UNIVERSITÄT DORTMUND

Fachbereich Informatik
Lehrstuhl für Software-Technologie

M E M O Nr. 123

Pipes and Filters: Modelling a Software Architecture Through Relations

Ernst-Erich Doberkat

Juni 2002

Internes Memorandum des
Lehrstuhls für Software-Technologie
Prof. Dr. Ernst-Erich Doberkat
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301

D-44227 Dortmund

ISSN 0933-7725

Pipes and Filters: Modelling a Software Architecture Through Relations

Ernst-Erich Doberkat
Chair for Software Technology
University of Dortmund
doberkat@acm.org

June 13, 2002

Abstract

Pipes and filters is a popular architecture which connects computational components (filters) through connectors (pipes) so that computations are performed in a stream like fashion. The data are transported through the pipes between filters, gradually transforming inputs to outputs. This kind of stream processing has been made popular through UNIX pipes that serially connect independent components for performing a sequence of tasks. We show in this paper how to formalize this architecture in terms of monads, hereby including relational specifications as special cases. The system is given through a directed acyclic graph the nodes of which carry the computational structure by being labelled with morphisms from the monad, and the edges provide the data for these operations. It is shown how fundamental compositional operations like combining pipes and filters, and refining a system by replacing simple parts through more elaborate ones, are supported through this construction. A notion of bisimilar pipes and filters is introduced, it is shown that bisimilarity of components carries over to bisimilarity of entire systems.
Keywords: Software architectures, pipes and filters, refinement, relational specifications, stochastic relations, monads, bisimulation.

1 Introduction and Motivation

Pipes and filters is a popular architecture which connects computational components (filters) through connectors (pipes) so that computations are performed in a stream like fashion. The data are transported through the pipes between filters, gradually transforming inputs to outputs. This kind of stream processing has been made popular through UNIX pipes that serially connect independent components for performing a sequence of tasks. Because of its simplicity and its easy to grasp functionality it is a pet architecture for demonstrating ideas about formalizing the architectural design space (not unlike the data type `Stack` for algebraic specifications or abstract data types). We will show in this paper how to formalize this architecture in terms of monads, hereby including specifications through set theoretic or probabilistic relations as special cases.

Software Architectures The structural aspects of a large programming system are captured through its (software) architecture. Initially, this term was used rather loosely, work being done during the 1990s in particular by M. Shaw and her associates (see e.g. [28, 25, 1]) have established a body of knowledge in the software engineering community about methods for structuring large systems. This translates into practical tools like architectural design languages.
An architecture for a system separates computation from control on the system's level; while the former is represented by algorithms formulated in a programming language, the latter is formulated in terms of *components* (which carry out the computations) and *connectors* (which transport data from one component to another one). Connectors are elevated to first class rank making it possible to reason explicitly about connecting components. Considering an architecture then means identifying connectors and components and describing the interplay between them. Since the emphasis is on structure, formalizing an architecture helps in investigating its salient features; formalizations can be done on different levels. Closest to implementations are formulations through an architectural description language ([27] discusses and assesses a number of them), but other formalisms are used, too:

- the paper by [19] investigates the suitability of the Unified Modelling Language for architectural descriptions (and discusses some desiderata for the language to be usable for architectural descriptions),
- the work reported on through [1, 28] discuss a formulation of pipes and filters through a denotational framework for developing formal models of architectural styles based on the specification language Z,
- category theory is used in formalizations e.g. of architectures for mobile programs based on `UNITY` [30, 13].

The formalization of an architecture permits reasoning about it since it provides precise and abstract models that usually come with analytical techniques. This is in marked contrast to architectural techniques where the shape of an architecture and its architectural parameters are determined experimentally ([12] provides an example for constructing a substantial real life system). Shaw and Garlan [28, Sec.6] discuss architectural formalisms, they distinguish three levels of formalization:

- The architecture of a specific system. This permits a precise characterization of the system-level functions that determine the overall product functionality.
- The formalization of an architectural style. Through the description of architectural abstractions it becomes possible to analyze various static or dynamic properties of common architectural patterns or reference architectures which are used informally e.g. as reference architectures. Essential ingredients in such a formalization are provided by connectors and by components.
- A theory of software architecture. By classifying architectures and representing them with a mathematical machinery a deductive basis for analyzing systems is provided.

We will focus in the present paper on the intermediate level and investigate an architecture where the computational elements are represented through relations.

Relations Computations may be modelled through relations, relating e.g. an input to an output or to a state. This gives rise to a whole calculus of relations (for which the contributions to [5] provides a reference), it draws heavily on ideas from Mathematical Logic in its classic version as embodied by E. Schröder or A. Tarski. This has been outlined in [6].
We will not follow this trail but rather capitalize on a common abstract description of non-deterministic and stochastic relations through monads. A non-deterministic relation R between sets X and Y, thus $R \subseteq X \times Y$ assigns to each $x \in X$ a subset $\{y | \langle x, y \rangle \in R\}$ of possible outcomes. Thus nondeterministic systems may be modelled with these relations. Relation R may be represented as a map from X to the power set $\mathcal{P}(Y)$ of Y. While nondeterministic relations may be interpreted as assigning equal weight to each possible outcome, a probabilistic relation assigns to each input $x \in X$ a probability distribution $K(x)$ on the output, so that $K(x)(B)$ yields the probability that on input x the output is an element of $B \subseteq Y$. If $K(x)(Y) = 1$, it is guaranteed that there will be some output, non-terminating computations may be taken into account by assuming that $K(x)(Y) < 1$, so that $1 - K(x)(Y)$ may be interpreted as the probability for *no output at all*. Hence K can be interpreted as a map from X to the set $\mathbf{S}(Y)$ of a subprobabilities on Y. For this to permit sensible models, X and Y have to be endowed with measurable and sometimes topological structures. Both constructions share a common structure in representing the Kleisli construction for a monad. The case of non-deterministic relations is covered through the power set functor on the category of sets, and for the stochastic case through the functor which assigns each measurable set the space of all subprobability measures. Thus monads (and their associated Kleisli categories) form the common abstraction for both cases, bringing us into the realm of Moggi's argumentation [21, 20] that monads form a suitable basis for modelling computations. Consequently, our architectural modelling will be done on the basis of a monad.

Categories vs. Architectures Categories with their emphasis on structure are a suitable formal tool for modelling software architectures. Focussing on structure implies the independence on any representation in a specification or programming language; technically this is achieved through the use of morphisms and functors. Synthesizing a design sometimes means formulating the components and amalgamating them through a suitable colimit (cf. [13]). Wermelinger and Fiadeiro [30] discuss some salient features of an architectural modelling through categories in the context of their modelling mobile programs:

- Programs may be represented as objects, morphisms show how programs can be composed; explicit use of connectors facilitates the separation of computation and coordination.
- The mechanisms for interconnecting components yielding complex systems are formalized using universal constructs, in this way providing a stage for arguing about these mechanisms formally.
- Extralogical design principles are internalized through properties of universal constructs (e.g., locality of names),
- Different levels of design can be related through functors.
- The evolution of architectures is supported.

When modelling an architecture, one has at least to take care of the computational components and the connectors. Working in a category, the connectors may be represented as objects while the computational components will be modelled as morphisms between the objects. Since computations will be represented as monads, the most natural way is representing a component through the work of the corresponding functor $\mathbf{T}$. Here the Kleisli construction enters the game: suppose for simplicity that the input and the output for a component λ are modelled respectively through the objects x and y, then the computation performed by λ is represented through a Kleisli morphism $x \to \mathbf{T}y$. These assignments are described when modelling a particular architecture, and the work of an instance of this architecture is described in terms of these assumptions. We will show how this can be done for a pipes and filters architecture. This architectural style is simple enough to be studied without having to discuss too many technical, architecture specific issues. It is rather general, hence not tied to a particular domain or application, but it is semantically rich enough to illustrate the constructions proposed and investigated here.

Pipes and Filters Shaw and Garlan describe a pipes and filters architecture [28, Sec. 6.3]: Filters transform streams of data; each filter has input ports from which data are read, and output ports, to which results are written. Computation is performed incrementally and locally: a portion of the data available at the input ports is read, transformed, and written to the output ports which in turn serve as input ports for other components (or as outputs for the system). The filters may be thought of working concurrently, and it is characteristic for this style that the data passing through a filter comes only through its input ports, and leaves only through its output ports, global data are not available. A pipe links an input port to an output port and transmits data from one component to another.
Fig. 1 shows an example for a simple pipes and filters system.
The system has two inputs w_1 and w_2 and two outputs b_1 and b_2, it has four independent components $1, \ldots, 4$. The edges are labelled with the types of the inputs the components accept, and produce, resp.: for example, component 1 accepts inputs of type X_1 and produces outputs of types X_3 and X_4, the former serving as an input to component 2 together with an input of type X_2, the latter serving as an input to component 3 together with an input of type X_5, which is produced by component 2. The entire system accepts two inputs of type X_1 and X_2 and produces two outputs of type X_7 and X_8.
We assume that the system forms a directed graph with filters as nodes and pipes as edges. The graph is assumed to be acyclic, so that loops among filters are not permitted. This is

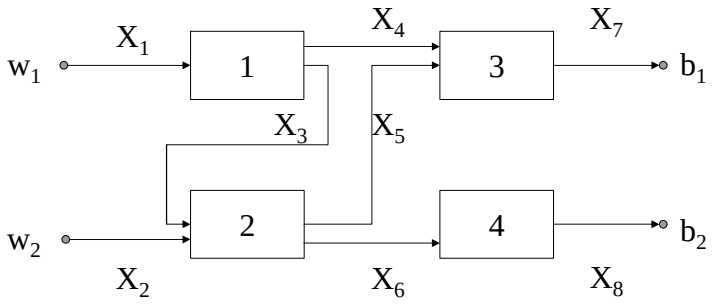


Figure 1: System of Pipes and Filters

a simplifying assumption which does not, however, seem to impose severe restrictions on the use of such a system from a practical point of view.

Overview The rest of this paper is organized as follows: in Sect. 2 we make our assumptions on the category we are working in explicit, we perform some basic constructions, and relate them to Mac Lane's monoidal categories, and to Moggi's strong monads. Sect. 3 illustrates these constructions for the two monoids representing relations, viz., the Manes, and the Giry monad, resp. We will base our construction on directed acyclic graphs (*dag*s), and we perform the construction first on a special class of graphs that we call *stratified*; this is done in Sect. 5. Bisimulations are introduced in Sect. 4, and we show that bisimilar pipes and filters systems will lead to bisimilar results; this is done also in Sect. 5. Sect. 7 shows that assuming stratified graphs is no loss of generality by constructing a stratified graph from a dag, and by showing that the behavior of the entire system is invariant against stratification. This then serves to demonstrate that this construction may be used for incrementally and hierarchically constructing pipes and filters systems. Finally, Sect. 8 compares the formalization proposed here to the one described by Abowd et al. [1] which is also discussed at length in the text book [28], some conclusions are drawn, and suggestions for further work are given.

