Bialgebraic Methods in Structural Operational Semantics

(Invited Talk)

Bartek Klin[1,2]

Warsaw University, Edinburgh University

Abstract

Bialgebraic semantics, invented a decade ago by Turi and Plotkin, is an approach to formal reasoning about well-behaved structural operational specifications. An extension of algebraic and coalgebraic methods, it abstracts from concrete notions of syntax and system behaviour, thus treating various kinds of operational descriptions in a uniform fashion.
In this talk, the current state of the art in the area of bialgebraic semantics is presented, and its prospects for the future are sketched. In particular, a combination of basic bialgebraic techniques with a categorical approach to modal logic is described, as an abstract approach to proving compositionality by decomposing modal logics over structural operational specifications.

Keywords: Structural operational semantics, category theory, algebra, coalgebra, bialgebra

1 Introduction

Since its invention in the early 1980's, Structural Operational Semantics (SOS) [36,35,1] has been one of the most popular and successful frameworks for the formal description of programming languages and process calculi. Not only it has become the formalism of choice for a clear and concise presentation of innumerable ideas and formalisms (see [5] for many examples), it has also been proved a viable option for the description of fully grown programming languages [31]. In the structural operational approach, the semantics of programs (or processes) is described by means of transition systems, induced by inference rules following their syntactic structure. The intuitive appeal of this approach and, importantly, its inherent support for modelling nondeterministic behaviour, makes it a natural framework for describing computational languages.

There are many types of standard reasoning applied to structural operational descriptions to check whether they "behave well". For example, one often wants

[1] Supported by EPSRC grant EP/D039045/1.

[2] Email: bklin@inf.ed.ac.uk

to prove that a certain equivalence relation on processes, defined by the transition system induced from a specification, is compositional and thus allows for inductive reasoning. Further, one might be interested in generating a set of equations characterising a chosen process equivalence. In other cases, a language with an operationally defined semantics is extended with new operators, and one would like to ensure that the extension is conservative, i.e., that the behaviour of the old operators is left unchanged. More generally, one would like to compare two languages and check whether one can be translated to the other such that its operational semantics, or a process equivalence, is preserved. One could also try to combine two languages, or a set of separately defined operators, and reason about the resulting language in a modular fashion, based on features of its components.

Proofs of properties such as listed above can be quite demanding, and it would be unfortunate if they had to be done from scratch for each language considered. Indeed, the multitude of existing examples of structural operational descriptions, and the continuous appearance of new ones, calls for a mathematical framework — a theory of SOS — that would facilitate standard types of reasoning performed on language specifications.

In the simplest and most widely studied form of SOS, operational specifications induce labelled transition systems (LTSs) where processes are closed terms over some algebraic signature, and labels have no specific structure. In that case, a reasonably general theory based on the notion of Transition System Specification (TSS) has been developed, and much progress towards overcoming the above difficulties has been made (see [1,14] for a survey). For most classical process equivalences congruence formats have been provided, i.e., syntactic restrictions on inference rules that guarantee the equivalences compositional. Much work on equational definability and conservative extensions has also been done. Least progress has been made in the area of language translation and modularity, with the notable exception of Modular Structural Operational Semantics by Mosses [32], which enjoys better modularity than its classical version when applied to standard cases.

However, the issue becomes more complex with the continuing appearance of new syntactic and computational paradigms dictated by the everchanging world of computing. Examples include probabilistic, timed and hybrid systems, ones with global or local state, name passing, process passing, and stochastic systems. All these features can be described in some forms of structural operational semantics. Arguably though, the treatment of these forms in the classical TSS framework is rather superficial, as it does not take into account the semantic structure of transition systems involved. As a result, the existing approaches to checking whether semantic descriptions are well behaved, are hard to adapt to these new paradigms; instead one usually needs to rework the old solutions from scratch in each case.

It is therefore desirable to have a theory of SOS that would abstract from the syntactic and computational paradigms involved, and allows one to define and prove useful properties of semantic descriptions on an abstract level. With such a theory, SOS will hopefully be better prepared for future changes in the world of computing.

