indexed by pairs of names, and we will rewrite it as follows.

$$\textit{rule-inp}_{a,b} : \quad \frac{}{\textit{inp}_a\langle c\rangle v_{a,b,c} \rightsquigarrow \mathsf{inp}_{a,b} : v_{a,b,b}} \tag{3b}$$

The purpose of this rewriting is to remove 'syntactic sugar': we have still not given the rule family a precise meaning. However, it is not difficult to derive a tyft TSS, in the sense of Def. 3, from this revised presentation. There is a rule for each pair of names: $R \cong \mathcal{N} \times \mathcal{N}$. There are no premises: $P = 0$. There is a variable for every rule and every possible binding name: $V \cong R \times \mathcal{N}$. We write the variables as $v_{a,b,c}$, as in (3b).

Notice that the substitution in the right-hand side of (3a) has become implicit by the modification of the variable in (3b). To use a rule *rule-inp*$_{a,b}$ in a proof tree, one must provide an valuation for the variables. A value must be given for $v_{a,b,c}$, for every name c. By the isomorphism $(\mathrm{W}_\Sigma)^{\mathcal{N}} \cong [\mathcal{N}]\mathrm{W}_\Sigma$, this amounts to giving a name bound in a term, i.e., a value for the parameter of the left-hand side.

Definition of the π-calculus: scope opening. For a second example, consider the scope opening rule from the π-calculus.

$$\frac{v \xrightarrow{\bar{a}b} v'}{\boldsymbol{\nu} b.\, v \xrightarrow{\bar{a}(b)} v'}(a \neq b) \tag{4a}$$

The syntactic sugar can be removed, and the rule can be rewritten with fresh variables as follows. We have a family of rules, one for each name a.

$$\textit{rule-open}_a : \quad \frac{\left(v_{a,c} \rightsquigarrow \mathsf{out}_{a,c} : v'_{a,c}\right)_{c\in\mathcal{N}}}{\textit{res}\langle b\rangle v_{a,b} \rightsquigarrow \mathsf{bout}_a : \langle d\rangle v'_{a,d}} \tag{4b}$$

Again, one can derive a categorical tyft TSS, in the sense of Def. 3, from this revised presentation. There is a rule *rule-open*$_a$ for each name $a \in \mathcal{N}$, so $R \cong \mathcal{N}$. For each rule, there is an $\mathcal{N}$-indexed family of premises, because there is a premise for every possible choice of binder for the left-hand side of the conclusion. Thus $P \cong \mathcal{N} \times R$. There are two variables, $v_{a,b}$, $v'_{a,b}$, for every pair (a, b) of names: so $V \cong 2 \times \mathcal{N} \times \mathcal{N}$.

The side-condition $(a \neq b)$ in the right-hand side of the conclusion of (4a) has become implicit in (4b). To use a rule *rule-open*$_a$ in a proof tree, one must fulfil the premises for every possible value of b, in a parametric manner. Thus the premises must be fulfilled for every $b \neq a$, and also for $b = a$. This one extra premise does no harm, because the transition relations are closed under substitution (see (2)).

Internal projectivity. The congruence theorem (Thm. 7) requires that the premise map $P \to R$ is internally projective. This is not a problem for the π-calculus, because the homomorphism $\mathcal{N} \to 1$ is internally projective. Some homomorphisms in $\mathbf{NSub}$ are *not* internally projective, though, and we use this fact to illustrate the necessity of the internal projectivity condition in Thm. 7.

Let $\mathrm{List}(\mathcal{N})$ and $\mathcal{P}_\mathrm{f}(\mathcal{N})$ be respectively the set of lists of names, and the set of finite sets of names. We equip these sets with the evident nominal substitution structure. The object $\mathcal{P}_\mathrm{f}(\mathcal{N})$ is not internally projective in $\mathbf{NSub}$: consider the epi homomorphism $|-| : \mathrm{List}(\mathcal{N}) \to \mathcal{P}_\mathrm{f}(\mathcal{N})$, taking a list of names to its set of members, and observe that the morphism $|-|^{\mathcal{P}_\mathrm{f}(\mathcal{N})} : \mathrm{List}(\mathcal{N})^{\mathcal{P}_\mathrm{f}(\mathcal{N})} \to \mathcal{P}_\mathrm{f}(\mathcal{N})^{\mathcal{P}_\mathrm{f}(\mathcal{N})}$ is *not* an epimorphism.