Acknowledgements The comments from Alexander Fronk and from Georgios Lajios are appreciated: they helped clarifying some obscure points.
The diagrams in this paper have been typeset using Paul Taylor's `diagrams` package.

2 First Steps

This section serves as a preparation for things to come: we remind the reader of the Kleisli product of two morphisms, and we formulate a compatibility condition which relates the product in the category under consideration to the monad which is used for modelling the computations.
Let $\mathfrak{X}$ be a category with finite products, $\langle \mathbf{T}, \eta, \mu \rangle$ a monad in $\mathfrak{X}$. The *Kleisli category* $\mathfrak{X}_{\mathbf{T}}$ has the same objects as $\mathfrak{X}$, an arrow between a and b in $\mathfrak{X}_{\mathbf{T}}$ is an arrow in $\mathfrak{X}$ between a and $\mathbf{T}b$. Let $f : a \to b$ and $g : b \to c$ be arrows in $\mathfrak{X}_{\mathbf{T}}$, thus $f : a \to \mathbf{T}b, g :\to \mathbf{T}c$ in $\mathfrak{X}$, then the

composition $g * f$ is defined as $\mu_c \circ \mathbf{T}g \circ f$, with $\circ$ as the composition in $\mathfrak{X}$, cf. [17, Thm. VI.5.1].
The category $\mathfrak{X}^{(n)}$ has as objects n-tuples of objects of $\mathfrak{X}$, and n-tuples of morphisms, the composition being defined componentwise. Define functors $\mathbf{G}_{\mathbf{T}}^{(n)}, \mathbf{H}_{\mathbf{T}}^{(n)} : \mathfrak{X}^{(n)} \to \mathfrak{X}$ upon setting ($n \geq 1$)

$$\begin{aligned} \mathbf{G}_{\mathbf{T}}^{(n)}\langle x_1, \ldots, x_n\rangle &:= \mathbf{T}x_1 \times \cdots \times \mathbf{T}x_n \\ \mathbf{H}_{\mathbf{T}}^{(n)}\langle x_1, \ldots, x_n\rangle &:= \mathbf{T}\left(x_1 \times \cdots \times x_n\right), \end{aligned}$$

and, if $\phi_i : x_i \to y_i$ are morphisms in $\mathfrak{X}$, then

$$\begin{aligned} \mathbf{G}_{\mathbf{T}}^{(n)}\langle \phi_1, \ldots, \phi_n\rangle &:= \mathbf{T}\phi_1 \times \cdots \times \mathbf{T}\phi_n \\ \mathbf{H}_{\mathbf{T}}^{(n)}\langle \phi_1, \ldots, \phi_n\rangle &:= \mathbf{T}\left(\phi_1 \times \cdots \times \phi_n\right). \end{aligned}$$

$\mathbf{T}$ models the computations performed in the components, and which are partially done in parallel. This in turn will be modelled through finite products. Hence $\mathbf{T}$ should be naturally related to the product in $\mathfrak{X}$; the present proposal assumes compatibility which mediates between $\mathbf{T}(x) \times \mathbf{T}(y) \times \mathbf{T}(z)$ and $\mathbf{T}(x \times y \times z)$ using the natural transformation $1_{\mathbf{T}} : \mathbf{T} \stackrel{\bullet}{\to} \mathbf{T}$ and introducing another one between $\mathbf{G}_{\mathbf{T}}^{(2)}$ and $\mathbf{H}_{\mathbf{T}}^{(2)}$. To be specific:

Definition 1 *Monad* $\mathbf{T}$ *is* compatible with the product *in* $\mathfrak{X}$ *iff there exists a natural transformation* $\theta : \mathbf{G}_{\mathbf{T}}^{(2)} \stackrel{\bullet}{\to} \mathbf{H}_{\mathbf{T}}^{(2)}$ *which makes this diagram commutative:*

$$\begin{array}{ccc} \mathbf{T}(x) \times \mathbf{T}(y) \times \mathbf{T}(z) & \xrightarrow{(1_{\mathbf{T}} \times \theta)_{\langle x,y,z\rangle}} & \mathbf{T}(x) \times \mathbf{T}(y \times z) \\ \Big\downarrow{\scriptstyle (\theta \times 1_{\mathbf{T}})_{\langle x,y,z\rangle}} & = & \Big\downarrow{\scriptstyle \theta_{\langle x,y\times z\rangle}} \\ \mathbf{T}(x \times y) \times \mathbf{T}(z) & \xrightarrow[\theta_{\langle x\times y,z\rangle}]{} & \mathbf{T}(x \times y \times z) \end{array}$$

θ *is called the* mediating *transformation.*

A mediating transformation θ spawns a sequence $(\theta^{(n)})_{n\geq 1}$ of natural transformations

$$\theta^{(n)} : \mathbf{G}_{\mathbf{T}}^{(n)} \stackrel{\bullet}{\to} \mathbf{H}_{\mathbf{T}}^{(n)}$$

in the following way:

$$\begin{aligned} \theta_x^{(1)} &:= 1_{\mathbf{T}x} \\ \theta_{\langle x,y\rangle}^{(2)} &:= \theta_{\langle x,y\rangle} \\ \theta_{\langle x_1,\ldots,x_{n+1}\rangle}^{(n+1)} &:= \theta_{\langle x_1\times\cdots\times x_n, x_{n+1}\rangle} \circ \left(\theta^{(n)} \times 1_{\mathbf{T}}\right)_{\langle x_1,\ldots,x_{n+1}\rangle} \end{aligned}$$

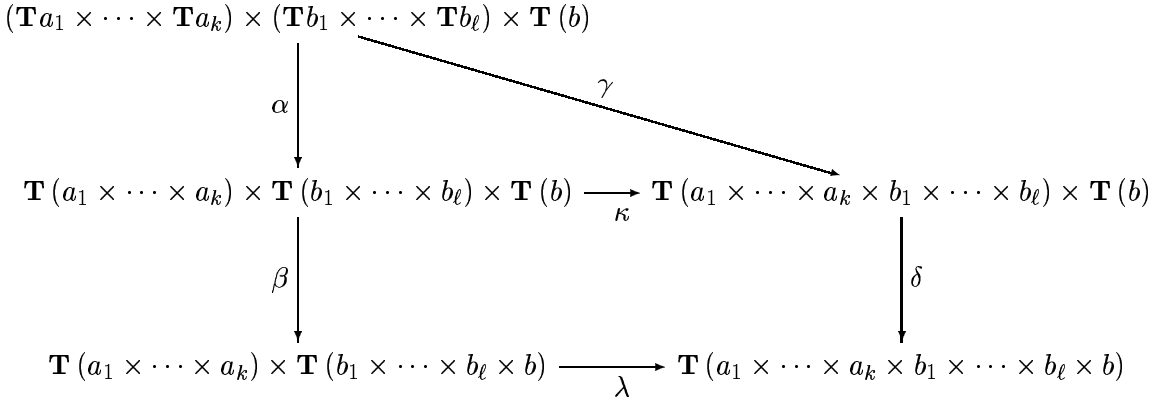
Lemma 1 *Suppose that* $\mathbf{T}$ *is compatible with the product in* $\mathfrak{X}$*, and let* θ *be the mediating transformation. Define* $\theta^{(n)}$ *as above, then this sequence has the following properties:*

1. $\theta^{(n)}$ *is a natural transformation,*

2. for all $k, \ell \in \mathbb{N}$ and for all objects x_i, y_i we have

$$\theta_{\langle x_1 \times \cdots \times x_k, y_1 \times \cdots \times y_\ell \rangle} \circ \left(\theta^{(k)} \times \theta^{(\ell)} \right)_{\langle x_1, \ldots, x_k, y_1, \ldots, y_\ell \rangle} = \theta^{(k+\ell)}_{\langle x_1, \ldots, x_k, y_1, \ldots, y_\ell \rangle}$$

Proof: 1. The first part is established by induction on n, since the composition of natural transformations is again a natural transformation.
2. The second part is proved by induction on ℓ, the start of the induction representing just the inductive definition from above. The induction step is established through the commutativity of this diagram:



with

$$\begin{aligned}
\alpha &:= \left(\theta^{(k)} \times \theta^{(\ell)} \times 1_{\mathbf{T}} \right)_{\langle a_1, \ldots, a_k, b_1, \ldots, b_\ell, b \rangle} \\
\beta &:= (1_{\mathbf{T}} \times \theta)_{\langle a_1 \times \cdots \times a_k, b_1 \times \cdots \times b_\ell \times b \rangle} \\
\gamma &:= \left(\theta^{(k+\ell)} \times 1_{\mathbf{T}} \right)_{\langle a_1, \ldots, a_k, b_1, \ldots, b_\ell, b \rangle} \\
\delta &:= \theta_{\langle a_1 \times \cdots \times a_k \times b_1 \times \cdots \times b_\ell, b \rangle} \\
\kappa &:= (\theta \times 1_{\mathbf{T}})_{\langle a_1 \times \cdots \times a_k, b_1 \times \cdots \times b_\ell, b \rangle} \\
\lambda &:= \theta_{\langle a_1 \times \cdots \times a_k, b_1 \times \cdots \times b_\ell \times b \rangle}
\end{aligned}$$

The upper triangle is commutative because of the induction hypothesis, the lower square is just the condition on θ from Def. 1. Then the assertion follows, since

$$\begin{aligned}
\lambda \circ \beta \circ \alpha &= \delta \circ \kappa \circ \alpha \\
&= \delta \circ \gamma,
\end{aligned}$$

and

$$\begin{aligned}
\beta \circ \alpha &= \left(\theta^{(k)} \times \theta^{(\ell+1)} \right)_{\langle a_1, \ldots, a_k, b_1, \ldots, b_\ell, b \rangle} \\
\delta \circ \gamma &= \theta^{(k+\ell+1)}_{\langle a_1, \ldots, a_k, b_1, \ldots, b_\ell, b \rangle}
\end{aligned}$$

□
The product in $\mathfrak{X}$ defines together with $\mathbf{T}$ an associative operation:

Definition 2 *Let $\tau : a \to \mathbf{T}b$ and $\tau' : a' \to \mathbf{T}b'$ be morphisms, then define*

$$\tau \times_{\mathbf{T}} \tau' : a \times a' \to \mathbf{T}\left(b \times b'\right)$$

upon setting

$$\tau \times_{\mathbf{T}} \tau' := \theta_{\langle b,b' \rangle} \circ \tau \times \tau'.$$