The basics of such a theory were proposed by Turi and Plotkin in [40]. Its beginnings lie in the coalgebraic account of transition systems (see [17] for a gentle introduction and [38] is a good reference), where the notion used to classify various

kinds of processes is that of behaviour, modelled as a functor on a category and formally representing the vague concept of computational paradigm. Combined with the classical algebraic approach to syntax, these techniques lead to the development of *bialgebraic semantics* of processes, which turned out to generalise and explain many aspects of well-behaved operational semantics.

Since then, the bialgebraic approach has developed considerably, and it has lead to several results interesting to a wider community. The purpose of this extended abstract is to give a gentle introduction to basic bialgebraic techniques, aimed at researchers familiar with SOS; to sketch their current stage of development; and to present the author's personal views on their future prospects. Note that much of interesting work on bialgebras in semantics, some of it less directly related to SOS, has been omitted in this introductory paper, and the bibliography included is far from complete.

Acknowledgements. The author is grateful to Gordon Plotkin for continuing cooperation, and to Alexander Kurz for useful discussions.

2 Bialgebraic semantics

In this section, the framework of bialgebraic semantics is presented together with some results that have been obtained through its use. To appreciate the following development, the reader is expected to be familiar with the basics definitions and techniques of structural operational semantics (see [1] for a reference). Bialgebras are defined in the language of category theory, so familiarity with basic notions such as category, functor and natural transformation is also recommended (the first chapters of [2,30] are good references).

2.1 Processes as coalgebras

In the standard framework of SOS as used e.g. in process algebra [1], structural operational descriptions specify labelled transition systems (LTSs), i.e., triples $\langle X, A, \rightarrow \rangle$, where X is a set of processes, A a set of labels, and $\rightarrow \subseteq X \times A \times X$ is a labelled transition relation. It is easy to see LTSs as functions

$$h : X \rightarrow \mathcal{P}(A \times X)$$

where $\mathcal{P}$ is the powerset construction and $\times$ is cartesian product. Indeed, an LTS maps a process $x \in X$ to the set of all tuples $\langle a, y \rangle$ such that $x \stackrel{a}{\longrightarrow} y$. In the language of category theory, a map as above is called a *coalgebra* for the *functor* $\mathcal{P}(A \times -)$. In general, for any functor B, a B-coalgebra is a map (function) $h : X \rightarrow BX$ for some object (set) X.

As it happens, coalgebras for some other functors on the category **Set** of sets and functions correspond to other well-known types of transition systems. For example:

- Coalgebras for $\mathcal{P}_{\mathrm{f}}(A \times -)$, where $\mathcal{P}_{\mathrm{f}}$ is the *finite* powerset functor, are *finitely branching* LTSs. Coalgebras for $(\mathcal{P}_{\mathrm{f}}(-))^A$ are *image finite* LTSs.
- Coalgebras for $\mathcal{D}(A \times -) + 1$, where D is the probability distribution functor, are generative probabilistic transition systems.

- Coalgebras for $(S \times (1 + -))^S$ are deterministic transition systems with state and termination.

Many other examples of systems modelled as coalgebras for functors on **Set** can be found in [38]. Coalgebras for functors on other categories have also been used; for example, in [8], coalgebras for a certain functor on the category **NSet** of nominal sets and equivariant functions [13] are shown to correspond to a kind of labelled transition systems with name binding. The coalgebraic abstraction allows one to treat many different kinds of systems in a uniform manner. At the same time, many important notions used in reasoning about transition systems can be explained at the abstract, coalgebraic level; examples include canonical process equivalences such as bisimilarity, or modal logics such as Hennessy-Milner logic.

