A Relational Model of Non-Deterministic Dataflow*

Thomas Hildebrandt[1], Prakash Panangaden[2], and Glynn Winskel[1]

[1] BRICS***, University of Aarhus, Denmark,
{hilde,gwinskel}@brics.dk,
[2] McGill University, Montreal, Canada,
prakash@cs.mcgill.ca

Abstract. We recast dataflow in a modern categorical light using profunctors as a generalisation of relations. The well known causal anomalies associated with relational semantics of indeterminate dataflow are avoided, but still we preserve much of the intuitions of a relational model. The development fits with the view of categories of models for concurrency and the general treatment of bisimulation they provide. In particular it fits with the recent categorical formulation of feedback using traced monoidal categories. The payoffs are: (1) explicit relations to existing models and semantics, especially the usual axioms of monotone IO automata are read off from the definition of profunctors, (2) a new definition of bisimulation for dataflow, the proof of the congruence of which benefits from the preservation properties associated with open maps and (3) a treatment of higher-order dataflow as a biproduct, essentially by following the geometry of interaction programme.

1 Introduction

A fundamental dichotomy in concurrency is the distinction between *asynchronous* communication and *synchronous* communication. In the present paper we unify the analysis of these situations in the framework of a categorical presentation of models for concurrency as initiated by Winskel and Nielsen [36]. In particular we have given a treatment of indeterminate dataflow networks in terms of (a special kind of) profunctors which is very close to the treatment of synchronous communication. This new semantical treatment has a number of benefits

1. the general functoriality and naturality properties of presheaves *automatically* imply the usually postulated axioms for asynchronous, monotone automata [27, 32]
2. we get a notion of bisimulation, which is crucial when one includes both synchronous and asynchronous primitives together,
3. it is closely connected to the extant models [15] expressed in terms of trace sets, but also provides a relational viewpoint which allows one to think of composing network components as a (kind of) relational composition,
4. gives a semantics of higher-order networks almost for "free" by using the passage from traced monoidal categories to compact-closed categories [2, 18] (the "geometry of interaction" construction).

* To appear in LNCS, proceedings of CONCUR '98,

*** **B**asic **R**esearch **i**n **C**omputer **S**cience, Centre of the Danish National Research Foundation.

The categorical presentation is critical for all these points. Without the realization that Kahn processes can be described as a traced monoidal category and knowledge of the results in [2, 18] it would be hard to see how one could have proposed our model of higher-order processes. It is notable that the profunctor semantics of dataflow yields automatically the axioms for monotone port automata used in modelling dataflow [27] in contrast to the work in [33]. At the same time we have to work to get a correct operation on profunctors to model the dataflow feedback; "the obvious" choice of modelling feedback by coend doesn't account for the subtle causal constraints which plague dataflow semantics.

The background for our paper includes work done on presenting models for concurrency as categories, as summarised in [36]. This enabled a sweeping definition of bisimulation based on open maps applicable to any category of models equipped with a distinguished subcategory of paths [17]. It also exposed a new space of models: presheaves. Presheaf categories possess a canonical choice of open maps and bisimulation, and can themselves be related in the bicategory of profunctors. This yields a form of domain theory but boosted to the level of using categories rather than partial orders as the appropriate domains.

One argument for the definition of bisimulation based on open maps is the powerful preservation properties associated with it. Notable is the result of [7] that any colimit preserving functor between presheaf categories preserves bisimulation, which besides obvious uses in relating semantics in different models with different notions of bisimulation is, along with several other general results, useful in establishing congruence properties of process languages. By understanding dataflow in terms of profunctors we are able to exploit the framework not just to give a definition of bisimulation between dataflow networks but also in showing it to be a congruence with respect to the standard operations of dataflow.

A difficulty has been in understanding the operational significance of the bisimulation which comes from open maps for higher-order process languages (where for example processes themselves can be passed as values). Another gap, more open and so more difficult to approach, is that whereas both interleaving models and independence models like event structures can be recast as presheaf models, as soon as higher-order features appear, the presheaf semantics at present reduce concurrency to nondeterministic interleaving. A study of nondeterministic dataflow is helpful here as its compositional models are forced to account for causal dependency using ideas familiar from independence models; at the same time the models are a step towards understanding higher-order as they represent nondeterministic functions from input to output.

The idea that non-deterministic dataflow can be modelled by some kind of generalised relations fits with that of others, notably Stark in [33, 34]. Bisimulation for dataflow is studied in [35]. That dataflow should fit within a categorical account of feedback accords for instance with [21, 2]. But in presenting a semantics of dataflow as profunctors we obtain the benefits to be had from placing nondeterministic dataflow centrally within categories of models for concurrency, and in particular within presheaf models. One of our future aims is a dataflow semantics of hardware-description languages, like for instance Verilog HDL [12], which presently only possesses a non-compositional,

operational definition. The semantics of a language of this richness requires a flexible yet abstract domain theory of the kind presheaf models seem able to support.

2 Models for Indeterminate Dataflow

The Dataflow paradigm for parallel computation, originated in work of Jack Dennis and others in the mid-sixties [19, 9]. The essential idea is that data flows between asynchronous computing agents, that are interconnected by communication channels acting as unbounded buffers. Traditionally, the *observable behaviour* is taken to be the *input-output* relation between sequences of values on respectively input and output channels, sometimes referred to as the *history model* [15]. For dataflow networks built from only *deterministic* nodes, Kahn [19] has observed that their behaviour could be captured *denotationally* in the history model, defining network composition by the least fixed point of a set of equations describing the components, which was later shown formally by several authors, e.g. Faustini [11], Lynch and Stark [23]. Subsequently, different semantics have been described as satisfying *Kahn's principle* when they are built up compositionally along similar lines.

2.1 The Need for Causality

For *indeterminate* networks, the situation is not so simple. Brock and Ackerman[6] showed the fact, referred to as the "Brock-Ackerman anomaly", that for networks containing the nondeterministic primitive *fair merge*, the history model preserves too little information about the structure of the networks to support a compositional semantics. Later, Trakhtenbrot and Rabinovich, and independently, Russell gave examples of anomalies showing that this is true even for the simplest nondeterministic primitive[1] the ordinary *bounded choice*.

We present a similar example to illustrate what additional information is needed. It works by giving two simple examples of automata $\mathcal{A}_1$ and $\mathcal{A}_2$, which have the same input-output relation, and a context $\mathcal{C}[-]$ as shown in the figure, in which they behave differently. The context consists of a fork process $\mathcal{F}$ (a process that copies every input to two outputs), through which the output of the automata $\mathcal{A}_i$ is fed back to the input channel. Automaton $\mathcal{A}_1$ has the following (deterministic) behaviour: It outputs a token; waits for a token on input and then outputs another token. Automaton $\mathcal{A}_2$ has the choice between two behaviours: Either it outputs a token and stops, *or* it waits for an input token, then outputs two tokens. For both automata, the IO-relation relates empty input to zero or one output token, and non-empty input to zero, one or two output tokens, but $\mathcal{C}[\mathcal{A}_1]$ can output two tokens, whereas $\mathcal{C}[\mathcal{A}_2]$ can only output a single token, choosing the first behaviour of $\mathcal{A}_2$. This example shows the need for a model that records a more detailed causality relation between individual data tokens than the history model.