Example 1 will show that $\times_{\mathbf{T}}$ does not exhibit the universal properties which would be necessary to form a product (thus the category $\mathfrak{X}_{\mathbf{T}}$ does not necessarily have finite products, even if $\mathfrak{X}$ has them).
We have, however:

Corollary 1 *Let $\tau : a \to \mathbf{T}b, \tau : a' \to \mathbf{T}b', \tau : a'' \to \mathbf{T}b''$ be morphisms in $\mathfrak{X}$, then*

$$\left(\tau \times_{\mathbf{T}} \tau'\right) \times_{\mathbf{T}} \tau'' = \tau \times_{\mathbf{T}} \left(\tau' \times_{\mathbf{T}} \tau''\right).$$

Proof: Lemma 1 shows that both sides of this equation equal $\theta^{(3)}_{\langle b,b',b'' \rangle} \circ (\tau \times \tau' \times \tau'')$. □
Remark: 1. Mac Lane [17, Ch. XI.2] defines a *monoidal functor* between monoidal categories which comes close to the compatibility definition proposed here for an endofunctor, where the role of the tensor product there is played by the product here. The present definition does not require any conditions on the terminal elements and its image under the functor, thus it is weaker. Mac Lane formulates a transformation quite similar to the one given in Lemma 1. The proof in [17] refers, however, to a coherence theorem and seems to be a bit inaccessible by not making the construction transparent. Consequently, a direct proof is given here.
2. Moggi [21, Def. 3.2] defines a *strong monad* in a category which is closed under finite products by postulating the existence of a natural transformation $t_{a,b} : a \times \mathbf{T}b \to \mathbf{T}(a \times b)$ having some properties which relate $\mathbf{T}$ to the product in the category (t called a *tensorial strength*). In [20, 3.2.3] it is shown how the tensorial strength induces a natural transformation $\mathbf{G}^{(2)}_{\mathbf{T}} \stackrel{\bullet}{\to} \mathbf{H}^{(2)}_{\mathbf{T}}$ in the terminology used here.

3 Examples

This section discusses two examples, indicating how set theoretic and probabilistic relations fit into the framework developed here. The first example derives from the well known monad investigated by Manes, giving rise to set theoretic relations, the second one generalizes an observation by Giry, yielding probabilistic relations. The respective relations are obtained through the Kleisli construction.
We fix for the rest of this section a monoid H with 1 as an identity. Such a monoid could be a group, the free semigroup over an alphabet, or a $\vee$-semilattice with a smallest element, forming the supremum in H as multiplication. The functors under consideration will have the general form $X \mapsto \mathbf{F}(H \times Y)$. While the component coming from Y will cater for the system's work, the component coming from H will be responsible for any output, which will be concatenated through the respective steps. Consider the case that computations are modelled through set-theoretic relations, then $\langle h, y \rangle \in R(x)$ means that upon input x the output piped to the next computation will be y, and that h is some control output (e.g., the cost incurred for computing y or, as in UNIX pipes, some output to port `stderr`). Hence the next computation is not affected by the value of h. The additional information generated in this way are collected at the end of the computation proper; the outcome from the entire computation consists of the computational output, and of additional information which are accumulated through the semigroup multiplication.

3.1 The Manes Monad

Let $\mathfrak{S}$ be the category of sets with maps as morphisms. It is well known that $\langle \mathcal{P}, \mu', \eta' \rangle$ forms a monad, where $\mathcal{P}$ assigns to each set its powerset, $\mu'_X : \mathcal{P}(\mathcal{P}(X)) \to \mathcal{P}(X)$ maps $A \subseteq \mathcal{P}(X)$ to $\bigcup A$, and η'_X assigns each element $x \in X$ the corresponding singleton $\{x\}$, cf.[17, Ex. VI.2.1]. A slight generalization making use of monoid H works as follows: define the functor $\mathbf{M}$ through

$$\mathbf{M}(X) := \mathcal{P}(H \times X),$$

and if $f : X \to Y$ is a map, $A \subseteq H \times X$, then

$$\mathbf{M}(f)(A) := \{\langle h, f(x)\rangle | \langle h, x\rangle \in A\}$$

defines the action of the functor on the morphisms of $\mathfrak{S}$. Now define

$$\mu_X : \mathbf{M}(\mathbf{M}(X)) \to \mathbf{M}(X)$$

upon setting

$$\mu_X(A) := \bigcup_{\langle h_1, b\rangle \in A} \{\langle h_1 h_2, x\rangle | \langle h_2, x\rangle \in b\},$$

then it is not difficult to see that

$$\mu : \mathbf{M}^2 \stackrel{\bullet}{\to} \mathbf{M}$$

is a natural transformation. The natural transformation

$$\eta : 1_{\mathfrak{S}} \stackrel{\bullet}{\to} \mathbf{M}$$

is defined by $\eta_X : x \mapsto \{\langle 1, x\rangle\}$. Then it is shown by standard calculations that $\langle \mathbf{M}, \eta, \mu\rangle$ is a monad in $\mathfrak{S}$. It is also immediate that

$$\mathbf{M}(X) \times \mathbf{M}(Y) \ni \langle A, B\rangle \mapsto \{\langle h_1 h_2, x, y\rangle | \langle h_1, x\rangle \in A, \langle h_2, y\rangle \in B\} \in \mathbf{M}(X \times Y)$$

defines a natural transformation that mediates between the functor and the product in $\mathfrak{S}$.
A Kleisli morphism between X and Y is a relation between X and $H \times Y$. This is well investigated for the case that the monoid H is trivial, cf. [4, 16.1.4]; this generalizes to the present case. Let $R : X \to \mathbf{M}(Y)$ and $S : Y \to \mathbf{M}(Z)$ be Kleisli morphisms, thus $R \subseteq X \times (H \times Y)$ and $S \subseteq Y \times (H \times Z)$ are relations. Then their product is stated in Prop. 1 which summarizes this example.

Proposition 1 *$\langle \mathbf{M}, \eta, \mu\rangle$ is a monad in the category $\mathfrak{S}$ of sets with maps as morphisms which is compatible with the product in $\mathfrak{S}$. The Kleisli product for the relations $R \subseteq X \times (H \times Y)$ and $S \subseteq Y \times (H \times Z)$ is given through*

$$(S * R)(x) = \{\langle h_1 h_2, z\rangle | \exists y \in Y : \langle h_1, y\rangle \in R(x) \wedge \langle h_2, z\rangle \in S(y)\}.\Box$$

The composition of two relations will have the outputs from the single steps concatenated, and will proceed via an intermediate state.
There is a topological variant of this monad: let X be a Polish space, and assume that H is a Polish semigroup (this is the case e.g. when H is the free semigroup generated by some Polish space Y endowed with the topological sum of $(Y^n)_{n\geq 0}$). Endow the set

$$\mathcal{K}(X) := \{K \subseteq H \times X | K \neq \emptyset \text{ is compact}\}$$

of all compact non-empty subsets of $H \times X$ with the Vietoris topology. This topology has the sets

$$\{K \in \mathcal{K}(X) | K \subseteq U_0, K \cap U_1 \neq \emptyset, \ldots, K \cap U_k \neq \emptyset\}$$

as a basis, where $U_0, \ldots, U_k$ are open subsets of $H \times X$. It is well known that $\mathcal{K}(X)$ is a Polish space, see e.g. [16, Thm. 4.25].
Since H is a topological semigroup, it is not difficult to see that

$$\mathcal{K}(\mathcal{K}(X)) \ni \mathcal{W} \mapsto \bigcup_{\langle h_1, b\rangle \in \mathcal{W}} \{\langle h_1 h_2, x\rangle | \langle h_1, x\rangle \in b\} \in \mathcal{K}(X)$$

constitutes a continuous map, cf. [16, Ex. 4.29]. The constructions are done exactly as for monad $\langle \mathbf{M}, \eta, \mu\rangle$. This yields:

Corollary 2 *$\langle \mathcal{K}, \eta, \mu\rangle$ is a monad in the category of Polish spaces with continuous maps as morphisms. The Kleisli product is given as in Prop. 1.* □

The hyperspace construction addressed in Cor. 2 is, when the monoid is trivial, of practical interest in medical image processing [18, 7].
Measurable set-valued maps (a.k.a. measurable relations [15], although the name is somewhat misleading) are the straightforward generalization of compact valued maps that are used in stochastic dynamic optimization. They may be represented as measurable maps from a measurable space to the space of all closed non-empty subsets of a Polish space, measurability on the latter being Borel measurability with respect to the Vietoris topology. It does not seem possible, however, to apply to this case a similar monadic construction as the ones discussed above. Under suitable topological assumptions, these relations may be represented through the support function of a stochastic relation [9]. This representation, however, turns out not to be always compositional: the composition of the set-valued relation does not always correspond to the representation through the composed stochastic relation, cf. [10]. Capturing this case requires then probably another approach.

3.2 The Giry Monad

Let $\mathfrak{M}$ be the category of measurable spaces with measurable maps as morphisms. If X is a measurable space (the σ-algebra is usually omitted in notation, its members are referred to as *measurable subsets* of X) we denote by $\mathbf{S}(X)$ the set of all sub-probability measures on X which is made into an measurable space itself by endowing it with the *-σ-algebra. The latter is the smallest σ-algebra which makes the evaluation $m \mapsto m(A)$ measurable for each measurable subset A of X . Let $f : X \to Y$ be a measurable map between the measurable spaces X and Y, then f induces a map (again denoted by f) between $\mathbf{S}(X)$ and $\mathbf{S}(Y)$ upon setting ($m \in \mathbf{S}(X)$, $B \subseteq Y$ measurable):

$$f(m)(B) := m(\{x \in X | f(x) \in B\} = m\left(f^{-1}[B]\right).$$

$f(m)$ is referred to as the *image measure* of m under f. Integration with respect to the image measure may be captured through the change of variable formula which will be somewhat helpful in the sequel: let $\psi : Y \to \mathbb{R}$ be a bounded and measurable function, then

$$\int_Y \psi(y)\ f(m)(dy) = \int_X \psi(f(x))\ m(dx).$$

The *Giry monad* $\langle \mathbf{P}, \eta', \mu' \rangle$ [14] assigns each measurable space X the set of all probability measures $\mathbf{P}(X)$ on X. A measurable map $f : X \to Y$ is assigned the map $\mathbf{P}f : \mathbf{P}(X) \to \mathbf{P}(Y)$ which sends m to $f(m)$. Then $f : \mathbf{P}(X) \to \mathbf{P}(Y)$ is easily seen to be measurable with the weak-*-σ-algebras, and to yield a probability. Consequently, $\mathbf{P}$ is an endofunctor on $\mathfrak{M}$. The unit $\eta'_X : X \to \mathbf{P}(X)$ assigns to each $x \in X$ the Dirac measure δ_x on x, and the multiplication $\mu'_X : \mathbf{P}(\mathbf{P}(X)) \to \mathbf{P}(X)$ is defined by