2.2 Terms as algebras

In simple types of structural operational semantics, processes are closed terms over some algebraic signature. It is standard to consider sets of such terms as algebras for certain functors on **Set**. For example, a language syntax described by the grammar

$$t ::= \mathtt{nil} \mid a.t \mid t + t \mid t||t\,,$$

where a ranges over a fixed set A, corresponds to the functor

$$\Sigma X = 1 + A \times X + X \times X + X \times X$$

where 1 is a singleton set, $\times$ is cartesian product and $+$ is disjoint union. Note that an element of the set ΣX can be seen as a simple term of the above grammar, build of exactly one syntactic construct with variables from X. It turns out that algebras for the signature (in the usual sense of universal algebra) are maps

$$g : \Sigma X \to X$$

with X the algebra carrier. This way, a simple syntax corresponds to a functor on **Set**. To model more advances syntactic features such as variable binding, one needs to move to more complex categories, such as **NSet**.

If a functor Σ corresponds to an algebraic signature, then the set of terms over the signature and over a set X of variables is denoted $T_\Sigma X$, or TX if Σ is clear from context. In particular, $T0$ is the set of closed terms over Σ. This set admits an obvious and canonical algebra structure, denoted $\psi : \Sigma T0 \to T0$. This Σ-algebra is *initial*: for any other algebra $g : \Sigma X \to X$ there is a unique homomorphism from ψ to g, i.e., a map $g^\sharp : T0 \to X$ such that $g^\sharp \circ \psi = g \circ \Sigma\psi$. Intuitively, $g^\sharp$ is defined by structural induction, where g defines the inductive step. The construction T is also a functor, and it is called the monad freely generated by Σ. The notions of initial algebra and freely generated monad do not depend on Σ corresponding to an algebraic signature, and can be defined for many other functors. For more intuitions about this categorical approach to induction, see e.g. [17].

2.3 Abstract GSOS

Simple structural operational descriptions induce LTSs with closed terms as processes. In other words, the set of processes is equipped with both a coalgebraic structure, which maps a process to a structure of its successors, and an algebraic structure, which describes how to obtain a process by combining other processes. Formally, induced LTSs are coalgebras $h : T0 \to BT0$ for a suitable behaviour B, and for T the monad freely generated by syntax Σ.

To model the process of inducing LTSs with syntax abstractly, a sufficiently abstract notion of structural operational description is needed. For a first attempt, consider a standard set of operational inference rules for a toy language with synchronous product:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \otimes y \xrightarrow{a} x' \otimes y'} \qquad \frac{}{a \xrightarrow{a} \mathtt{nil}} \tag{1}$$

where a ranges over a fixed set A of labels. The syntax of this language corresponds, as in §2.2, to the functor

$$\Sigma X = 1 + A + X \times X\,.$$

Rules (1) induce a standard LTS labelled with A, i.e., a coalgebra for the functor

$$BX = \mathcal{P}(A \times X)\,.$$

But how to model the rules on the abstract level? Informally, they define the behaviour (i.e., the set of successors) of a term built of a single syntactic construct and variables, based on some information about the behaviour of subterms represented by the variables. For example, given processes x, y from any set X, and sets of successors for x and for y, the leftmost rule defines the set of successors for the process $x \otimes y$. Note that while successors of x and y are variables and therefore can be thought of as arbitrary elements of X, the derived successors of $x \otimes y$ are simple terms from ΣX. Formally, the left rule in (1) can be modelled as a function

$$\lambda_\otimes : BX \times BX \to B\Sigma X$$

defined by

$$\lambda_\otimes(\beta, \gamma) = \{\, \langle a, x \otimes y\rangle \in A \times \Sigma X \mid \langle a, x\rangle \in \beta \wedge \langle a, y\rangle \in \gamma \,\}\,.$$