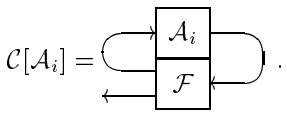


Fig. 1. The automata $\mathcal{A}_i$ inserted in context $\mathcal{C}[-]$

[1] See [27, 26] for a study of the differences between the nondeterminate primitives.

Jonsson [15] and Kok [22] have independently given fully abstract models for indeterminate dataflow. Jonsson's model is based on trace[2] sets, which are sets of possible interaction sequences, finite or infinite, between a process and its environment. Kok's model turned out to be equivalent. Rabinovich and Traktenbrot analyzed the same issues from the point of view of finite observations and came up with general conditions under which a Kahn-like principle would hold [30, 29, 31]. Abramsky has generalised Kahn's principle to indeterminate networks [1].

3 A Traced Monoidal Category of Kahn Processes

In this section we summarize the basic theory of traced monoidal categories and describe a category of Kahn processes as an instance of a traced monoidal category. The notion of traced monoidal category abstracts the notion of trace of a matrix from multilinear algebra. However it has emerged in a variety of new contexts including the study of feedback systems [4], knot theory [14] and recursion [13]. The axiomatization presented below is the definition of Joyal, Street and Verity [18], slightly simplified as in [13] and specialized to the context of symmetric monoidal categories so that the axioms appear simpler; in particular we do not consider braiding or twists. In the Joyal, Street and Verity paper the fact that trace models feedback (or iteration) is attributed to Bloom, but as far back as 25 years ago Bainbridge had been studying trace in the context of feedback in systems and control theory. Indeed Bainbridge had noticed that there were two kinds of trace (associated with two different monoidal structures) in Rel, the category of sets and binary relations. Furthermore he noted that one of the traces corresponds to feedback in what are essentially memoryless Kahn networks. [3]

3.1 Traced Monoidal Categories

In this section we give the axioms for a symmetric monoidal category equipped with a trace. We assume that the reader is familiar with the notion of a symmetric tensor product. We write $\otimes$ for the tensor product and $\sigma_{XY} : X \otimes Y \to Y \otimes X$ for the natural isomorphism (the symmetry) in this case.

Definition 1. *A **trace** for a symmetric monoidal category $\mathcal{C}$ is a* natural *family of functions*

$$Tr^U_{X,Y}() : \mathcal{C}(X \otimes U, Y \otimes U) \to \mathcal{C}(X, Y)$$

satisfying the following conditions

1. Bekic*: $f : X \otimes U \otimes V \to Y \otimes U \otimes V$ and $g : X \to Y$*

$$Tr^{U \otimes V}_{X,Y}(f) = Tr^U_{X,Y}(Tr^V_{X \otimes U, Y \otimes U}(f)) \quad \textit{and } Tr^I_{X,Y}(g) = g \ .$$

[2] This word commonly used in the literature unfortunately clashes with "trace" in linear algebra. Normally this is not a problem but the present paper uses this word in both senses, we hope the reader will be able to disambiguate from the context.

[3] We are indebted to Samson Abramsky for pointing this reference out to us.

2. Yanking: $Tr^U_{U,U}(\sigma_{UU}) = I_U$.
3. Superposing: *Given* $f\colon X \otimes U \to Y \otimes U$

$$Tr^U_{Z\otimes X, Z\otimes Y}(I_Z \otimes f) = I_Z \otimes Tr^U_{X,Y}(f) \ .$$

The following proposition is an easy consequence of the definitions. It shows how composition can be defined from trace and tensor.

Proposition 1. *Given* $g\colon U \to Y$ *and* $f\colon X \to U$ *we have*

$$Tr^U_{X,Y}(\sigma_{UY} \circ (f \otimes g)) = g \circ f \ .$$

This could be viewed as a generalisation of the yanking condition.

It is instructive to consider the two well-known examples of trace in Rel. In the first case one takes the tensor product to be the cartesian product of the underlying sets and in the second case one takes the tensor product to be disjoint union of sets (with the evident action on relations); we call these structures $(\mathsf{Rel}, \times)$ and $(\mathsf{Rel}, +)$ respectively. The trace in $(\mathsf{Rel}, \times)$ is given by

$$Tr^U_{X,Y}(R)(x, y) = \exists u \in U.R(x, u, y, u) \ ,$$

where R is a binary relation from $X \times U$ to $Y \times U$. This is very close to the trace in linear algebra - the sum along the diagonal - here it is an existential quantification along the "diagonal." Now for the other structure one proceeds as follows. Let R be a binary relation from $X \uplus U$ to $Y \uplus U$. This can be seen as consisting of 4 pieces, namely the relations R_{XY}, R_{XU}, R_{UY} and R_{UU}. For example we say that $R_{XY}(x, y)$ holds for $x \in X, y \in Y$ iff $R(x, y)$ holds. Now the trace is given by

$$Tr^U_{X,Y}(R) = R_{XY} \cup R_{XU}; R^*_{UU}; R_{UY} \ ,$$

where we are using the standard relational algebra concepts; $*$ for reflexive, transitive closure, ; for relational composition and $\cup$ for union of the sets of pairs in the relation. Intuitively this is the formula expressing feedback: either x and y are directly related or x is related to some u and that u is related to y (once around the feedback loop) or, more generally, we can go around the "feedback loop" an indefinite number of times.

3.2 The Kahn Category

The basic intuitions behind Kahn networks are, of course, due to Kahn [19] and a formal operational semantics in terms of coroutines is due to Kahn and McQueen [20]. The particular axiomatisation presented here builds on the ideas of Stark [33] but using the formalism of traces presented in [26]. No originality is claimed for the trace model, it was Bengt Jonsson [15] who showed that traces form a fully abstract model of dataflow networks and there were several others with similar ideas at the time.

We have a fixed set $\mathcal{V}$ of *values* and a fixed set $\mathcal{P}$ of *ports*. An *event* is a triple $\langle a, i/o, v\rangle$ where $a \in \mathcal{P}$ and $v \in \mathcal{V}$. We say that $\langle a, v\rangle$ is the *label* of the event $\langle a, i/o, v\rangle$. An event of the form $\langle a, o, v\rangle$ is called an *output* event and one of the form

$\langle a, i, v\rangle$ is called an *input* event. We consider sequences of these events. If α is a sequence of events we write $l(\alpha)$ for the sequence of labels obtained by discarding the input/output tags. We write $\alpha|_o$ (or $\alpha|_i$) for the sequence of output (or input) events discarding the input (or output) events. We write $\alpha|_A$ for the sequence obtained by keeping only the events on the ports in A and $\alpha|_{A_o}$ (or $\alpha|_{A_i}$) for the sequence of output (or input) events on ports in A. We extend these notations to sets of sequences. We use the notation $\alpha \leq \beta$ for the prefix order on sequences. We write $\mathcal{I}_A$ for the set $A \times \{i\} \times \mathcal{V}$ of all input events on ports in A and similarly $\mathcal{O}_A$ for $A \times \{o\} \times \mathcal{V}$. Finally, we write $\mathcal{L}_A$ for the set $A \times \mathcal{V}$ of labels on ports in A.