$$\mu'_X(\Phi)(A) := \int_{\mathbf{P}(X)} m(A)\ \Phi(dm).$$

The morphisms in the Kleisli category for this monad are just the transition probabilities, i.e., the probabilistic relations (cf. [23, 2, 11]). Remember that K is a *transition probability* between the measurable spaces X and Y iff $K(x)$ is a probability for $x \in X$, and if $x \mapsto K(x)(A)$ is a measurable function for each measurable set $A \subseteq Y$. This is written as $K : X \rightsquigarrow Y$. Let $K : X \rightsquigarrow Y$ and $L : Y \rightsquigarrow Z$ be transition probabilities, then their product $L \circ K$ is defined through ($x \in X, C \subseteq Z$ is a measurable subset)

$$(L \circ K)(x)(C) := \int_Y L(y)(C)\ K(x)(dy).$$

We endow the monoid H with a measurable structure which makes multiplication measurable, when $H \times H$ carries the product σ-algebra, i.e., the smallest σ-algebra which makes the projections measurable. It is then not difficult to see that for each measure $m \in \mathbf{S}(H \times X)$, for each measurable map $f : X \to Y$, and for each measurable subset $C \subseteq H \times Y$ the map

$$s \mapsto m(\{\langle t, x\rangle | \langle st, f(x)\rangle \in C\})$$

is measurable on H.
Examples for such measurable monoids are given by topological monoids; the Borel sets then form the canonical measurable structure. Topological groups are probably the most prominent examples. If H is a $\vee$-semilattice with a smallest element, then it is not difficult to see that $\vee$ is a continuous operation, when H is endowed with the interval topology (i.e. the topology which has open intervals as subbase). Taking again the Borel sets for this topology, we see that these semilattices yield measurable monoids, too.
Now define for the measurable space X, the element $x \in X$, the measure $m \in \mathbf{S}(H \times \mathbf{S}(H \times X))$ and the measurable subset $A \subseteq H \times X$:

$$\begin{aligned} \mathbf{G}(X) &:= \mathbf{S}(H \times X) \\ \eta_X(x) &:= \delta_{\langle 1, x\rangle} \\ \mu_X(m)(A) &:= \int_{H \times \mathbf{S}(H \times X)} p(\{\langle h, x\rangle | \langle gh, x\rangle \in A\})\ m(d\langle g, p\rangle) \end{aligned}$$

If $f : X \to Y$ is a measurable map, then we put

$$\begin{aligned} \mathbf{G}(f)(m)B &:= m(\{\langle h, x\rangle | \langle h, f(x)\rangle \in B\}) \\ &= m\left((id_H \times f)^{-1}[B]\right) \\ &= (id_H \times f)(m)(B). \end{aligned}$$

Consequently, $\mathbf{G}(f) : \mathbf{S}(X) \to \mathbf{S}(Y)$ is measurable, and if $\psi : H \times Y \to \mathbb{R}$ is measurable, the change of variable formula implies that

$$\int_{H\times Y} \psi(h,y)\ \mathbf{G}(f)(m)(d\langle h,y\rangle) = \int_{H\times X} \psi(h,f(x))\ m(d\langle h,x\rangle)$$

holds.
We will show now that $\langle \mathbf{G}, \eta, \mu\rangle$ is a monad in $\mathfrak{M}$, adapting and extending Giry's proofs [14] to the situation at hand. It will also be shown that this monad is compatible with the product on $\mathfrak{M}$.

Lemma 2 *$\mathbf{G}$ is an endofunctor in $\mathfrak{M}$; $\eta : 1_{\mathfrak{M}} \stackrel{\bullet}{\to} \mathbf{G}$ and $\mu : \mathbf{G}^2 \stackrel{\bullet}{\to} \mathbf{G}$ are natural transformations.*

Proof: 1. It is immediate that $\mathbf{G} : \mathfrak{M} \to \mathfrak{M}$ is a functor, and that η is a natural transformation.
2. Let $f : X \to Y$ be a measurable map, then we have for $m \in \mathbf{G}(Y)$ and for the measurable subset $B \subseteq G \times Y$

$$\begin{aligned}\left(\mu_Y \circ \mathbf{G}^2 f\right)(m)(B) &= \int_{H\times\mathbf{S}(H\times Y)} (\mathbf{G}f)(q)(\{\langle h,y\rangle | \langle gh,y\rangle \in B\})\ m(d\langle s,q\rangle) \\ &= \int_{H\times\mathbf{S}(H\times X)} q(\{\langle h,x\rangle | \langle gh,f(x)\rangle \in B\})\ m(d\langle s,q\rangle).\end{aligned}$$

This coincides with $(\mathbf{G}f \circ \mu_X)(m)(B)$. Consequently, $\mu : \mathbf{G}^2 \stackrel{\bullet}{\to} \mathbf{G}$. □

Proposition 2 *$\langle \mathbf{G}, \eta, \mu\rangle$ is a monad in $\mathfrak{M}$.*

Proof: 1. We need to demonstrate that the associative, and the unit laws hold. The change of variable formula implies that

$$\int_{G\times\mathbf{G}(X)} \psi\ d(\mathbf{G}\eta_X)(p) = \int_{H\times X} \psi(h,\eta_X(x))\ p(d\langle h,x\rangle)$$

holds, whenever $p \in \mathbf{G}(X)$, and $\psi : H \times \mathbf{G}(X) \to \mathbb{R}$ is measurable and bounded. Consequently,

$$\begin{aligned}(\mu_X \circ \mathbf{G}\eta_X)(p)(B) &= \int_{H\times X} \eta_X(\{\langle g,x\rangle | \langle hg,x\rangle \in B\})\ p(d\langle h,x\rangle) \\ &= p(B) \\ &= \left(\mu_X \circ \eta_{\mathbf{G}(X)}\right)(p)(B).\end{aligned}$$

holds for every $p \in \mathbf{G}(X)$, and every measurable subset B of $H \times X$. This establishes the unit laws.
2. As far as the associative law is concerned, fix $r \in \mathbf{G}^3 X$, and a measurable subset E of $H \times \mathbf{G}(X)$. The change of variable formula implies that

$$\begin{aligned}\left(\mu_X \circ \mu_{\mathbf{G}(X)}\right)(r)(E) &= \int_{H\times\mathbf{G}(X)} q(\{\langle g,y\rangle | \langle hg,y\rangle \in E\})\ \mu_{\mathbf{G}(X)}(r)(d\langle h,q\rangle) \\ &= \int_{H\times\mathbf{G}^2 X} \left(\int_{H\times\mathbf{G}(X)} q(\{\langle j,y\rangle | \langle ghj,y\rangle \in E\})\ p(d\langle h,q\rangle)\right)\ r(d\langle g,p\rangle)\end{aligned}$$

On the other hand, expanding the definitions, and applying the change of variables formula suitably, it is seen that these transformations hold:

$$\begin{aligned}(\mu_X \circ \mathbf{G}\mu_X)(r)(E) &= \int_{H\times\mathbf{G}(X)} p(\{\langle h,y\rangle|\langle gh,y\rangle \in E\})\ \mathbf{G}\mu_X(r)(d\langle g,p\rangle)\\ &= \int_{H\times\mathbf{G}^2X} \mu_X(q)(\{\langle h,y\rangle|\langle gh,y\rangle \in E\})\ r(d\langle g,q\rangle)\\ &= \left(\mu_X \circ \mu_{\mathbf{G}(X)}\right)(r)(E).\end{aligned}$$

This shows that the associative law is valid. □
The monad is compatible with the product in $\mathfrak{M}$. Recall that the product of two measurable spaces X and Y lives on the Cartesian product of the underlying sets, and carries the smallest σ-algebra there which contains all the measurable rectangles $A\times B$ with A a measurable subset of X, and B of Y, resp. Now let $m_i \in \mathbf{G}(X_i)$ $(i=1,2)$, and define for the measurable subset C of $H\times X_1\times X_2$ the *H-product of m_1 and m_2*:

$$(m_1 \otimes_H m_2)(C) := \int_{H\times X_1} m_2(\{\langle h_2,y\rangle|\langle h_1h_2,x,y\rangle \in C\})\ m_1(d\langle h_1,x\rangle),$$

then $m_1 \otimes_H m_2 \in \mathbf{G}(X_1\times X_2)$. In fact, we can say more:

Lemma 3 *The H-product is associative, it constitutes a natural transformation $\mathbf{G}^{(2)}_{\mathbf{G}} \stackrel{\bullet}{\to} \mathbf{H}^{(2)}_{\mathbf{G}}$.*

Proof: 1. Associativity of the H-product is an easy consequence of Fubini's Theorem on product integration.
2. Let $f: X\to X'$ and $g: Y\to Y'$ be measurable maps, then we have for the measures $m_1\in\mathbf{G}(X), m_2\in\mathbf{G}(Y)$ and for the measurable subset C' of $H\times X'\times Y'$

$$\begin{aligned}(\mathbf{G}(f)(m_1)\otimes_H \mathbf{G}(g)(m_2))(C') &= \int_{H\times X} m_2(\{\langle h_2,y\rangle|\langle h_1h_2,f(x),g(y)\rangle \in C'\})\ m_1(d\langle h_1,x\rangle)\\ &= (m_1\otimes_H m_2)(\{\langle h,x,y\rangle|\langle g,f(x),g(y)\rangle \in C'\})\\ &= \mathbf{G}(f\times g)(\langle m_1,m_2\rangle \mapsto m_1\otimes_H m_2)(C').\end{aligned}$$

But this means that the H-product is natural for $\mathbf{G}^{(2)}_{\mathbf{G}}$ and $\mathbf{H}^{(2)}_{\mathbf{G}}$. □
Let $K: X\to Y$ and $L: Y\to Z$ be morphisms in the Kleisli category $\mathfrak{M}_{\mathbf{G}}$, thus $K: X\rightsquigarrow H\times Y$ and $L: Y\rightsquigarrow H\times Z$ are probabilistic relations. With this in mind, proposition 3 summarizes this example and shows what the Kleisli product of K and L looks like.