Similarly, the right rule represents a function $\lambda_A : A \to B\Sigma X$ defined by

$$\lambda_A(a) = \{\langle a, \mathtt{nil}\rangle\}\,,$$

and even the lack of any rules for the construct `nil` defines its behaviour: the process `nil` has no successors. This can be viewed as a function $\lambda_{\mathtt{nil}} : 1 \to B\Sigma X$:

$$\lambda_{\mathtt{nil}}(*) = \emptyset\,.$$

The three functions can be combined into a function

$$\lambda : \Sigma BX \to B\Sigma X$$

defined by cases and corresponding to (1). Note that the structure of X and the nature of its elements are completely ignored in the definition of λ. Formally, λ is natural over X:

$$\lambda : \Sigma B \Longrightarrow B\Sigma\,. \tag{2}$$

A natural transformation like this is called a *distributive law* of Σ over B, and a first attempt to model structural operational rules would be to consider distributive laws of the syntax functor over the behaviour functor. We have just seen a reasonable example covered by this notion. Moreover, it turns out that the process of inferring LTSs from SOS rules can be explained abstractly at the level of distributive laws. Indeed, the unique algebra morphism h_λ from the initial Σ-algebra as below:

$$\begin{array}{ccccc} T0 & \xleftarrow{\quad\psi\quad} & & & \Sigma T0 \\ {\scriptstyle h_\lambda}\big\downarrow & & & & \big\downarrow{\scriptstyle \Sigma h_\lambda} \\ BT0 & \xleftarrow[B\psi]{} & B\Sigma T0 & \xleftarrow[\lambda_{T0}]{} & \Sigma BT0 \end{array}$$

is an LTS of the required type.[3] The reader is encouraged to check that for the transformation λ defined as above, the inductively defined h_λ is exactly the expected LTS induced by (1).

The above encourages one to model sets of SOS rules as distributive laws. However, there are many examples which do not fit into the simple framework described so far. Consider, for example, rules like

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x}+\mathtt{y} \xrightarrow{a} \mathtt{x}'} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{\mathtt{x}|\mathtt{y} \xrightarrow{a} \mathtt{x}'|\mathtt{y}} \qquad \frac{\mathtt{x} \xrightarrow{a} \mathtt{x}'}{!\mathtt{x} \xrightarrow{a} \mathtt{x}'|!\mathtt{x}}\,. \tag{3}$$

According to the first rule, an inferred successor of a process $x + y$ does not need to be a term built of a single syntactic construct; indeed, it is merely a variable. In the second rule, the inferred successor is not built solely of successors of the subprocesses x and y; instead, the variable y is used itself. In the third rule, both problems occur, although here the inferred successor is not a variable, but a complex term with more than one syntactic construct. All these rules cannot be represented as a natural transformation (2) for the obvious choices of B and Σ.

A more general framework was proposed already in the original paper [40], where distributive laws

$$\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow BT \tag{4}$$

were considered. B-coalgebras can be induced from such laws much the same as shown above for the simple laws. It turns out that for $B = \mathcal{P}(A \times -)$, and for Σ and T corresponding to an algebraic signature, laws of this kind correspond exactly to specifications in the well-known format GSOS [6]; hence the name *abstract GSOS*. In particular, the three problematic rules (3) can be modelled as distributive laws.

In [40], distributive laws dual to (4) were also considered, i.e., natural transformations

$$\nu : \Sigma D \Longrightarrow B(\mathrm{Id} + \Sigma) \tag{5}$$

[3] The pair $\langle \psi, h_\lambda \rangle$ is a λ-*bialgebra* [40], the central notion of bialgebraic semantics.

where D is the comonad cofreely generated by B, just as T in (4) is the monad freely generated by Σ. For $B = \mathcal{P}_{\mathrm{f}}(A \times -)$, DX is the set of finite or infinite trees edge-labelled by A and node-labelled by X, quotiented by strong bisimilarity, and distributive laws (5) correspond to sets of SOS rules in the *safe ntree format* [11,40].

Both GSOS and safe ntree formats guarantee bisimilarity to be a congruence on the induced LTS. An important contribution of [40] was to show that these congruence properties can be proved on the abstract level of distributive laws, and thus they are immediately translated to SOS frameworks based on different notions of syntax and behaviour. Several applications of this result, together with some other work on bialgebraic semantics published so far, are mentioned in the remainder of this section.