Definition 2. *A **process** of **sort** (A, B), where $A, B \subseteq \mathcal{P}$ is a non-empty prefix closed set of finite sequences over the alphabet $\mathcal{I}_A \cup \mathcal{O}_B$. The set of sequences, say S, satisfies the following closure properties, α and β are sequences of events:*

K1. *If $\alpha\langle b, o, v\rangle\langle a, i, u\rangle\beta \in S$ then $\alpha\langle a, i, u\rangle\langle b, o, v\rangle\beta \in S$.*
K2. *If $\alpha\langle b, o, v\rangle\langle b', o, u\rangle\beta \in S$ and if $b \neq b'$ then $\alpha\langle b', o, u\rangle\langle b, o, v\rangle\beta \in S$.*
K3. *If $\alpha\langle a, i, u\rangle\langle a', i, v\rangle\beta \in S$ and if $a \neq a'$ then $\alpha\langle a', i, v\rangle\langle a, i, u\rangle\beta \in S$.*
K4. *If $\alpha \in S$ then $\alpha\langle a, i, v\rangle$ for all $\langle a, v\rangle \in \mathcal{L}_A$.*

*We call A the **input ports** and B the **output ports** of the process.*

The last condition above is called *receptivity*, a process could receive any data on its input ports; unlike with synchronous processes. Receptivity is the basic reason why traces suffice to give a fully-abstract model for asynchronous processes; in calculi with synchronous communication one needs branching information.

The first three conditions express concurrency conditions on events occurring at different ports. Note an asymmetry in the first condition. If an output occurs before an input then it could also occur after the input instead. However, if an output occurs after an input then the pair of events cannot be permuted because the output event may be in response to the input. Furthermore we are assuming, again in (1), that the arrival of input does not disable already enabled output. In an earlier investigation [27] these were called *monotone* automata and it was shown that many common primitives, such as fair merge, timeouts, interrupts and polling cannot be expressed as monotone automata. However, as we will see in the end of this section, monotonicity and receptivity together imply that the IO-relations of Kahn processes are *buffered* in a formal sense. This makes them reasonable assumptions for the type of networks we consider. The restriction to finite sequences is a simplification, which is not necessary for the results in this section, but simplifies the exposition in the following section. In the model of Jonsson infinite sequences are used to express fairness properties.

Given processes as sets of sequences we define *composition* as follows. We begin by defining the shuffle of two sets of sequences.

Definition 3. *Given two sets of sequences of events, say S of sort (A, B) and S' of sort (A', B'), with $A \cap A' = \varnothing = B \cap B'$, we define the set $S \Delta S'$ (read, S **shuffle** S') as the set of all sequences γ of sort $(A \cup A', B \cup B')$ s.t. $\gamma|_{A_i \cup B_o} \in S$ and $\gamma|_{A'_i \cup B'_o} \in S'$.*

We then define composition, by picking from the shuffle the sequences having the right causal precedence of events on B and then discarding these, now "internal", events.

Definition 4. *Given processes $f\colon A \to B$ and $g\colon B \to C$ we define the composite of f and g by $f;g = S|_{A_i \cup C_o}$, where $S \subseteq f \Delta g$ (with ports renamed if necessary to avoid name clashes) is the the largest set s.t. $\forall \delta \in S.\ l(\delta|_{B_o}) = l(\delta|_{B_i})$ & $\forall \delta' \leq \delta.\ l(\delta'|_{B_i}) \leq l(\delta'|_{B_o})$.*

Definition 5. *The category* Kahn *of Kahn processes has as objects finite subsets of $\mathcal{P}$ and as morphisms from A to B, processes with A as the input ports and B as the output ports. Composition of morphisms is defined by composition of processes as defined above.*

A monoidal structure is given by disjoint union on objects and for $f\colon A \to B$ and $f'\colon A' \to B'$, $f \otimes f'\colon A \uplus A' \to B \uplus B'$ is given by $f \Delta g$. The trace construction is as follows. Given $f\colon X \uplus U \to Y \uplus U$ we define $Tr^U_{X,Y}(f)\colon X \to Y$ as the set of all sequences γ such that there is a sequence $\delta \in f$ with

1. $\delta|_{X_i \cup Y_o} = \gamma$,
2. $l(\delta|_{U_o}) = l(\delta|_{U_i})$ and
3. $\forall \delta' \leq \delta.\ l(\delta'|_{U_i}) \leq l(\delta'|_{U_o})$.

Theorem 1. *With the structures given above,* Kahn *is a traced monoidal category.*

The generalised yanking property can be interpreted in this category as saying that composition can be obtained as a combination of parallel composition (that is, shuffling) and feedback. This is a well-known fact in dataflow folklore.

3.3 From Kahn Processes to Input-output Relations

The category of Kahn processes can be related to the history model by a functor to a category of buffered relations between histories. Given a set of portnames $A \subseteq \mathcal{P}$ let $\bar{A}$, the *histories* on A, be the elements in the free partially commutative monoid $\mathcal{L}^*_A/\eqsim$ [10], where $\eqsim$ is the smallest equivalence relation such that $\alpha\langle a', v'\rangle\langle a, v\rangle\beta \eqsim \alpha\langle a, v\rangle\langle a', v'\rangle\beta$ if $a \neq a'$. We will refer to the elements as *traces* and for a sequence $\alpha \in \mathcal{L}^*_A$ let $\overline{\alpha}$ denote its trace. The traces $\bar{A}$ can be partial ordered by $\overline{\alpha} \sqsubseteq \overline{\beta}$ iff $\exists \gamma.\overline{\alpha\gamma} = \overline{\beta}$ (see Sect.7 of [36]). Let ϵ_A (or just ϵ) denote the empty trace in $\bar{A}$. Given the ordering on traces, we can define a *buffer* $B_A \subseteq \bar{A} \times \bar{A}$ for each port set $A \subseteq \mathcal{P}$, as the relation $\{(\overline{\alpha}, \overline{\beta}) \mid \overline{\beta} \sqsubseteq \overline{\alpha}\}$. Inspired by the work in [32], we say that an IO-relation $R \subseteq \bar{A} \times \bar{B}$ is *buffered* if it satisfies that $R = B_A; R; B_B$, i.e. adding buffers to input and output makes no difference. From the fact that $B_A; B_A = B_A$ for any $A \subseteq \mathcal{P}$ it follows that we can define a category Hist of buffered IO-relations, in which the identities are given by the buffers.

Definition 6. *Let* Hist *be the category with objects $\bar{A}$ for $A \subseteq \mathcal{P}$, and morphisms being buffered IO-relations. A symmetric monoidal structure is given as in* $(\mathsf{Rel}, \times)$.

A Kahn process S defines naturally a relation between input and output *sequences* by considering the set $\mathcal{H}S = \{(\gamma, \delta) \mid \exists \alpha \in S.\ \gamma = l(\alpha|_{A_i})\ \&\ \delta = l(\alpha|_{B_o})\}$. It can be shown that $\mathcal{H}$ extends to a buffered relation between traces and preserves composition.