Proposition 3 *$\langle \mathbf{G},\eta,\mu\rangle$ forms a monad in the category of all measurable spaces with measurable maps as morphisms, the natural transformation assigning two measures their H-product mediates between $\mathbf{G}$ and the product in $\mathfrak{M}$, and the Kleisli product for the probabilistic relations $K: X\rightsquigarrow G\times Y$ and $L: Y\rightsquigarrow G\times Z$ is given through*

$$(L * K)(x)(C) = \int_{H\times Y} L(y)(\{\langle h,x\rangle|\langle gh,x\rangle \in C\})\ K(x)(d\langle g,y\rangle).\square$$

Concluding the discussion about the Giry monad, we show that $\mathfrak{M}_{\mathbf{G}}$ does not have finite products; this has been hinted at just before Cor. 1.

Example 1 Let, for simplicity, H be the trivial monoid $\{1\}$, so that we omit it from the notation. Suppose that $\times_{\mathbf{G}}$ is a product in $\mathfrak{M}_{\mathbf{G}}$, and fix two non-empty measurable spaces X_1 and X_2. There exists a measurable space X and the two projections $p_i : X \to \mathbf{G}(X_i)$ such that, whenever $K_i : S \to \mathbf{G}(X_i)\ (i = 1, 2)$ is a morphism, we can find a morphism $K : S \to \mathbf{G}(X)$ such that $K_i = p_i * K$ holds for $i = 1, 2$. This means that

$$K_i(s)(B_i) = \int_X p_i(x)(B_i)\ K(s)(dx)$$

always holds. Now let K_i be a Markov kernel, hence $K_i(s)(X_i)$ equals always 1. This implies that $p_i(x)(X_i) = 1$ is true $K(s)$-almost everywhere for each s, and for each K which can be so constructed. Note that p_1, p_2 do not depend on the specific choice of K_1, K_2. But then we have for any $L_i : S \to \mathbf{G}(X_i)$ with product L:

$$\begin{aligned} L_1(s)(X_1) &= \int_X p_1(x)(X_1)\ L(s)(dx) \\ &= \int_X p_2(x)(X_2)\ L(s)(dx) \\ &= L_2(s)(X_2) \end{aligned}$$

is implied. Since we cannot always maintain $L_1(s)(X_1) = L_2(X_2)$ it follows that $\mathfrak{M}_{\mathbf{G}}$ does not have finite products.

4 Bisimulation

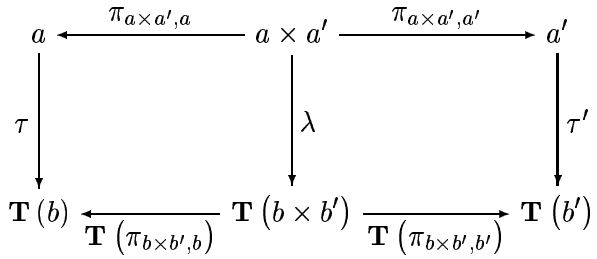
Roughly speaking, two systems are bisimilar if reactions in one system correspond to reactions in the other one, and vice versa. This is being made precise through an intermediate system from which the bisimilar systems can be derived by suitable morphisms. Bisimilarity will be discussed now, it will be defined, and related to some other notions of bisimilarity found in the literature. In particular, we indicate that the present definition is weaker than the one used by Aczel and Mendler, and by Rutten. It is shown that the product $\times_{\mathbf{T}}$ does preserve bisimilarity, and so does the Kleisli product of morphisms (it is currently not clear whether this is also true for the stronger notions of bisimilarity).
A Kleisli morphism $\tau : a \to \mathbf{T}b$ can be interpreted as an object $\langle a, b, \tau\rangle$ in the comma category $\mathbb{1}_{\mathfrak{X}} \downarrow \mathbf{T}$; a morphism $\langle a, b, \tau\rangle \to \langle a', b', \tau'\rangle$ in the comma category is a pair $\langle f, g\rangle$ of morphisms $f : a \to b$ and $g : a' \to b'$ such that $\tau' \circ f = \mathbf{T}g \circ \tau$ holds. A bisimulation between the objects $\langle a, b, \tau\rangle$ and $\langle a', b', \tau'\rangle$ is then an object $\langle a \times a', b \times b', \lambda\rangle$ such that

$$\begin{aligned} \langle \pi_{a\times a', a}, \pi_{b\times b', b}\rangle : &\quad \langle a \times a', b \times b', \lambda\rangle \to \langle a, b, \tau\rangle \\ \langle \pi_{a\times a', a'}, \pi_{b\times b', b'}\rangle : &\quad \langle a \times a', b \times b', \lambda\rangle \to \langle a', b', \tau'\rangle \end{aligned}$$

are morphisms. This yields:

Definition 3 *$\tau : a \to \mathbf{T}b$ and $\tau' : a' \to \mathbf{T}b'$ are called* bisimilar *iff there exists a Kleisli*

morphism $\lambda : a \times a' \to \mathbf{T}(b \times b')$ *such that the following diagram is commutative:*



λ *is called the* mediating morphism*; bisimilarity of* $\tau : a \to \mathbf{T}b$ *and* $\tau' : a' \to \mathbf{T}b'$ *is abbreviated through* $\langle a, b, \tau\rangle \sim \langle a', b', \tau'\rangle$.

The notion of bisimulation is inspired by the one proposed by Aczel and Mendler [3, p. 363], and by Rutten [24, p. 11], resp., for coalgebras, and in [8, 11] for probabilistic relations. The notion proposed here is weaker, however. Consider as an illustration the Manes monad $\mathbf{M}$ on the category of sets. Put for the Kleisli morphisms $R : X \to \mathbf{M}(Y)$ as an abbreviation

$$x \ \vdash_R^a \ y :\Leftrightarrow \langle a, y\rangle \in R(x),$$

(the suggestive notation $x \ \overset{a}{\to}_R \ y$ used by Rutten is not adequate since we are not dealing with coalgebras, hence cannot talk about transitions); $x \ \vdash_R^a \ y$ may be read as: x *yields* y *under* a *by* R. It is not difficult to see that

$$\langle X, Y, R\rangle \sim \langle X', Y', R'\rangle$$

is equivalent to the conditions

1. for all $x \in X, y \in Y$: if $x \ \vdash_R^a \ y$, and $x' \in X'$, then there exists $y' \in Y'$ such that $x' \ \vdash_{R'}^a \ y'$,
2. for all $x' \in X', y' \in Y'$: if $x' \ \vdash_{R'}^a \ y'$, and $x \in X$, then there exists $y \in Y$ such that $x \ \vdash_R^a \ y'$

which are assumed to hold for all $a \in H$. In fact, Rutten's proof [24, Ex. 2.1] carries over. But this provides an adequate interpretation of bisimilar objects for the Manes monad: $\langle X, Y, R\rangle$ is bisimilar to $\langle X', Y', R'\rangle$ iff whenever an input x yields a reaction y under a by R, and if x' is an input to the bisimilar system, then x' yields a reaction y' under a by R', and vice versa. Adapting bisimilarity in the way it is defined in the context of coalgebras would require for the Manes monad postulating relations $R_0 \subseteq X \times X'$, and $S_0 \subseteq Y \times Y'$ and a Kleisli morphism $\lambda : R_0 \to \mathbf{M}(S_0)$ as a mediating morphism. This generalizes to the existence a subobject a'' of $a \times a'$, and of b'' of $b \times b'$, resp., with a mediating morphism $\lambda : a'' \to b''$.
Alternatively, bisimulation could have been defined as a span of zigzag morphisms in the comma category $\mathbb{1}_{\mathcal{X}} \downarrow \mathbf{T}$ as in [8]. But this would be difficult to interpret even for the Manes monad.
We will see that bisimulation is preserved through forming the Kleisli product, and the Kleisli composition now:

Proposition 4 *Bisimilarity has the following properties:*

1. If $\langle a_1, b_1, \tau_1\rangle \sim \langle a_1', b_1', \tau_1'\rangle$ *and* $\langle a_2, b_2, \tau_2\rangle \sim \langle a_2', b_2', \tau_2'\rangle$ *then*

$$\langle a_1 \times a_2, b_1 \times b_2, \tau_1 \times_{\mathbf{T}} \tau_2\rangle \sim \langle a_1' \times a_2', b_1' \times b_2', \tau_1' \times_{\mathbf{T}} \tau_2'\rangle.$$

2. $\langle a, b, \tau\rangle \sim \langle a', b', \tau'\rangle$ *and* $\langle b, c, \sigma\rangle \sim \langle b', c', \sigma'\rangle$ *together imply*

$$\langle a, c, \sigma * \tau\rangle \sim \langle a', c', \sigma' * \tau'\rangle.$$

Proof: 1. Suppose λ_i mediates between $\langle a_i, b_i, \tau_i\rangle$ and $\langle a_i', b_i', \tau_i'\rangle$, then a straightforward computation shows that

$$\mathbf{T}\left(q_{b_1} \times q_{b_2} \times q_{b_1'} \times q_{b_2'}\right) \circ \theta_{\langle b_1 \times b_1', b_2 \times b_2'\rangle} \circ \lambda_1 \times \lambda_2 \circ \left(p_{a_1} \times p_{a_1'} \times p_{a_2} \times p_{a_2'}\right)$$

mediates between $\tau_1 \times_{\mathbf{T}} \tau_2$ and $\tau_1' \times_{\mathbf{T}} \tau_2'$, where we abbreviate

$$\begin{aligned} p_z &:= \pi_{a_1 \times a_2 \times a_1' \times a_2', z}, \\ q_z &:= \pi_{b_1 \times b_2 \times b_1' \times b_2', z}. \end{aligned}$$

2. Let λ_1 and λ_2 mediate between $\langle a, b, \tau\rangle$ and $\langle a', b', \tau'\rangle$, and between $\langle b, c, \sigma\rangle$ and $\langle b', c', \sigma'\rangle$, respectively. We claim that $\lambda_2 * \lambda_1$ mediates between $\langle a, c, \sigma * \tau\rangle$ and $\langle a', c', \sigma' * \tau'\rangle$. In fact,

$$\begin{aligned} (\sigma * \tau) \circ \pi_{a \times a', a} &= \mu_c \circ \mathbf{T}(\sigma) \circ \tau \circ \pi_{a \times a', a} \\ &= \mu_c \circ \mathbf{T^2}\left(\pi_{c \times c', c}\right) \circ \mathbf{T}(\lambda_2) \circ \lambda_1 \\ &= \mathbf{T}\left(\pi_{c \times c', c}\right) \circ (\mu_{c \times c'} \circ \mathbf{T}(\lambda_2) \circ \lambda_1) \\ &= \mathbf{T}\left(\pi_{c \times c', c}\right) \circ (\lambda_2 * \lambda_1) \end{aligned}$$

holds, because $\mu : \mathbf{T^2} \stackrel{\bullet}{\to} \mathbf{T}$ is a natural transformation. The equation

$$\left(\sigma' * \tau'\right) \circ \pi_{a \times a', a'} = \mathbf{T}\left(\pi_{c \times c', c'}\right) \circ (\lambda_2 * \lambda_1)$$

is established in the same way. □

5 The Basic Construction

We will show now how to associate to a pipes and filters system a computation by composing computations done in its components. This construction will be carried out for technical reasons for graphs that exhibit a certain regularity: the nodes are partitioned into layers so that the information flows strictly from one layer to the next one. This restriction will be removed in Sect. 7, after we have shown in Sect. 6 that each graph can be stratified.
Fix in this section a dag $\mathcal{G} = (V, E)$ with roots W and leaves B; for convenience we assume the set V of nodes to be somehow linearly ordered. Put for node n

$$\begin{aligned} \bullet n &:= \{m \in V | \langle m, n\rangle \in E\} \\ n\bullet &:= \{m \in V | \langle n, m\rangle \in E\} \end{aligned}$$

as the sets of nodes which have an edge into that node or out of it, resp. $\mathcal{G}$ is not supposed to have any isolated nodes, i.e., nodes n with $\bullet n \cup n\bullet = \emptyset$.
We define sets $(S_j)_{0 \le j \le k}$ through

$$\begin{aligned} S_0 &:= W, \\ S_{j+1} &:= \{n \in V | \bullet n \subseteq S_j\} \ (j \ge 0). \end{aligned}$$