2.4 Categorical foundations

In [40], natural transformations of the type (4) and (5) are considered as special cases of the more general notion of a distributive law of a monad over a comonad. In [28,29,37], various types of distributive laws are studied on the abstract, categorical level. In [4], different kinds of distributive laws are studied and related on the concrete example of LTSs; also a complete proof of one-to-one correspondence between abstract GSOS and concrete GSOS specifications is included there.

2.5 Abstract GSOS for probabilistic and timed systems

In [3,4], the abstract GSOS framework is applied to reactive probabilistic systems and probabilistic automata, represented as coalgebras for suitable functors. A congruence format for probabilistic bisimilarity is derived.

In [18,19], the same framework is applied to processes with timed transitions. Congruence results regarding time bisimilarity are proved, and a congruence format for the case of discrete time is derived. In [20,21], the combination of timing with action is studied more carefully, with insights on combining different behaviours to obtain a modular account of semantics.

2.6 Recursion

In [23], abstract GSOS is studied in a CPO-enriched setting, where recursion is possible to express via straightforward fixpoint constructions. There, it is shown how to combine standard GSOS distributive laws with recursive equations to obtain other well-behaved distributive laws. Another bialgebraic approach to recursive equations is [16].

2.7 Name binding

In [9,10], syntax with variable binding was modelled algebraically in a presheaf category, and the standard SOS description of the π-calculus was shown to fit in the abstract GSOS format there, although no actual format was proposed. Recently [8], such a format, a special case of abstract GSOS, has been proposed in the closely related setting of nominal sets [13], with congruence properties related to a version of open bisimilarity. Interestingly, in nominal settings the syntax and behaviour

functors reside in different categories. The basic bialgebraic setting is suitably generalised to accommodate this.

2.8 Van Glabbeek spectrum

In [25,22,24], abstract GSOS is interpreted in certain fibered categories. This allows one to derive congruence formats for process equivalences other than the canonical coalgebraic notion of bisimilarity. In particular, novel formats for completed trace and failures equivalences on LTSs were obtained.

3 Future directions

Bialgebraic techniques are still in the initial stage of development and much remains to be done if a general and practical theory of structural operational semantics is to be achieved. This section briefly presents the author's personal view on the most promising directions of development, and the most important challenges to be taken.

3.1 Relations to reactive systems

In modern process algebra, much attention is paid to a semantic framework alternative to SOS, i.e., to reactive systems [27]. Many languages, such as the π-calculus or ambient calculus, are naturally described in the language of reactive systems, where dynamic behaviour is described by an unlabelled reaction relation together with a suitable structural congruence. The reactive approach is quite intuitive and easy to use; however, it imposes less structure on the described language and in particular it does not easily facilitate compositionality. It would be very desirable to build a bridge between SOS and reactive systems, and be able to translate or compare operational descriptions between the two formalisms. It seems that sufficiently abstract theories of both approaches is indispensable to that end. The bialgebraic approach will hopefully become such a theory for SOS, and [27,39] can be seen as attempts to develop such a theory for reactive systems.

3.2 Modal logic decomposition

In a coalgebraic approach to modal logics and system testing [15,26,34], one considers a contravariant adjunction between a category $\mathcal{C}$ of processes, where coalgebras for a functor B are systems, and a category $\mathcal{D}$ of tests, with a functor L representing the syntax of a logic. The functors B and L are connected along the adjunction, and the connection provides an interpretation of the logic over B-coalgebras.

In the bialgebraic setting, processes are equipped with syntax represented by a functor Σ on $\mathcal{C}$, with a distributive law λ of Σ over B representing operational semantics. It is then natural to consider a functor Γ on $\mathcal{D}$, connected to Σ along the adjunction and representing the "behaviour of the logic". Intuitively, Γ-coalgebras describe ways of decomposing modal logics over the syntax Σ. One can try to come up with a distributive law of L over Γ, connected to λ along the adjunction. If this succeeds, then logic decomposition is compatible with the operational semantics and

the process equivalence defined by the logic L is compositional with respect to the language defined by λ. This provides a general framework for proving congruence results, of which [22,24] is a special case. On the concrete level, the resulting congruence principle is related to work on decomposing modal logics such as [12].