Theorem 2. *There is a monoidal functor $\mathcal{H}\colon$ Kahn $\to$ Hist that maps port sets $A \subseteq \mathcal{P}$ to $\bar{A}$ and Kahn processes S of sort (A, B) to the relation $\{(\overline{\gamma}, \overline{\delta}) \mid \exists \alpha \in S.\ \gamma = l(\alpha|_{A_i})\ \&\ \delta = l(\alpha|_{B_o})\}$.*

4 Generalising Relations

Kahn processes are typical of the solutions to the problem of obtaining a compositional semantics for nondeterministic dataflow. A correct compositional semantics is got by representing processes as interaction sequences, keeping track of the causal dependency between events. However, this depends on networks being built purely by asynchronous primitives and seems far removed from the relational model. In this section we will describe another solution that *contains* the Kahn processes, but allows other, e.g. synchronous network primitives, and comes about as a natural (categorical) extension of the history model. Let $\overline{\mathsf{A}}$ denote the partial order category given by $\bar{A}$ and the ordering $\sqsubseteq$. If $\overline{\alpha} \sqsubseteq \overline{\gamma}$, let $[\overline{\alpha}, \overline{\gamma}] \colon \overline{\alpha} \to \overline{\gamma}$ denote the unique arrow in $\overline{\mathsf{A}}$. We will refer to these categories as the *path categories*.[4]

The key observation is that buffered IO-relations correspond exactly to functors $\overline{\mathsf{A}} \times \overline{\mathsf{B}}^{op} \to \mathbf{2}$, where $\mathbf{2}$ is the category consisting of two objects 0 and 1 and only one non-identity arrow $0 \to 1$. This is an immediate categorical analogy to characteristic functions $\bar{A} \times \bar{B} \to \{0, 1\}$ of relations. Viewing the relations in this way, composition of $R \colon \overline{\mathsf{A}} \times \overline{\mathsf{B}}^{op} \to \mathbf{2}$ and $R' \colon \overline{\mathsf{B}} \times \overline{\mathsf{C}}^{op} \to \mathbf{2}$ can be written as

$$R; R'(\overline{\alpha}, \overline{\gamma}) = \bigvee_{\overline{\beta} \in \overline{\mathsf{B}}} R(\overline{\alpha}, \overline{\beta}) \wedge R'(\overline{\beta}, \overline{\gamma}) \ , \tag{1}$$

where we make use of the obvious join and meet operations on $\mathbf{2}$.

This defines a category BRel of buffered relations, with path categories as objects, arrows being relations and composition as defined above. As stated below, BRel is just an alternative definition of the category Hist given in the previous section.

Proposition 2. *The category* Hist *is equivalent to the category* BRel.

Note that the anomaly given in Sec. 2.1 shows that there is no way of defining a trace on BRel such that the functor $\mathcal{H}$ given in the last section preserves the trace of Kahn. It must be possible to represent *different* dependencies between input and output for a particular input-output pair in the relation. This is precisely what moving to the bicategory of *profunctors* does for us.

4.1 Profunctors

The category Prof of profunctors, (or bimodules, or distributors [5]) are a categorical generalisation of sets and relations. The objects of Prof are small categories and arrows are profunctors; profunctors are like the buffered relations above but with the category $\mathbf{2}$ replaced by Set.

Definition 7. *Let* P *and* Q *be small categories. A profunctor* $X \colon \mathsf{P} \mathrel{-\!\!\!\mapsto} \mathsf{Q}$ *is a bifunctor* $X \colon \mathsf{P} \times \mathsf{Q}^{\mathsf{op}} \to \mathsf{Set}$ *(or equivalently, a presheaf in* $\widehat{\mathsf{P}^{\mathsf{op}} \times \mathsf{Q}}$*).*

The tensor product is given by the product of categories on objects and set-theoretic product on arrows. This defines a symmetric monoidal structure on Prof.

[4] The traces can also be viewed as a specific kind of pomsets [28] and the path categories as a subcategory of the category of pomsets given in [17].

Definition 8. *Let* P,P' *and* Q,Q' *be small categories and* $X\colon \mathsf{P}\mathrel{-\!\!\!\mapsto}\mathsf{Q}$, $Y\colon \mathsf{P}'\mathrel{-\!\!\!\mapsto}\mathsf{Q}'$ *profunctors. Define* $\mathsf{P}\otimes\mathsf{P}' = \mathsf{P}\times\mathsf{P}'$ *and* $X\otimes Y = X\times Y\colon \mathsf{P}\otimes\mathsf{P}'\mathrel{-\!\!\!\mapsto}\mathsf{Q}\otimes\mathsf{Q}'$*, so* $(X\otimes Y)(p,p',q,q') = X(p,q)\times Y(p',q')$ *(and similarly for morphisms).*

Composition[5] of profunctors is given by the *coend* [25]

$$Y;Z(p,q) = \int^{u} Y(p,u)\times Z(u,q) \ , \tag{2}$$

for $Y\colon \mathsf{P}\mathrel{-\!\!\!\mapsto}\mathsf{U}$ and $Z\colon \mathsf{U}\mathrel{-\!\!\!\mapsto}\mathsf{Q}$. This generalises the expression for relational composition given by (1) earlier. Identities are given by hom functors. The obvious choice of trace on Prof is to take the trace of a profunctor $X\colon \mathsf{P}\otimes\mathsf{U}\mathrel{-\!\!\!\mapsto}\mathsf{Q}\otimes\mathsf{U}$ to be given by

$$Tr^{U}_{P,Q}(X)(p,q) = \int^{u} X(p,u,q,u) \ , \tag{3}$$

which satisfies the properties of a trace (up to isomorphism). Since we are working with functors into Set, the coend has an explicit definition. For p,q objects of respectively P and Q, we have

$$\int^{u} X(p,u,q,u) \cong \biguplus_{u\in\mathsf{U}} \{\, x\in X(p,u,q,u)\,\}_{/\sim} \ , \tag{4}$$

where $\sim$ is the symmetric, transitive closure of the relation $\rightsquigarrow$ defined as follows. For $x\in X(p,u,q,u)$ and $x'\in X(p,u',q,u')$, let $x\rightsquigarrow x'$ if $\exists m\colon u\to u'$ and $y\in X(p,u,q,u')$ such that $X(p,u,q,m)y = x$ and $X(p,m,q,u')y = x'$.

For our purpose, we focus on the subcategory PProf of Prof induced by the path categories, which generalises the buffered IO-relations. The category PProf, of *port profunctors*, inherits the traced symmetric monoidal structure of Prof, since we have $\overline{\mathsf{A}}\otimes\overline{\mathsf{B}} = \overline{\mathsf{A}}\times\overline{\mathsf{B}} \cong \overline{\mathsf{A}\uplus\mathsf{B}}$. The unit of the tensor is $I = \overline{\varnothing}$. It can be shown that the category BRel is a reflective sub traced monoidal category of PProf, which says that PProf is a direct categorical generalisation of buffered IO-relations. However, it also reveals rather quickly that the trace in PProf as given by the coend does not capture the *causality constraint* that a token must appear as output before it appears as input on the feedback channels, as stated in the third requirement of the trace in Kahn. For this reason taking trace as coend fails to express feedback. This is illustrated by the fork process $\mathcal{F}\colon \overline{\mathsf{A}}\mathrel{-\!\!\!\mapsto}\overline{\mathsf{A}}\otimes\overline{\mathsf{A}}$ used in the example of Sect. 2.1. The process $\mathcal{F}$ is just a buffer that has two output channels for each input channel. Connecting one set of outputs to the inputs should result in a process of sort $(\varnothing, A)$ that can output nothing but the empty trace, which is indeed what one gets in Kahn. To see what the coend formula gives, note that $\mathcal{F}$ can be regarded as the buffered relation $\{\,(\overline{\alpha},(\overline{\beta},\overline{\gamma}))\mid \overline{\beta}\sqsubseteq\overline{\alpha}\ \&\ \overline{\gamma}\sqsubseteq\overline{\alpha}\}$. When restricted to the category of buffered relations, the trace as given by the coend reduces to

$$Tr^{A}_{I,A}(\mathcal{F})(\overline{\beta}) = \bigvee_{\overline{\alpha}\in\overline{\mathsf{A}}} \mathcal{F}(\overline{\alpha},\overline{\beta},\overline{\alpha}) \ , \tag{5}$$

[5] This defines composition only to within isomorphism, explaining why we get a *bi*category.

which is simply the trace in $(\mathsf{Rel}, \times)$. Now, it is easy to see that $Tr^A_{I,A}(\mathcal{F})(\overline{\beta})$, for any $\overline{\beta} \in \overline{\mathsf{A}}$, i.e. the resulting process can output *any* sequence.