Definition 4 *The dag $\mathcal{G} = (V, E)$ is called* stratified *iff the sets $(S_j)_{0 \leq j \leq k}$ form a partition of V for some k. The maximal index k such that $S_k \neq \emptyset$ is denoted by $\Lambda(\mathcal{G})$.*

Let for the rest of this section $\mathcal{G}$ be a stratified graph.

Observation 1 *The set of inputs*

$$\{\langle m, n\rangle | \langle m, n\rangle \in E, n \in S_j\}$$

into the set S_j of nodes equals the set of outputs

$$\{\langle m, \ell\rangle | \langle m, \ell\rangle \in E, m \in S_{j-1}\}$$

from the sets S_{j-1} for $j \geq 1$.

Proof: This follows directly from the fact that $\mathcal{G}$ is stratified. □
This observation shows that each node n is in some uniquely determined set S_j. If n is an inner node (thus if $j > 0$ and $j < k$), then $\langle m, n\rangle \in E$ implies $m \in S_{j-1}$, and $\langle n, m\rangle \in E$ implies $m \in S_{j+1}$. Depicting $S_0, \ldots, S_k$ as blocks from left to right, information flows into n only from nodes in S_{j-1}, thus from nodes on the left, and flows from n only into nodes in S_{j+1}, hence into nodes on the right.
We associate now objects from category $\mathfrak{X}$ with edges, and nodes with morphisms in $\mathfrak{X}_{\mathbf{T}}$. To be specific, each edge $\langle k, n\rangle \in E$ is assigned an object $\gamma_{\langle k,n\rangle}$ in $\mathfrak{X}$. If $\mathbf{T}$ is the Manes functor, this means that an edge $\langle k, n\rangle$ is assigned a set which represents the flow from node k to node n. For $\mathbf{T}$ as the Giry functor, the edge is assigned a measurable space which also represents the flow along this edge: if it is used as an input, then it is the sample space of all inputs for a probabilistic relation, if it is used as an output, then it represents the space of all probability measures over this space, cf. Example 2.
The input to node n and the output from this node are then reflected respectively through the respective products

$$\begin{aligned} \mathfrak{i}(\gamma, n) &:= \prod\{\gamma_{\langle k,n\rangle} | k \in \bullet n\} \ (n \notin W) \\ \mathfrak{o}(\gamma, n) &:= \prod\{\gamma_{\langle n,k\rangle} | k \in n\bullet\} \ (n \notin B) \end{aligned}$$

Each inner node n is labelled with a Kleisli morphism

$$\mathfrak{a}(\gamma, n) : \mathfrak{i}(\gamma, n) \to \mathbf{T}(\mathfrak{o}(\gamma, n)),$$

so that $\mathfrak{a}(\gamma, n)$ models the work being performed by node n.

Example 2 Suppose that $\bullet n = \{m_1, \ldots, m_r\}$ and $n\bullet = \{\ell_1, \ldots, \ell_s\}$.

1. For the Manes monad we assign sets $X_1, \ldots, X_r$ to the edges $\langle m_1, n\rangle, \ldots, \langle m_r, n\rangle$ and sets $Y_1, \ldots, Y_s$ to the edges $\langle n, \ell_1\rangle, \ldots, \langle n, \ell_s\rangle$. The node n itself is assigned a relation

   $$\mathfrak{a}(\gamma, n) \subseteq (X_1 \times \cdots \times X_r) \times (H \times Y_1 \times \cdots \times Y_s).$$

   Suppose that $\langle x_1, \ldots, x_r\rangle \in X_1 \times \cdots \times X_r$ is an input to node n which is related to output $\langle h, y_1, \ldots, y_s\rangle$. That tuple represents the node's work, and we have two kinds of results: the tuple $\langle y_1, \ldots, y_s\rangle$ which in turn is being communicated to other nodes in the pipeline, and $h \in H$ which may be interpreted as immediate result which could be read off this processing element. Prop. 1 indicates that all these results, which will not be communicated as input to other filters, will be accumulated as control percolates through the system.

2. For the Giry monad, X_i and Y_j are measurable spaces, and

$$\mathfrak{a}(\gamma, n) : X_1 \times \cdots \times X_r \rightsquigarrow H \times Y_1 \times \cdots \times Y_s$$

is a stochastic relation. Thus for an input $\langle x_1, \ldots, x_r \rangle \in X_1 \times \cdots \times X_r$, and for a measurable $B \subseteq H \times Y_1 \times \cdots \times Y_s$ we get

$$\mathfrak{a}(\gamma, n)(x_1, \ldots, x_r)(B)$$

as the probability that the computation in node n terminates, and that $\langle h, y_1, \ldots, y_s \rangle$ will be a member of B; the interpretation of the components for this tuple is the same as above.

This indicates that a relational environment for modelling the basic scenario for pipes and filters is provided, capturing both the nondeterministic and the probabilistic case.

Definition 5 *Call $\langle \mathcal{G}, \gamma \rangle$ a* pipes and filters system *(abbreviated as* PF-system*) over the monad $\langle \mathbf{T}, \eta, \mu \rangle$ iff the following conditions hold:*

- *$\mathcal{G} = \langle V, E \rangle$ is a directed graph with W and B as the sets of roots, and leaves, resp.*
- *$\forall \langle n, m \rangle \in E : \gamma_{\langle n,m \rangle}$ is an object in $\mathfrak{X}$,*
- *$\forall n \in V \setminus (W \cup B) : \mathfrak{a}(\gamma, n) : \mathfrak{i}(\gamma, n) \to \mathbf{T}(\mathfrak{o}(\gamma, n))$ is a morphism in $\mathfrak{X}$*

The system $\langle \mathcal{G}, \gamma \rangle$ is called stratified *iff $\mathcal{G}$ is stratified.*

Since the monad will be fixed in the sequel, we will not mention it explicitly when talking about PF-systems; unless explicitly mentioned, PF-systems will be stratified in this Section. Now define for $0 < j \leq k$ the object

$$\mathfrak{g}(\gamma, S_j) := \prod \{\mathfrak{i}(\gamma, n) \,|n \in S_j\},$$

then $\mathfrak{g}(\gamma, S_j)$ indicates the kind of flow into S_j (which is, because of Obs. 1, the flow out of S_{j-1}); hence the component S_j has the "input signature" $\mathfrak{g}(\gamma, S_{j-1})$ and the "output signature" $\mathfrak{g}(\gamma, S_j)$.
The work being done in S_j can be represented through the Kleisli morphism

$$\mathfrak{A}(\gamma, S_j) : \mathfrak{g}(\gamma, S_{j-1}) \to \mathbf{T}(\mathfrak{g}(\gamma, S_j)),$$

with

$$\mathfrak{A}(\gamma, S_j) := \theta^{(\#S_j)}_{\langle n | n \in S_j \rangle} \circ \prod \{\mathfrak{a}(\gamma, n) \,|n \in S_j\},$$

where θ is the natural transformation which mediates between $\mathbf{T}$ and the product in $\mathfrak{X}$, cf. Def. 1. The linear order on V makes $\theta^{(\#S_j)}_{\langle n | n \in S_j \rangle}$ uniquely determined. The work of the entire system is then represented through

$$\mathfrak{P}(\mathcal{G}, \gamma) := \mathfrak{A}(\gamma, S_{k-1}) \ * \ \ldots \ * \ \mathfrak{A}(\gamma, S_1).$$

The construction shows that

$$\mathfrak{P}(\mathcal{G}, \gamma) : \mathfrak{g}(\gamma, S_1) \to \mathbf{T}(\mathfrak{g}(\gamma, S_k))$$

is a Kleisli morphism between the inputs to the system and the outputs from it, thus represents the systems's work.
The example that follows discusses a particular pipes and filters system, stratifies the graph and exercises the construction proposed here for our example monads, hence for set theoretic and for probabilistic relations.

Example 3 Consider the system of pipes and filters represented through the graph in Fig. 1 again. It has the roots $\{w_1, w_2\}$, the leaves $\{b_1, b_2\}$ and the filters $\{1, 2, 3, 4\}$. The edges are decorated with the expected types of inputs, thus with the γ-values. The graph is not stratified, but the version in Fig. 2 is. Note that we have introduced two new artificial nodes Δ_2 and Δ_4.