Consider a TSS with a unary label for every finite set of names, and a constant label ok. The syntax is formed from a constant *set*$_A$ and *list*$_L$ for each set A and each list L, together with two constants, *gen-sets* and *gen-lists*, and a unary operator *check*. The rules are as follows.

$$\frac{}{\textit{gen-sets} \rightsquigarrow A : \textit{set}_A}\, A \in \mathcal{P}_\mathrm{f}(\mathcal{N}) \qquad \frac{\left(v \rightsquigarrow A : v'_A\right)_{A\in\mathcal{P}_\mathrm{f}(\mathcal{N})}}{\textit{check}(v) \rightsquigarrow \mathsf{ok}}$$

$$\frac{}{\textit{gen-lists} \rightsquigarrow |L| : \textit{list}_L}\, L \in \mathrm{List}(\mathcal{N})$$

Notice that the rule for *check* has a $\mathcal{P}_\mathrm{f}(\mathcal{N})$-indexed family of premises. In the induced transition system, the constants *gen-sets* and *gen-lists* are bisimilar. Bisimilarity is *not* a congruence, though, since the context *check*$(-)$ distinguishes them. The term *check*(*gen-lists*) is stuck. An appropriate valuation of the variables would amount to an element of the set $\mathrm{List}(\mathcal{N})^{\mathcal{P}_\mathrm{f}(\mathcal{N})}$, which contains only the constant functions.

6. Advanced example II: Open bisimulation

For a more sophisticated example, we consider categorical tyft/tyxt TSSs in a presheaf category. This presheaf category has much in common with the category studied in [19; 31]. In this category, the categorical tyft/tyxt format is essentially the format proposed by [45]. Indeed, following [8] we arrive at a treatment of the open bisimilarity of [39].

Pragmatic motivation. Wide-open bisimilarity can be criticised for being too discriminating. Suppose that we add a name-equality check to the π-calculus. Wide-open bisimilarity distinguishes the processes

$$\boldsymbol{\nu}a.\,\bar{b}a.\,\textit{if } a = b \textit{ then } \bar{a}a \qquad \text{and} \qquad \boldsymbol{\nu}a.\,\bar{b}a. \tag{5}$$

This distinction contradicts our intuitions about name generation in the π-calculus: the name a generated in the left-hand term should never be equal to the pre-existing b.

Open bisimilarity has been proposed as a remedy to such problems [39, Sec. 6]. Roughly speaking, open bisimilarity works by considering processes within environments to keep track of which names should be kept distinct.

Once we work with open bisimilarity, the treatment of rules in Sec. 5 appears rather primitive. There is nothing to stop us from adding a unary operator *bout-to-inp* to the π-calculus, and the following rule.

$$\frac{v \xrightarrow{\bar{a}(b)} v'}{\textit{bout-to-inp}(v) \xrightarrow{ac} [{}^{c}/_{b}]v'} \tag{6}$$

This semantics breaks the congruence of genuine open bisimilarity. The processes of (5) are open bisimilar, but are distinguished in the context *bout-to-inp*$(-)$.

Environments. A *distinction relation* is a finite irreflexive and symmetric binary relation on $\mathcal{N}$. An *environment* is a triple $(A, A', \#_A)$: A is a set of names; A' is a subset of A; $\#_A$ is a distinction relation on A. An environment is to be thought of as a specification of which names are known, which names have been generated through a bound output transition, and which names are known to be distinct. Considering the left-hand process of (5), we might write transitions with environments, as

$$\begin{aligned}&\{b\}, \varnothing, \varnothing \vdash \boldsymbol{\nu}a.\,\bar{b}a.\,\textit{if } a = b \textit{ then } \bar{a}a\\ &\quad\xrightarrow{\bar{b}(a)} \{a, b\}, \{a\}, \{a \neq b\} \vdash \textit{if } a = b \textit{ then } \bar{a}a.\end{aligned} \tag{7}$$

After one step, the environment records that the names a and b are known to be distinct, and that the name a has been introduced through a bound output.

A category of presheaves. A morphism between environments, $(A, A', \#_A) \to (B, B', \#_B)$ is a function $f : A \to B$, such that $f(A') \subseteq B'$ and $f(\#_A) \subseteq \#_B$. Thus the collection of environments forms a category, $\mathbb{E}\text{nv}$.

Given this category $\mathbb{E}\text{nv}$ of environments, the category of presheaves $\mathbf{Set}^{\mathbb{E}\text{nv}}$ is a standard setting for considering sets of elements in environments.

Every nominal substitution X can be considered as a presheaf: we let $X(A, A', \#_A)$ be the set of elements of X that are supported by A. The functorial action is derived from the substitution action. This embedding preserves colimits, finite limits, and exponentials.