3.3 Equations and weak equivalences

An important tool in practical operational semantics are equations between process terms, allowing language designers to prepare shorter and more intuitive descriptions. While in principle, equations are easily to express on the abstract bialgebraic level by considering syntactic monads other than the freely generated ones, the development of concrete formats based on this observation is a difficult challenge. One would like to match the more concrete development of [33].

Dually, it is also important to treat weak equivalences such as weak bisimilarity in the bialgebraic setting, providing congruence results for them. This would require a general coalgebraic approach to weak equivalences, a major challenge with no satisfactory solution so far.

3.4 Modularity

The ultimate practical goal of a theory of SOS must be a general, easy-to-use and expressive framework for the modular development of operational semantics. The ambition of the bialgebraic approach is to make the framework parametrised by notions of syntax and behaviour. However, many problems need to be solved before such a framework appears. A theory of well-behaved translations between operational descriptions needs to be developed, and notions of morphisms between distributive laws such as those in [41] seem good steps to this end. Modular construction of SOS descriptions will also require techniques for combining rules based on different types of behaviour. Initial attempts to that end were made in [20,21]. In the coalgebraic world, much work on composing behaviours has been done in relation with modal logics [7].

References

[1] Aceto, L., W. J. Fokkink and C. Verhoef, *Structural operational semantics*, in: J. A. Bergstra, A. Ponse and S. Smolka, editors, *Handbook of Process Algebra*, Elsevier, 2002 pp. 197–292.

[2] Adámek, J., H. Herrlich and G. E. Strecker, "Abstract and Concrete Categories," 2004, available from `http://katmat.math.uni-bremen.de/acc`.

[3] Bartels, F., *GSOS for probabilistic transition systems*, in: *Proc. CMCS'02*, ENTCS **65** (2002), pp. 29–54.

[4] Bartels, F., "On Generalised Coinduction and Probabilistic Specification Formats," PhD dissertation, CWI, Amsterdam (2004).

[5] Bergstra, J. A., A. Ponse and S. Smolka, "Handbook of Process Algebra," Elsevier, 2002.

[6] Bloom, B., S. Istrail and A. Meyer, *Bisimulation can't be traced*, Journal of the ACM **42** (1995), pp. 232–268.

[7] Cirstea, C. and D. Pattinson, *Modular construction of modal logic*, in: *Proc. CONCUR'04*, LNCS **3170** (2004), pp. 258–275.

[8] Fiore, M. and S. Staton, *A congruence rule format for name-passing process calculi from mathematical structural operational semantics*, in: *Proc. LICS'06*, IEEE Computer Society Press, 2006 pp. 49–58.

[9] Fiore, M. P., G. D. Plotkin and D. Turi, *Abstract syntax with variable binding*, in: *Proc. LICS'99*, IEEE Computer Society Press, 1999 pp. 193–202.

[10] Fiore, M. P. and D. Turi, *Semantics of name and value passing*, in: *Proc. LICS'01*, IEEE Computer Society Press, 2001 pp. 93–104.

[11] Fokkink, W. and R. van Glabbeek, *Ntyft/ntyxt rules reduce to ntree rules*, Information and Computation **126** (1996), pp. 1–10.

[12] Fokkink, W. J., R. J. van Glabbeek and P. de Wind, *Compositionality of Hennessy-Milner logic through structural operational semantics*, in: *Proc. FCT'03*, LNCS **2751** (2003), pp. 412–422.

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

[14] Groote, J. F., M. Mousavi and M. A. Reniers, *A hierarchy of SOS rule formats*, in: *Proc. SOS'05*, 2005 (2005), pp. 3–25.

[15] Jacobs, B., *Towards a duality result in the modal logic for coalgebras*, in: *Proc. CMCS 2000*, ENTCS **33** (2000), pp. 160–195.

[16] Jacobs, B., *Distributive laws for the coinductive solution of recursive equations*, Information and Computation **204** (2006), pp. 561–587.