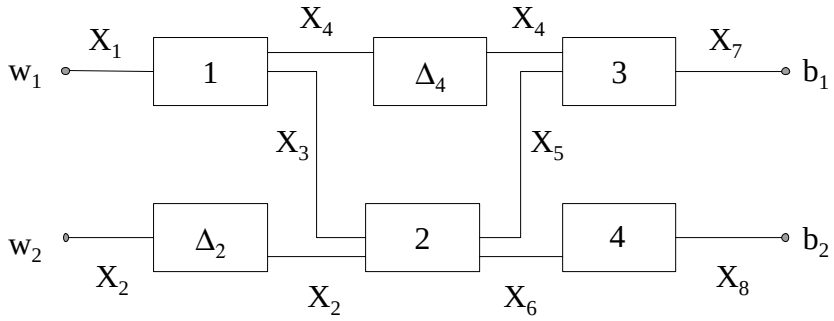


Figure 2: Stratified System of Pipes and Filters

Then we have these sets $S_0, \ldots, S_4$:

$$S_0 = \{w_1, w_2\}, S_1 = \{1, \Delta_2\}, S_2 = \{\Delta_4, 2\}, S_3 = \{3, 4\}, S_4 = \{b_1, b_2\}.$$

Let us first discuss the set valued case. The monoid is fixed, also for later use. Since the Δ_i are expected to have no functional effect, we will have

$$\begin{aligned} \Delta_2 &:= \{\langle x, 1, x\rangle | x \in X_2\}, \\ \Delta_4 &:= \{\langle x, 1, x\rangle | x \in X_4\}. \end{aligned}$$

We assume that node n is represented by a relation R_n between the corresponding sets, augmented by the information provided by the monoid, thus e.g.,

$$R_2 \subseteq (X_2 \times X_3) \times (H \times X_5 \times X_6)\,.$$

The construction yields e.g. for component S_2:

$$\mathfrak{A}\,(\gamma, S_2) = \{\langle x_2, x_3, x_4, h_2, x_4, x_5, x_6\rangle | x_4 \in X_4, \langle x_2, x_3, h_2, x_5, x_6\rangle \in R_2\}$$

The work of the entire pipes and filters system is then described through the following relation, which is a subset of $(X_1 \times X_2) \times (H \times X_7 \times X_8)$, as expected:

$$\begin{aligned} &\{\langle x_1, x_2, h_1h_2h_3h_4, x_7, x_8\rangle | \exists x_3 \in X_3, x_4 \in X_4, x_5 \in X_5, x_6 \in X_6 : \\ &\quad \langle x_1, h_1, x_3, x_4\rangle \in R_1, \langle x_2, x_3, h_2, x_5, x_6\rangle \in R_2, \langle x_4, x_5, h_3, x_7\rangle \in R_3, \langle x_6, h_4, x_8\rangle \in R_4\}. \end{aligned}$$

Note the accumulating effect which the filters exhibit through the elements of the monoid. Turning to stochastic relations, we assume that the monoid carries a measurable structure which makes multiplication measurable, cf. Sect. 3.2. The Δ_i $(i = 2, 4)$ are Dirac kernels: we put

$$\Delta_i(x) := \delta_{\langle 1,x\rangle} \in \mathbf{G}X_i.$$

Node n represented is this time through a stochastic relation K_n between the appropriate sets, e.g.,

$$K_2 : X_2 \times X_3 \rightsquigarrow H \times X_5 \times X_6.$$

The construction gives then e.g. for component S_2:

$$\mathfrak{A}\,(\gamma, S_2)\,(x_2, x_3, x_4) = K_2(x_2, x_3) \otimes_H \Delta_4(x_4),$$

thus the probability that the computation in components S_2 will give an element of the measurable set $D \subseteq H \times X_4 \times X_5 \times X_6$ after input of $\langle x_1, x_2, x_3\rangle \in X_1 \times X_2 \times X_3$ is computed as

$$\begin{aligned}
&\mathfrak{A}\,(\gamma, S_2)\,(x_2, x_3, x_4)\,(D) \\
&\quad = \int_{H\times X_5\times X_6} \Delta_4(x_4)\left(\{\langle h_4, x_4'\rangle | \langle h_2 h_4, x_4', x_5, x_6\rangle \in D\}\right)\ K_2(x_2, x_3)(d\langle h_2, x_5, x_6\rangle) \\
&\qquad\qquad = K_2(x_2, x_3)\,(\{\langle h_2, x_5, x_6\rangle | \langle h_2, x_4, x_5, x_6\rangle \in D\})
\end{aligned}$$

Let $f : H \times X_4 \times X_5 \times X_6 \to \mathbb{R}$ be a bounded and measurable function, then a computation of the Kleisli product according to Proposition 3 shows that $(x_1 \in X_1, x_2 \in X_2)$

$$\begin{aligned}
&\int_{H\times X_4\times X_5\times X_6} f\ d\left((\mathfrak{A}\,(\gamma, S_2) * \mathfrak{A}\,(\gamma, S_1))\right)(x_1, x_2) \\
&\quad = \int_{H\times X_3\times X_4}\int_{H\times X_5\times X_6} f(gh, x_4, x_5, x_6)\ K_2(x_2, x_3)(d\langle h, x_5, x_6\rangle)\ K_1(x_1)(d\langle g, x_3, x_4\rangle)
\end{aligned}$$

We get in this way for $\langle x_1, x_2\rangle \in X_1 \times X_2$ and the measurable subset $F \subseteq H \times X_7 \times X_8$

$$\begin{aligned}
&\mathfrak{P}\,(\mathcal{G}, \gamma)\,(x_1, x_2)(F) \\
&\qquad = \int_{H\times X_4\times X_5\times X_6} \mathfrak{A}\,(\gamma, S_3)\,(x_4, x_5, x_6)\,(\{\langle h, x_7, x_8\rangle | \langle gh, x_7, x_8 \in F\}) \times \\
&\qquad\qquad \times d\,(\mathfrak{A}\,(\gamma, S_2) * \mathfrak{A}\,(\gamma, S_1))\,(x_1, x_2)(d\langle g, x_4, x_5, x_6\rangle) \\
&\qquad = \int_{H\times X_3\times X_4}\int_{H\times X_5\times X_6}\int_{H\times X_7} K_4(x_6)(\{\langle h_1, x_8\rangle | \langle g_1 h g h_1, x_7, x_8\rangle \in F\}) \times \\
&\qquad\qquad \times K_3(x_4, x_5)(d\langle g, x_7\rangle) K_2(x_2, x_3)(d\langle h, x_5, x_6\rangle) K_1(x_1)(d\langle g_1, x_3, x_4\rangle)
\end{aligned}$$

as the work of the entire pipes and filters system.
We compare the work done by two assignments of objects and morphisms to the same pipes and filters system. Call two PF-systems $\langle \mathcal{G}, \gamma_1\rangle$ and $\langle \mathcal{G}, \gamma_2\rangle$ over the same graph $\mathcal{G}$, which need not be stratified (cf. Prop.9), bisimilar iff the constituting parts are bisimilar. To be specific:

Definition 6 *Let $\langle \mathcal{G}, \gamma_1 \rangle$ and $\langle \mathcal{G}, \gamma_2 \rangle$ be PF-systems. We say that $\langle \mathcal{G}, \gamma_1 \rangle$* is bisimilar to *$\langle \mathcal{G}, \gamma_2 \rangle$ (abbreviated as $\langle \mathcal{G}, \gamma_1 \rangle \sim \langle \mathcal{G}, \gamma_2 \rangle$) iff this condition holds:*

$$\forall n \in V \setminus (W \cup B) : \langle \mathfrak{i}(\gamma_1, n), \mathfrak{o}(\gamma_1, n), \mathfrak{a}(\gamma_1, n) \rangle \sim \langle \mathfrak{i}(\gamma_2, n), \mathfrak{o}(\gamma_2, n), \mathfrak{a}(\gamma_2, n) \rangle.$$

Hence $\langle \mathcal{G}, \gamma_1 \rangle \sim \langle \mathcal{G}, \gamma_2 \rangle$ iff the respective functionalities associated with each inner node in one system are bisimilar. This is a local definition, confining the attention to each node individually. For the systems' level, the question arises whether or not bisimilarity is preserved through the pipes and filters system, i.e., if the resulting morphisms are bisimilar.
Tracing the construction, it can be established that bisimilarity is indeed preserved by a pipes and filters system:

Proposition 5 *If the stratified PF-systems $\langle \mathcal{G}_1, \gamma_1 \rangle$ and $\langle \mathcal{G}_2, \gamma_2 \rangle$ are bisimilar, then*

1. *$\langle \mathfrak{g}(\gamma_1, S_{j-1}), \mathfrak{g}(\gamma_1, S_j), \mathfrak{A}(\gamma_1, S_j) \rangle$ and $\langle \mathfrak{g}(\gamma_2, S_{j-1}), \mathfrak{g}(\gamma_2, S_j), \mathfrak{A}(\gamma_2, S_j) \rangle$ are bisimilar for $0 < j < k$*
2. *$\langle \mathfrak{g}(\gamma_1, S_1), \mathfrak{g}(\gamma_1, S_k), \mathfrak{P}(\mathcal{G}, \gamma_1) \rangle$ and $\langle \mathfrak{g}(\gamma_2, S_1), \mathfrak{g}(\gamma_2, S_k), \mathfrak{P}(\mathcal{G}, \gamma_2) \rangle$ are bisimilar.*

Proof: 1. The proof of Cor. 1 and an easy inductive argument shows that

$$\begin{aligned} \mathfrak{A}(\gamma, S_j) &:= \theta^{(\#S_j)}_{\langle n | n \in S_j \rangle} \circ \prod \{ \mathfrak{a}(\gamma, n) \,|\, n \in S_j \} \\ &= \mathfrak{a}(\gamma, n_1) \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \mathfrak{a}(\gamma, n_\ell), \end{aligned}$$

if $S_j = \{n_1, \ldots, n_\ell\}$. Consequently, the first part follows from the first part of Prop. 4 with the associativity of $\times_{\mathbf{T}}$.
2. The Kleisli product $*$ is associative, thus the second part follows also from Prop. 4 by induction. □
We will now prepare for removing the condition that a PF-system should be stratified.

6 Stratifying Graphs

The assumption in carrying out the basic construction in Sect. 5 has been that the graph underlying the PF-system is stratified. But graphs rarely are, so it becomes necessary to make provisions for generalizing the construction to general directed graphs. The strategy is to devise a way of stratifying a graph, to perform the construction on the new graph, and to make sure that all graphs that are stratified versions of the given one perform the same work. The present section is auxiliary in character and provides an algorithm for stratifying, Sect. 7 will do the generalization.
The algorithm in Fig. 3 produces from $\mathcal{G} = (V, E)$ a stratified graph $\mathcal{G}' = (V', E')$ with $V \subseteq V'$ and $E' \cap (V \times V) \subseteq E$. It assumes that $\mathcal{G}$ does not have any isolated nodes, and that each node lies on a path from a root to a leaf. We assume that we have a source Q of fresh nodes which is disjoint from $V \cup E$; invoking the function $newq()$ will produce a fresh node. The map α initially gives the in-degree of a node; we use some auxiliary values which will be needed and discussed in the sequel.
Thus we iterate over all edges, removing roots as we go; for a node n we use $H(n)$ for recording which nodes have edges leading into n that will be removed. When we see that a node n has no longer any edges having n as a target, this node will be promoted to a root (and removed

```
Edges := E; Nodes := V; zeta := 0;
while Edges ≠ ∅ do
      forall n ∈ range Edges do
            H(n) := {a|⟨a, n⟩ ∈ Edges, α(a) = 0};
      od;
      forall a ∈ domain H do
            d(a) := zeta;
      od;
      Edges := Edges \ {⟨a, n⟩|⟨a, n⟩ ∈ Edges, α(a) = 0};
      forall n ∈ Nodes do
            r := #{k|⟨k, n⟩ ∈ Edges};
            if r = 0 then
                  α(n) := 0;
            else
                  for j := r + 1 to α(n) do
                        choose m from H(n);
                        H(n) := H(n) \ {m};
                        q := newq();
                        α(q) := 0; V := V ∪ {q};
                        E := (E \ {⟨m, n⟩}) ∪ {⟨m, q⟩, ⟨q, n⟩};
                        Edges := Edges ∪ {⟨q, n⟩};
                  od; -- forall
            fi;
      od; -- forall
      zeta := zeta + 1;
od; -- while
```

Figure 3: Algorithm `Stratify`

in due course); promotion to a root means changing the in-degree $\alpha(n)$ to 0. If it turns out, however, that there are still edges going into that node (note that in this case $\#H(n) =$ equals $\alpha(n) - r$), we replace each edge $\langle m, n\rangle$ by a pair of edges $\langle m, q\rangle$ and $\langle q, n\rangle$, where q is a fresh node which is put into the set V of nodes.
Since each dag has roots, and since $\mathcal{G}$ is assumed to have no isolated nodes, it is not difficult to see that the algorithm `Stratify` terminates. It is also evident that the new graph $\mathcal{G}' = (V', E')$ has the given one as a subgraph in the sense that $V \subseteq V'$ and $E' \cap (V \times V) \subseteq E$ both hold.