The nominal substitution of names, $\mathcal{N}$, thus gives rise to a presheaf: we have $\mathcal{N}(A, A', \#_A) = A$. We also define a presheaf of *name-constants*, $\mathcal{NC} : \mathbb{E}\text{nv} \to \mathbf{Set}$ as the subpresheaf of $\mathcal{N}$ satisfying $\mathcal{NC}(A, A', \#_A) = A'$.

The category of environments has an asymmetric monoidal structure: the tensor is given by

$$\begin{aligned}&(A, A', \#_A) \ltimes (B, B', \#_B) = \\ &\qquad\left(\begin{array}{l} A \uplus B, A' \uplus B', \\ \quad \#_A \uplus \#_B \cup \{(a, b) \mid a \in A', b \in B\}^{\S} \end{array}\right)\end{aligned}$$

where $(-)^{\S}$ indicates symmetric closure. Following [10], this monoidal structure induces a biclosed monoidal structure on $\mathbf{Set}^{\mathbb{E}\text{nv}}$. We write $\ltimes$ for the monoidal product, and $[X](-)$ for the right adjoint of $X \ltimes (-)$. The functor $[\mathcal{N}](-)$ introduces new binding names, as before, whereas $[\mathcal{NC}](-)$ introduces new distinct names.

The unit of $\ltimes$ is terminal, and so there is a natural transformation from $\ltimes$ to the cartesian product. As a result, the monoidal closed structure arises in all the slice categories.

Syntax and labels revisited. The signatures for π-calculus syntax and labels, introduced in Sec. 5, can be reinterpreted in this setting. It is important, however, to modify the arities: the restriction operator and the bound output label must be of arity $\mathcal{NC}$, rather than $\mathcal{N}$.

- $\begin{aligned}\mathrm{Ar}_{\Sigma_\pi} &= \textit{nil} : 0 + \textit{par} : 2 + \textit{res} : \mathcal{NC}\\ &\quad + \textit{inp} : (\mathcal{N} \times \mathcal{N}) + \textit{out} : (\mathcal{N} \times \mathcal{N}) \quad .\end{aligned}$

The free algebra W_{Σ_π} for this revised signature is the same as the presheaf version of the nominal substitution of π-calculus terms.

- $\begin{aligned}\mathrm{Ar}_{L_\pi} &= \mathsf{tau} : 1 + \mathsf{inp} : (\mathcal{N} \times \mathcal{N})\\ &\quad + \mathsf{out} : (\mathcal{N} \times \mathcal{N}) + \mathsf{bout} : (\mathcal{N} \times \mathcal{NC}).\end{aligned}$

An L_π-labelled transition system is a presheaf X together with a subpresheaf of $X \times L_\pi(X)$. Externally, this can be seen as a transition relation over states-in-environments, as indicated in (7).

An L_π-bisimulation for such labelled transition systems, in the sense of Def. 5, is a bisimulation that is a subpresheaf in $\mathbf{Set}^{\mathbb{E}\text{nv}}$. Thus it is a "$T$-open bisimulation" in the sense of [8]. The correspondence between T-open bisimilarity and open bisimilarity is explained in loc. cit..

TSSs for open bisimulation. By considering the categorical tyft/tyxt TSSs of Sec. 3 in the category $\mathbf{Set}^{\mathbb{E}\mathrm{nv}}$, we arrive at a categorical treatment of the semantics of open bisimulation. The theory of Sec. 3 must be modified to use the local monoidal closed structure of $\mathbf{Set}^{\mathbb{E}\mathrm{nv}}$, rather than the local cartesian closed structure, where relevant.

TSSs in this setting can be understood as the TSSs of Sec. 5, for *wide*-open bisimilarity, subject to certain constraints. These constraints are very similar to the typing constraints in [45]. We have that $\mathcal{NC} \subseteq \mathcal{N}$ and, by contravariance, $[\mathcal{N}]X \subseteq [\mathcal{NC}]X$, for each X. There is a substitution morphism, $\mathcal{N} \ltimes [\mathcal{N}]X \to X$, but *not* a morphism $\mathcal{N} \ltimes [\mathcal{NC}]X \to X$ in general. As a consequence, the malign rule (6) is not a valid TSS in this category.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, 2001.

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inform. and Comput.*, 148(1): 1–70, 1999.

[3] L. Aceto, W. Fokkink, and F. Vaandrager. Structural operational semantics. In Bergstra, Ponse, and Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2001.

[4] P. Aczel and N. Mendler. A final coalgebra theorem. In *CTCS'89*, volume 389 of *Lecture Notes in Comput. Sci.*, pages 357–365. Springer, 1989.