In the next section we shall see that it is possible to adopt the causal constraint on the trace in Kahn to the trace in profunctors.

4.2 An Operational Reading

Given a profunctor $X : \overline{\mathsf{A}} \mathrel{-\!\!\!\vdash\!\!\!\to} \overline{\mathsf{B}}$, the elements in $X(\overline{\alpha}, \overline{\beta})$ can be viewed as the possible states of a process in which it has read $\overline{\alpha}$ and written $\overline{\beta}$. For a port profunctor $X : \overline{\mathsf{A}} \mathrel{-\!\!\!\vdash\!\!\!\to} \overline{\mathsf{B}}$, we define its associated *port automaton* as follows.[6]

Definition 9. *Let $X : \overline{\mathsf{A}} \mathrel{-\!\!\!\vdash\!\!\!\to} \overline{\mathsf{B}}$ be a port profunctor. Define its* associated (A, B)-port automaton $\mathcal{A}(X)$ *to be the quintuple $(S, R, \longrightarrow, A, B)$, where $S = \{\big((\overline{\alpha}, \overline{\beta}), x\big) \mid x \in X(\overline{\alpha}, \overline{\beta})\}$ is a set of* states, *$R = \{\big((\epsilon, \epsilon), x\big) \mid x \in X(\epsilon, \epsilon)\}$ is a set of* initial states, *A and B are sets of resp.* input ports *and* output ports, *and $\longrightarrow \subseteq S \times (\mathcal{I}_A \cup \mathcal{O}_B) \times S$ is a* transition relation, *given by*

- $\big((\overline{\alpha}, \overline{\beta}), x\big) \xrightarrow{\mathsf{i}\, a, v} \big((\overline{\alpha\langle a, v\rangle}, \overline{\beta}), y\big)$, *if* $X([\overline{\alpha}, \overline{\alpha\langle a, v\rangle}], \overline{\beta})x = y$,
- $\big((\overline{\alpha}, \overline{\beta}), x\big) \xrightarrow{\mathsf{o}\, b, v} \big((\overline{\alpha}, \overline{\beta\langle b, v\rangle}), y\big)$, *if* $X(\overline{\alpha}, [\overline{\beta\langle b, v\rangle}, \overline{\beta}])y = x$.

Define $Seq(X)$ to be the set of finite sequences of events labelling sequences of transitions of $\mathcal{A}(X)$ beginning at an initial state.

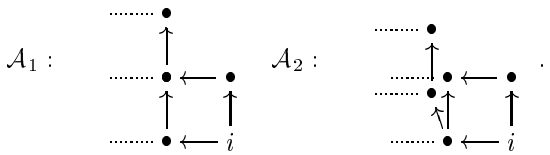


Fig. 2. Port automata of profunctors representing the processes given in Sect. 2.1. All vertical arrows are output transitions and horisontal arrows are input transitions. Note how the two runs in process $\mathcal{A}_2$, with same input-output relation but different dependencies, are represented

We can restore the *category of elements* of the presheaf X from its associated port automaton, which thus determines X up to isomorphism [24], allowing us to work with the more concrete representation when convenient. Thus, we can freely confuse elements $x \in X(\overline{\alpha}, \overline{\beta})$ with their corresponding states in $\mathcal{A}(X)$.

Remarkably the axioms usually postulated for monotone port automata [27] follow for port automata of profunctors simply by functoriality.

Proposition 3. *Let $X : \overline{\mathsf{A}} \mathrel{-\!\!\!\vdash\!\!\!\to} \overline{\mathsf{B}}$ and $\mathcal{A}(X) = (S, R, \longrightarrow, A, B)$. Then*

A1. *Receptivity:* $\forall \langle a, i, v\rangle \in \mathcal{I}_A \;\&\; s \in S.\; \exists! s' \in S.\; s \xrightarrow{\mathsf{i}\, a, v} s'$,

A2. *Monotonicity: If* $t \xrightarrow{\mathsf{o}\, b, v} s \;\&\; t \xrightarrow{\mathsf{i}\, a, v'} s'$ *then* $\exists! u \in S.\; s \xrightarrow{\mathsf{i}\, a, v'} u \;\&\; s' \xrightarrow{\mathsf{o}\, b, v} u$,

A3. *Commutativity: If* $c \neq c' \;\&\; s \xrightarrow{\mathsf{i}\, c, v} t \xrightarrow{\mathsf{i}\, c', v'} u$ *(or* $s \xrightarrow{\mathsf{o}\, c, v} t \xrightarrow{\mathsf{o}\, c', v'} u$*) then* $\exists! t' \in S.\; s \xrightarrow{\mathsf{i}\, c', v'} t' \xrightarrow{\mathsf{i}\, c, v} u$ *(or* $s \xrightarrow{\mathsf{o}\, c', v'} t' \xrightarrow{\mathsf{o}\, c, v} u$*)* .

As a corollary, we get a mapping from port profunctors to Kahn processes.

[6] The present construction generalises the one on presheaves given in [37]

Corollary 1. *For any $X : \overline{\mathsf{A}} \multimap\!\!\!\!\mid\!\!\rightarrow \overline{\mathsf{B}}$ s.t. $X(\epsilon,\epsilon) \neq \varnothing$, the set of sequences $Seq(X)$ is a Kahn process of sort (A, B).*

The above observations make port profunctors look promising as a model of dataflow. However, they are a bit too general for our purpose. We will restrict attention to *stable port profunctors*. These are the profunctors for which the associated port automaton satisfies the additional axioms

A4. *Stability: If $s \neq s'$, $s \xrightarrow{\mathsf{i}\, a,v} t$ & $s' \xrightarrow{\mathsf{i}\, a',v'} t$ then $a \neq a'$ & $\exists!u.\ u \xrightarrow{\mathsf{i}\, a',v'} s$ & $u \xrightarrow{\mathsf{i}\, a,v} s'$,*
A5. *Reachability: R is a singleton $\{r\}$ with $\forall s \in S.\ r \longrightarrow^* s$.*

Categorically, these axioms are equivalent to requiring that the profunctor (when regarded as a functor into sets) preserves pullbacks in its input argument and that for any $\overline{\alpha} \in \overline{\mathsf{A}}$, $X(\overline{\alpha}, \epsilon)$ is the singleton set. Stable port profunctors define a sub monoidal category of PProf, which we will refer to as $\mathsf{PProf}_\perp$. We will use the notation $X : \overline{\mathsf{A}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}}$ to indicate that X is a profunctor in $\mathsf{PProf}_\perp$. It can be shown that any Kahn process can be obtained from a stable port profunctor via the map Seq defined in Def. 9.