[17] Jacobs, B. and J. J. M. M. Rutten, *A tutorial on (co)algebras and (co)induction*, Bulletin of the EATCS **62** (1996).

[18] Kick, M., *Bialgebraic modelling of timed processes*, in: *Proc. ICALP'02*, LNCS **2380** (2002), pp. 525–536.

[19] Kick, M., *Rule formats for timed processes*, in: *Proc. CMCIM'02*, ENTCS **68** (2002), pp. 12–31.

[20] Kick, M. and J. Power, *Modularity of behaviours for mathematical operational semantics*, in: *Procs. CMCS'04*, ENTCS **106** (2004), pp. 185–200.

[21] Kick, M., J. Power and A. Simpson, *Coalgebraic semantics for timed processes*, Information and Computation **204** (2006), pp. 588–609.

[22] Klin, B., "Abstract Coalgebraic Approach to Process Equivalence for Well-Behaved Operational Semantics," Ph.D. thesis, BRICS, Aarhus University (2004).

[23] Klin, B., *Adding recursive constructs to bialgebraic semantics*, Journal of Logic and Algebraic Programming **60-61** (2004), pp. 259–286.

[24] Klin, B., *From bialgebraic semantics to congruence formats*, in: *Proc. SOS 2004*, ENTCS **128** (2005), pp. 3–37.

[25] Klin, B. and P. Sobocinski, *Syntactic formats for free: An abstract approach to process equivalence*, in: *Proc. CONCUR 2003*, LNCS **2761** (2003), pp. 72–86.

[26] Kurz, A., *Coalgebras and their logics*, ACM SIGACT News **37** (2006).

[27] Leifer, J. and R. Milner, *Deriving bisimulation congruences for reactive systems*, in: *Proc. CONCUR 2000*, LNCS **1877** (2000), pp. 243–258.

[28] Lenisa, M., J. Power and H. Watanabe, *Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads*, in: *Proc. CMCS'00*, ENTCS **33**, Elsevier, 2000 pp. 230–260.

[29] Lenisa, M., J. Power and H. Watanabe, *Category theory for operational semantics*, Theoretical Computer Science **327** (2004), pp. 135–154.

[30] Mac Lane, S., "Categories for the Working Mathematician," Springer, 1998, second edition.

[31] Milner, R. and M. Tofte, "The Definition of Standard ML," MIT Press, 1997, revised edition.

[32] Mosses, P. D., *Modular structural operational semantics*, Journal of Logic and Algebraic Programming **60-61** (2004), pp. 195–228.

[33] Mousavi, M. R. and M. A. Reniers, *Congruence for structural congruences*, in: *Proc. FOSSACS'05*, LNCS **3441**, 2005, pp. 47–62.

[34] Pavlovic, D., M. Mislove and J. B. Worrell, *Testing semantics: connecting processes and process logics*, in: *Proc. AMAST'05*, LNCS **4019** (2005), pp. 308–322.

[35] Plotkin, G. D., *The origins of structural operational semantics*, Journal of Logic and Algebraic Programming **60-61** (2004), pp. 3–15.

[36] Plotkin, G. D., *A structural approach to operational semantics*, Journal of Logic and Algebraic Programming **60-61** (2004), pp. 17–139.

[37] Power, J. and H. Watanabe, *Distributivity for a monad and a comonad*, in: *Procs. CMCS'99*, ENTCS **19** (1999), p. 102.

[38] Rutten, J. J. M. M., *Universal coalgebra: a theory of systems*, Theoretical Computer Science **249** (2000), pp. 3–80.

[39] Sassone, V. and P. Sobocinski, *Locating reaction with 2-categories*, Theoretical Computer Science **333** (2003), pp. 297–327.

[40] Turi, D. and G. D. Plotkin, *Towards a mathematical operational semantics*, in: *Proc. LICS'97*, IEEE Computer Society Press, 1997 pp. 280–291.

[41] Watanabe, H., *Well-behaved translations between structural operational semantics*, ENTCS **65** (2002).