[5] T. Altenkirch, N. Ghani, P. Hancock, C. McBride, and P. Morris. Indexed containers. Available online, 2006.

[6] S. Awodey and A. Bauer. Propositions as [types]. *J. Log. Comput.*, 14(4):447–471, 2004.

[7] J. Bengtson and J. Parrow. Formalising the pi-calculus using nominal logic. In *FoSSaCS'07*, pages 63–77, 2007.

[8] S. Briais and U. Nestmann. Open bisimulation, revisited. *Theoret. Comput. Sci.*, 386:236–271, 2007.

[9] A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: a coalgebraic view of open systems. *Theoret. Comput. Sci.*, 280:163–192, 2002.

[10] B. Day. On closed categories of functors. In *Reports of the Midwest Category Seminar, IV*, volume 137 of *Lecture Notes in Math.*, pages 1–38. Springer, 1970.

[11] R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoret. Comput. Sci.*, 37:245–267, 1985.

[12] M. P. Fiore and S. Staton. A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In *LICS'06*, pages 49–58, 2006.

[13] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding (extended abstract). In *LICS'99*, 1999.

[14] W. Fokkink. The tyft/tyxt format reduces to tree rules. In *TACS'94*, pages 440–453, 1994.

[15] W. J. Fokkink and C. Verhoef. A conservative look at operational semantics with variable binding. *Inform. and Comput.*, 146(1):24–54, 1998.

[16] M. J. Gabbay. The π-calculus in FM. In *Thirty Five Years of Automating Mathematics*. Kluwer, 2003.

[17] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[18] N. Gambino and M. Hyland. Wellfounded trees and dependent polynomial functors. In *TYPES'03*, 2003.

[19] N. Ghani, K. Yemane, and B. Victor. Relationally staged computations in calculi of mobile processes. In *CMCS'04*, pages 105–120, 2004.

[20] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inform. and Comput.*, 100(2):202–260, 1992.

[21] F. Honsell, M. Miculan, and I. Scagnetto. π-calculus in (co)inductive-type theory. *Theoret. Comput. Sci.*, 253:239–285, 2000.

[22] M. Hyland, G. D. Plotkin, and J. Power. Combining effects: Sum and tensor. *Theoret. Comput. Sci.*, 357:70–99, 2006.

[23] P. T. Johnstone. *Sketches of an elephant: A Topos Theory Compendium*. Oxford University Press, 2003.

[24] A. Joyal and I. Moerdijk. *Algebraic set theory*. Cambridge University Press, 1995.

[25] A. Joyal and I. Moerdijk. A completeness theorem for open maps. *Ann. Pure Appl. Logic*, 70(1):51–86, 1994.

[26] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Inform. and Comput.*, 127(2):164–185, 1996.

[27] G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bull. Austral. Math. Soc.*, 22:1–83, 1980.

[28] B. Klin. Bialgebraic methods in structural operational semantics. In *SOS'06*, pages 33–43, 2006.

[29] M. E. Maietti. Modular correspondence between dependent type theories and categorical universes including pretopoi and topoi. *Math. Structures Comput. Sci.*, 15(6), 2005.

[30] P. Martin-Löf. *Intuitionistic type theory*. Bibliopolis, Napoli, 1984.

[31] M. Miculan and K. Yemane. A unifying model of variables and names. In *FoSSaCS'05*, pages 170–186, 2005.

[32] C. A. Middelburg. Variable binding operators in transition system specifications. *J. Log. Algebr. Program.*, 47:15–45, 2001.

[33] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

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

[35] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Ann. Pure Appl. Logic*, 104:189–218, 2000.

[36] M. R. Mousavi, M. A. Reniers, and J. Groote. SOS formats and meta-theory. *Theoret. Comput. Sci.*, 373:238–272, 2007.

[37] E. Palmgren. A categorical version of the Brouwer-Heyting-Kolmogorov interpretation. *Math. Structures Comput. Sci.*, 14:57–72, 2004.

[38] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Aarhus, 1981.

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

[40] D. Sangiorgi and D. Walker. *The π-calculus: a theory of mobile processes*. Cambridge University Press, 2001.

[41] S. Staton. *Name-Passing Process Calculi: Operational Models and Structural Operational Semantics*. PhD thesis, Cambridge, 2007. Technical Report UCAM-CL-TR-688.

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

[43] A. Tiu and D. Miller. A proof search specification of the π-calculus. In *FGUC'04*, pages 79–101, 2004.

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

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