The relation $\rightsquigarrow$ defined in the explicit definition of the coend (4) can be interpreted as a relation between states connected by a chain of *internal communications* within a port automaton. More precisely, if $s = \big((\overline{\alpha}, \overline{\gamma}, \overline{\beta}, \overline{\gamma}), x\big)$ and $s' = \big((\overline{\alpha}, \overline{\gamma'}, \overline{\beta}, \overline{\gamma'}), x'\big)$ are states of a profunctor $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$, then it can be shown that $x \rightsquigarrow^* x'$ iff there exists a sequence of output-input pairs

$$s = s_0 \xrightarrow{\mathsf{o}\, \phi_0} t_0 \xrightarrow{\mathsf{i}\, \phi_0} s_1 \cdots s_n \xrightarrow{\mathsf{o}\, \phi_n} t_n \xrightarrow{\mathsf{i}\, \phi_n} s_{n+1} = s' \ ,$$

such that $\overline{\gamma \phi_0 \phi_1 \ldots \phi_n} = \overline{\gamma'}$, i.e. there is a sequence of transitions from s to s' in which the values ϕ_i is being fed back one by one. This leads to the following definition of *causally secured states.*

Definition 10. *Let $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$ and $\mathcal{A}(X) = (S, r, \longrightarrow, A \uplus C, B \uplus C)$. We say that $s \in S$ is* secured in $\overline{\mathsf{C}}$ *if there exists a sequence*

$$r \xrightarrow{\mathsf{i}\, \alpha_0} t_0 \xrightarrow{\mathsf{i}\, \alpha_1} t_1 \cdots \xrightarrow{\mathsf{i}\, \alpha_n} t_n \rightsquigarrow^* s_0 \xrightarrow{\mathsf{o}\, \beta_0} s_1 \xrightarrow{\mathsf{o}\, \beta_1} s_2 \cdots s_m \xrightarrow{\mathsf{o}\, \beta_m} s \ ,$$

where $\alpha_i \in \mathcal{L}_A$ and $\beta_j \in \mathcal{L}_B$.

The lemma below, which follows from stability, implies that each equivalence class of $\sim$ defined in (4) has a minimal state, from which any other state in the class is reachable by a chain of internal communications.

Lemma 1. *Let $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$ and $\mathcal{A}(X) = (S, r, \longrightarrow, A \uplus C, B \uplus C)$. Then any $\sim$-equivalence class is countable, and if $s \sim t$, for $s, t \in S$, then there exists a state $z \in S$ such that $z \rightsquigarrow^* s$ and $z \rightsquigarrow^* t$.*

This allows us to define a trace satisfying the causal constraints of feedback.

Definition 11. *Let $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$. Define $Tr^C_{A,B}(X) : \overline{\mathsf{A}} \multimap\!\!\!\!\perp\!\!\rightarrow \overline{\mathsf{B}}$, the trace of X to be given by*

$$Tr^C_{A,B}(X)(\overline{\alpha}, \overline{\beta}) \cong \biguplus_{\overline{\gamma} \in \overline{\mathsf{C}}} \{\, x \in X(\overline{\alpha}, \overline{\gamma}, \overline{\beta}, \overline{\gamma}) \mid x \text{ is secured in } \overline{\mathsf{C}} \}_{/\sim}, \tag{6}$$

where $\sim$ is defined as in (4) and the action on arrows is defined as for the coend.

In the case of composition, it can be shown that the securedness condition is satisfied for all states, so the secured trace coincides with the coend. This is the first step in showing that the trace as given by Def. 11 satisfies all the properties of a traced monoidal category. The proof of the Bekic property makes crucial use of the stability condition, indeed there exist a simple, non-stable port profunctor for which the Bekic property is not satisfied.

Theorem 3. *With the trace operator given in Def. 11,* $\mathsf{PProf}_\perp$ *is a traced monoidal category.*

The trace can be expressed on port automata as follows.

Proposition 4. *Let* $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$ *and* $\mathcal{A}(X) = (S, r, \longrightarrow, A \uplus C, B \uplus C)$*. Then* $\mathcal{A}(Tr^C_{A,B}(X)) = (S', [r]_\sim, \longrightarrow_\sim, A, B)$*, where* $\sim$ *is defined as in (4),* $S' = \{s \in S \mid s \text{ is secured in } \overline{\mathsf{C}}\}_{/\sim}$ *and* $[s]_\sim \xrightarrow{\mathsf{i}\, a,v}_\sim [s']_\sim$ *(or* $[s]_\sim \xrightarrow{\mathsf{o}\, b,v}_\sim [s']_\sim$*) if* $s \xrightarrow{\mathsf{i}\, a,v} s'$ *(or* $s \xrightarrow{\mathsf{o}\, b,v'} s'$*).*

There is not room here for a detailed discussion of the operational reading of the trace. However, it follows from Lem. 1 that the trace indeed has a reasonable operational interpretation. Moreover, the trace in $\mathsf{PProf}_\perp$ is consistent with the trace in Kahn.

Proposition 5. *The map* Seq *given in Def. 9 defines the action on arrows of a functor* $Seq : \mathsf{PProf}_\perp \to \mathsf{Kahn}$*, that preserves the traced monoidal structure, on objects simply mapping path categories to their underlying port set.*

We end this section with an important remark, namely that the secured trace can be defined as the composition of two functors on hom-categories; first a functor restricting to secured states and then a *colimit*, by using a standard construction of the *subdivision category* [25] which allows any coend to be expressed as a colimit. For a port profunctor $X : \overline{\mathsf{A}} \otimes \overline{\mathsf{C}} \multimap \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}$, we get

$$Tr^C_{A,B}(X) \cong \mathrm{Colim}_{\overline{\mathsf{C}}^\S} \mathcal{S}(X), \tag{7}$$

where $\overline{\mathsf{C}}^\S$ is the *subdivision category* of $\overline{\mathsf{C}}$ (dual to that in [25]) and $\mathcal{S} : \mathsf{PProf}_\perp[\overline{\mathsf{A}} \otimes \overline{\mathsf{C}}, \overline{\mathsf{B}} \otimes \overline{\mathsf{C}}] \to \mathsf{Prof}[\overline{\mathsf{A}} \otimes \overline{\mathsf{C}}^\S, \overline{\mathsf{B}}]$ is the standard construction, except it is restricted to secured states. Below we will benefit from the colimit formulation of the trace.

5 Some Consequences

We will briefly go through some of the consequences of the categorical semantics of dataflow given in the two previous sections.

5.1 A Bisimulation Congruence

The presentation of models for concurrency as categories allows us to apply a general notion of bisimulation from spans of open maps proposed in [17]. The general idea is to identify a *path category* $\mathsf{P} \hookrightarrow \mathsf{M}$ as a subcategory of the model M, with objects representing runs or histories and morphisms compatible extensions of these. A morphism

is then said to be P-*open* if it reflects extensions of histories, and two objects are said to be P-*bisimilar* if they are connected by a span of P-open maps. For a presheaf model $\hat{\mathsf{P}}$ the canonical choice of path category is the category P under the Yoneda embedding. Recall that a port profunctor $X : \overline{\mathsf{A}} \multimap \overline{\mathsf{B}}$ can be viewed as a presheaf in $\widehat{\overline{\mathsf{A}}^{op} \times \overline{\mathsf{B}}}$.

As for the presheaves as transition systems in [37], the $\overline{\mathsf{A}}^{op} \times \overline{\mathsf{B}}-$bisimulation can be characterised as a back-&-forth bisimulation between the associated port automata.

Proposition 6. *Let $X_i : \overline{\mathsf{A}} \multimap \overline{\mathsf{B}}$ and $\mathcal{A}(X_i) = (S_i, r_i, \longrightarrow_i, A, B)$ for $i \in \{1, 2\}$. X_1 and X_2 are $\overline{\mathsf{A}}^{op} \times \overline{\mathsf{B}}$-bisimilar iff $\mathcal{A}(X_1)$, $\mathcal{A}(X_2)$ are* back-&-forth bisimilar*: There exists a relation $R \subseteq S_1 \times S_2$ such that $(r_1, r_2) \in R$ and*

- $(s, s') \in R \;\&\; t \xrightarrow{\phi}_1 s \Rightarrow \exists t'.\ t' \xrightarrow{\phi}_2 s' \;\&\; (t, t') \in R$,
- $(s, s') \in R \;\&\; s \xrightarrow{\phi}_1 t \Rightarrow \exists t'.\ s' \xrightarrow{\phi}_2 t' \;\&\; (t, t') \in R$,
- $(s, s') \in R \;\&\; t' \xrightarrow{\phi}_2 s' \Rightarrow \exists t.\ t \xrightarrow{\phi}_1 s \;\&\; (t, t') \in R$,
- $(s, s') \in R \;\&\; s' \xrightarrow{\phi}_2 t' \Rightarrow \exists t.\ s \xrightarrow{\phi}_1 t \;\&\; (t, t') \in R$.

It is worth remarking, that the bisimulation is closely related to the *strong history-preserving bisimulation* obtained by the same approach for independence models in [17]. In particular, two bisimilar port automata will be strong history-preserving bisimilar for any fixed input sequence.

It is important to check that bisimulation on $\mathsf{PProf}_\perp$ is a congruence with respect to the operations tensor and trace. Here we can exploit some general properties of open maps and so bisimulation on presheaves: the product of (surjective) open maps in a presheaf category is (surjective) open [16]; any colimit-preserving functor between presheaf categories preserves (surjective) open maps [7]. The proof that trace on $\mathsf{PProf}_\perp$ preserves bisimulation uses the latter property, exploiting the fact that trace can be expressed as a colimit, first showing that $\mathcal{S}$ as a functor between presheaf categories preserves open maps. The proof of the corresponding result for tensor rests on a construction of tensor from more basic functors, which are all colimit-preserving and so preserve (surjective) open maps.

By placing dataflow within profunctors and the broader class of presheaf models, constructions of dataflow could be mixed with constructions from other paradigms of computation such as those traditionally from process calculi. As an example, synchronous communication is given by the product of presheaves. In this richer world of constructions bisimulation would appear to be the more suitable equivalence.

5.2 Higher-order Dataflow via Geometry of Interaction

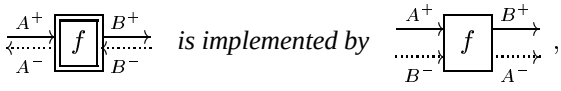
The geometry of interaction programme can be seen in a method of constructing a compact closed category from a traced monoidal category[7] due to Joyal, Street and

[7] In Girard's original treatment this was expressed in terms of traces in the category of Hilbert spaces. That situation is more complicated because not every morphism has a trace, so the categorical presentation of geometry of interaction referred to here is a simplification of the original program.

Verity [18] and also to Abramsky [2]. As such it gives a method for realizing higher-order constructs in terms of feedback. In our setting one takes the categories Kahn and $\mathsf{PProf}_\perp$ and constructs compact-closed categories HKahn and $\mathsf{HProf}_\perp$ which then serve as the interpretations of higher-order Kahn processes and port profunctors.

We will just give the main definition, for more details see [18, 2]. Essentially, one obtains a higher-order model by working with processes with bi-directional "input" and "output". These processes are implemented by uni-directional processes of the underlying category in the obvious way, regarding negative input channels as output channels and negative output as input.

Definition 12. *Given a traced monoidal category $\mathcal{C}$ we define a new category $\mathcal{G}(\mathcal{C})$ as follows. The objects of $\mathcal{G}(\mathcal{C})$ are pairs of objects $(A^+, A^\perp)$ of $\mathcal{C}$. A morphism $f : (A^+, A^\perp) \to (B^+, B^\perp)$ of $\mathcal{G}(\mathcal{C})$ is a $\mathcal{C}$-morphism $f : A^+ \otimes B^\perp \to B^+ \otimes A^\perp$, ie.*



where dotted lines indicate channels that play the opposite role in $\mathcal{G}(\mathcal{C})$. Composition is implemented using composition, trace and symmetries of $\mathcal{C}$ to connect channels with same polarity, ie. for $g : (B^+, B^\perp) \to (C^+, C^\perp)$, $f; g$ is implemented by

$$Tr^{B^-}_{A^+ \otimes C^-, C^+ \otimes A^-}((I_{A^+} \otimes \sigma); (f \otimes I_{C^-}); (I_{B^+} \otimes \sigma'); (g \otimes I_{A^-}); (I_{C^+} \otimes \sigma'')) \ ,$$

for the appropriate symmetries σ, σ' and σ''.

Note that $\mathcal{C}$ embeds into $\mathcal{G}(\mathcal{C})$ as arrows with no negative flow, mapping objects A to (A, I). A symmetric monoidal structure $\odot$ is defined on objects by $(A^+, A^\perp) \odot (B^+, B^\perp) = (A^+ \otimes B^+, B^\perp \otimes A^\perp)$. An obvious duality is defined on objects by $(A^+, A^\perp)^* = (A^\perp, A^+)$, and on arrows by "rotating the underlying process" and swapping the roles of channels. This defines a contravariant functor $(-)^* : \mathcal{G}(\mathcal{C}) \to \mathcal{G}(\mathcal{C})$. Internal hom sets are given by $(A^+, A^\perp) \multimap (B^+, B^\perp) = (B^+, B^\perp) \odot (A^+, A^\perp)^*$.

We immediately get, since it preserves tensor and trace, that the functor Seq from $\mathsf{PProf}_\perp$ to Kahn extends to one between the higher-order categories.

Proposition 7. *We have a functor $HSeq : \mathsf{HProf}_\perp \to \mathsf{HKahn}$, defined using Seq on the base category.*

The higher order structure of HKahn and $\mathsf{HProf}_\perp$ has a very intuitive interpretation in the underlying categories Kahn and $\mathsf{PProf}_\perp$ as plugging networks into contexts. The evaluation map $e_X : X^* \odot X \to I$ is essentially a "router", copying values from ingoing channels to the corresponding out-going ones. As an example, the context $C[-]$ of Sect. 2.1 can be regarded as a higher order process $C : I \to (\overline{\mathsf{A}} \multimap \overline{\mathsf{A}}) \multimap \overline{\mathsf{A}}$ and the processes $\mathcal{A}_i : \overline{\mathsf{A}} \multimap\!\!\perp\!\!\mapsto \overline{\mathsf{A}}$ can be embedded as $\mathcal{A}_i : I \to \overline{\mathsf{A}} \multimap \overline{\mathsf{A}}$. Now, the evaluation $(C \odot \mathcal{A}_1); (I_{\overline{\mathsf{A}}} \odot e_{(\overline{\mathsf{A}}_\perp \circ \overline{\mathsf{A}})}) : I \to \overline{\mathsf{A}}$ is simply the process $C[\mathcal{A}_1]$.

6 Concluding Remarks

The upshot of the work in this paper is a treatment of dataflow that unifies different phenomena - asynchrony and synchrony in our case - and different viewpoints of dataflow networks: dataflow composition as relational composition, dataflow processes as categorical constructs and the very concrete views of dataflow networks as port automata and as sequences of events encoding causality. In particular, dataflow feedback is shown as an instance of a trace operation in a category and this allows one to adapt the ideas from the geometry of interaction program to give a very smooth treatment of higher-order processes. The higher-order models should be compared to the work in [3]. It also remains to explore systematically the full family of models for dataflow, relating automata, event structure and traces-based models to the relational model, following the pattern set in [36]. Work is underway on a bicategory of (higher order) port automata. This will provide further operational back up to the trace on port profunctors and help in the understanding of independence at higher-order. Early attempts have been made to incorporate fairness into the profunctor model; it is hoped to exploit independence along the lines in [8] and include maximal or *completed* observations.

References

[1] S. Abramsky. A generalized kahn principle for abstract asynchronous networks. In *MFPS'89*, volume 442 of *LNCS*, pages 1–21. Springer, 1990.

[2] S. Abramsky. Retracing some paths in process algebra. In U. Montanari and V. Sassone, editors, *CONCUR'96*, volume 1119 of *LNCS*, pages 1–17. Springer, August 1996.

[3] S. Abramsky, S. Gay, and R. Nagarajan. Interaction categories and the foundations of typed concurrent programming. In *Proc. of the 1994 Marktoberdorf summer school*. Springer, 1994.

[4] E. S. Bainbridge. Feedback and generalized logic. *Information and Control*, (31):75–96, 1976.

[5] F. Borceux. *Handbook of categorical logic*, volume 1. Cambridge University Press, 1994.

[6] J. Brock and W. Ackerman. Scenarios: A model of non-determinate computation. In J. Diaz and I. Ramos, editors, *Formalization of Programming Concepts*, volume 107 of *LNCS*. Springer, 1981.

[7] G. L. Cattani and G. Winskel. Presheaf models for concurrency. In *CSL'96*, volume 1258 of *LNCS*, pages 58–75. Springer, 1997.

[8] A. Cheng. Petri nets, traces, and local model checking. Research Series RS-95-39, BRICS, Department of Computer Science, University of Aarhus, July 1995.

[9] J. B. Dennis. First version of a dataflow procedure language. In B. Robinet, editor, *Proceedings Colloque sur la Programmation*, volume 19 of *LNCS*, pages 362–376. Springer, 1974.

[10] V. Diekert and Y. Métivier. *Handbook of Formal Languages.*, volume 3, chapter Partial Commutation and Traces. Springer, 1997.

[11] A. A. Faustini. An operational semantics for pure dataflow. In *ICALP'82*, volume 140 of *LNCS*, pages 212–224. Springer, 1982.

[12] M. Gordon. The semantic challenge of verilog hdl. www.cl.cam.ac.uk/mjcg/, April 1996. Revised version of an invited paper published in LICS'95.

[13] M. Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *TLCA'97*, volume 1210 of *LNCS*, pages 196–213, April 1997.

[14] V. F. Jones. A polynomial invariant for links via von neumann algebras. *Bull. Amer. Math. Soc.*, 129:103–112, 1985.
[15] B. Jonsson. A fully abstract trace model for dataflow networks. In *POPL'89*, pages 155–165. ACM, 1989.
[16] A. Joyal and I. Moerdijk. A completeness theorem for open maps. *Annals of Pure and Applied Logic*, 70(1):51–86, 1994.
[17] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. Research Series RS-94-7, BRICS, Department of Computer Science, University of Aarhus, May 1994. 42 pp. Appears in LICS '93 special issue of *Information and Computation*, 127(2):164–185.
[18] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. volume 119 of *Math. Proc. Camb. Phil. Soc.*, pages 447–468, 1996.
[19] G. Kahn. The semantics of a simple language for parallel programming. In *Information Processing*, volume 74, pages 471–475, 1974.
[20] G. Kahn and D. MacQueen. Coroutines and networks of parallel processes. In Gilchrist, editor, *Proceedings of Information Processing*, pages 993–998. North-Holland, 1977.
[21] P. Katis, N. Sabadini, and R. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, (115):141–178, 1997.
[22] J. Kok. A fully abstract semantics for dataflow nets. In *Proceedings of Parallel Architectures And Languages Europe*, pages 351–368, Berlin, 1987. Springer.
[23] N. A. Lynch and E. W. Stark. A proof of the kahn principle for input/output automata. *Information and Computation*, 82:81–92, 1989.
[24] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 1992.
[25] S. Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1971.
[26] P. Panangaden and V. Shanbhogue. The expressive power of indeterminate dataflow primitive. *Information and Computation*, 98(1):99–131, 1992.
[27] P. Panangaden and E. W. Stark. Computations, residuals and the power of indeterminacy. In *Proc. of the 15th ICALP*, pages 439–454. Springer, 1988.
[28] V. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, (1), 1986.
[29] A. Rabinovich and B. A. Trakhtenbrot. Nets and data flow interpreters. In *Proceedings of the 4th LICS*, pages 164–174, 1989.
[30] A. Rabinovich and B. A. Trakhtenbrot. Nets of processes and dataflow. volume 354 of *LNCS*, 1989. To appear in Proceedings of ReX School on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS.
[31] A. Rabinovich and B. A. Trakhtenbrot. Communication among relations. In M. S. Paterson, editor, *Proc. of the 7th ICALP*, volume 443 of *LNCS*, pages 294–307. Springer, 1990.
[32] P. Selinger. First-order axioms for asynchrony. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR'97*, volume 1243 of *LNCS*, pages 376–390. Springer, 1997.
[33] E. W. Stark. Compositional relational semantics for indeterminate dataflow networks. In *CTCS*, volume 389 of *LNCS*, pages 52–74, Manchester, U.K., 1989. Springer.
[34] E. W. Stark. Dataflow networks are fibrations. In *CTCS*, volume 530 of *LNCS*, pages 261–281. Springer, September 1991.
[35] E. W. Stark. A calculus of dataflow networks. In *Proceedings of the 7th LICS*, pages 125–136, June 1992.
[36] G. Winskel and M. Nielsen. *Handbook of Logic in Computer Science*, volume IV, chapter Models for concurrency. OUP, 1995.
[37] G. Winskel and M. Nielsen. Presheaves as transition systems. In D. Peled, V. Pratt, and G. Holzmann, editors, *POMIV'96*, volume 29 of *DIMACS*. AMS, july 1996.