Observation 2 *Let $n \in E'$ be a node in $\mathcal{G}'$, and assume that there exists a path from a root of $\mathcal{G}'$ to n. Then this path has length $d(n)$.*

Proof: 1. We proceed by induction on the value of *zeta*. The begin is trivial, since exactly the roots of $\mathcal{G}$ are removed, and no new roots are introduced.
2. Now let $zeta = k$, and assume that $d(n)$ equals $k + 1$. This means that $\alpha(n)$ is set to 0 when *zeta* has the value k. We distinguish the cases that n is a new node introduced in this step from the case that $\alpha(n)$ is set to 0 because $r = 0$ holds.

- If n is a new node, we can find an edge $\langle m_1, n_1\rangle$ which gave rise to this creation, hence that edge is replaced by the pair of edges $\langle M_1, n\rangle$ and $\langle n, n_1\rangle$. Edge $\langle m_1, n_1\rangle$ is a member

of the set *Edges* before control enters the body of the actual loop, thus will be removed. The induction hypothesis makes sure that each path from a root to m in the graph constructed so far has length k, thus $d(n) = k + 1$.

- If n is no new node, the assumption that there is a path in the new graph to n implies that, since there is no node m with $\langle m, n\rangle \in Edges$, there are edges $\langle m, n\rangle$ which have been deleted in the step before. For all these m we have $d(m) = k$. In the new graph have all these nodes m the property that each path from a root to them has length k.

This implies the assertion. □
An immediate consequence of Obs. 2 is

Proposition 6 *Algorithm* `Stratify` *produces a stratified graph.*

Proof: Put in the notation from above

$$S_j := \{n \in V' | d(n) = j\}.$$

Then S_0 is the set of roots for $\mathcal{G}'$ as well as for $\mathcal{G}$, and if node n is in S_{j+1}, then all its predecessors (w. r. t. $\mathcal{G}'$ are in S_j. These sets are mutually disjoint, and $S_{k'} = \emptyset$ for all $k' \geq k$ for some minimal index k. Since $n \in S_{d(n)}$ holds for each node $n \in V'$, we see that $(S_j)_{0 \leq j \leq k}$ forms a partition of V'. □
Armed with this tool, we may now enter the discussion of the general case.

7 The General Case

We will show now that all the stratified PF-systems which can be constructed from a given one will do the same work, so that this morphism is an invariant, and that it is sensible to assign it as work to a non-stratified PF-system. This has as a remarkable consequence that two constructions can be carried out that help in composing larger systems from smaller ones: we show that two PF-systems can be glued together (as a horizontal extension), and that hierarchical refinement is available as construction technique, permitting the expansion of a node by an entire subsystem. This is a vertical extension.
Both the PF-system $\langle \mathcal{G}, \gamma\rangle$, and $\mathcal{G} = \langle V, E\rangle$ as the graph underlying it are fixed. The sets W and B denote the roots, and the leaves of $\mathcal{G}$, resp. We fix also the set Q which serves as a reservoir of fresh nodes for stratification.
We begin with an adaptation of the algorithm in Fig. 3 to PF-systems by taking the labels for edge and nodes coming with such a system into account. To be specific, suppose we replace an edge $\langle m, n\rangle$ from the set of edges by the pair $\langle m, q\rangle$ and $\langle q, n\rangle$ with the fresh node $q \in Q$. Then we put

$$\begin{aligned} \gamma_{\langle m,q\rangle} &:= \gamma_{\langle m,n\rangle}, \\ \gamma_{\langle q,n\rangle} &:= \gamma_{\langle m,n\rangle}, \\ \mathfrak{a}\,(\gamma, q) &:= \eta_{\gamma_{\langle q,n\rangle}}. \end{aligned}$$

Thus if the edge $\langle m, n\rangle$ carries type a, where a is an object in $\mathfrak{X}$, then the new edges carry this type, and the node inserted is assigned the Kleisli morphism η_a; note that the natural transformation η provides the identities in the Kleisli category $\mathfrak{X}_{\mathbf{T}}$. In terms of pipes and

filters, by inserting $\eta_{\gamma_{\langle q,n\rangle}}$ we insert a no-op into the system, since the filter introduced in this way evidently does not do any other work than transporting inputs unchanged to outputs. In this way we obtain from $\langle \mathcal{G}, \gamma\rangle$ a stratified PF-system $\langle \mathcal{G}_1, \gamma\rangle$, reusing γ for simplicity.
The graph constructed by algorithm `Stratify` is an extension of the given graph. This is made precise now.

Definition 7 *The graph $\mathcal{G}' = (V', E')$ is called a* Q-extension *to $\mathcal{G}$ iff*

1. *$\mathcal{G}'$ is stratified with $E' \cap (V \times V) \subseteq E$, and $\mathcal{G}'$ has the same roots as $\mathcal{G}$,*
2. *$V \subseteq V'$, and $V' \setminus V \subseteq Q$,*
3. *if $\langle n, m\rangle \in E \setminus E'$, then there exists a unique path $n = q_0, \ldots, q_k = m$ from n to m in $\mathcal{G}'$ with $\langle q_i, q_{i+1}\rangle \in E'$ for $0 \leq i < k$,*
4. *for all $q \in V' \setminus V$, $\#(\bullet q) = \#(q\bullet) = 1$.*

Thus a Q-extension has new nodes from the fountain Q of nodes only, an edge in E is either an edge in E', or its endpoints are connected through a unique path that runs entirely through Q (apart from the endpoints, of course). The new nodes in $\mathcal{G}'$ do not have a rich social life by being neighbor to only two other nodes, thus such a node receives inputs from exactly one node and propagates it to a unique other node.

Observation 3 *The graph constructed from algorithm* `Stratify` *is an Q-extension to $\mathcal{G}$.*

Proof: If $\mathcal{G}'$ is the graph constructed from $\mathcal{G}$, then $\mathcal{G}'$ has been shown to be stratified in Prop. 6. The construction makes sure that the other conditions from Def. 7 are satisfied. □
Any Q-extension can be decorated as indicated above: the nodes from Q receive η_x as their function, where x is an appropriate object which labels the edges leading into that node, and out of it, resp. This leads to the notion of an Q-extension to a PF-system which will not be formally defined since the definition is obvious (the reader is invited to formulate it).
We want to establish that the work of a PF-system is an invariant for all Q-extensions to a given PF-system. For this we should make sure that the composition of Kleisli morphisms and the operation $\times_{\mathbf{T}}$ which resembles a product so closely relate to each other like composition and product:

Definition 8 *The monad $\langle \mathbf{T}, \eta, \mu\rangle$ satisfies the* $\sharp$-condition *iff*

1. *$\eta_{a\times b} = \eta_a \times_{\mathbf{T}} \eta_b$ for all objects a and b in $\mathfrak{X}$,*
2. *for the morphisms $f_i : a_i \to \mathbf{T}b_i, g_i : b_i \to \mathbf{T}c_i$ $(i = 1, 2)$ the equality*

$$(g_1 \times_{\mathbf{T}} g_2)*(f_1 \times_{\mathbf{T}} f_2) = (g_1*f_1) \times_{\mathbf{T}} (g_2*f_2)$$

holds.

Thus the identity on $a \times b$ in $\mathfrak{X}_{\mathbf{T}}$ is obtained from the respective identities on a and b by performing the $\times_{\mathbf{T}}$-operation. In terms of computation, combining the identities on a and on b independently to a component yields the identity in $a \times b$. The second condition explains the name: viewing the Kleisli composition $*$ as a horizontal operation along the flow of information which indicates piping, and $\times_{\mathbf{T}}$ as a vertical operation modelling independent composition, the equation is visualized in Fig. 4. Hence piping of composed computations is tantamount to composing piped computations.
Let us investigate our reference categories:

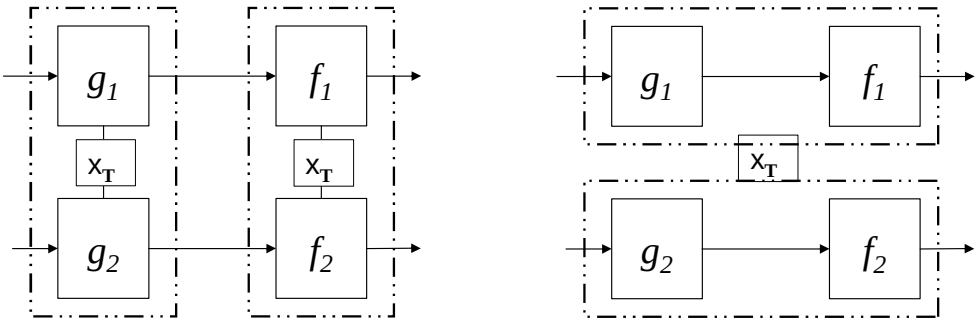


Figure 4: $(g_1 \times_{\mathbf{T}} g_2)*(f_1 \times_{\mathbf{T}} f_2)$ vs $(g_1*f_1) \times_{\mathbf{T}} (g_2*f_2)$

Proposition 7 *Both the Manes and the Giry category satisfy the ♯-condition, provided the monoid H which comes with the respective monads is commutative.*

Proof: 1. The first condition is readily established for both monads.
2. Let $R_i : A_i \to \mathbf{M}(B_i), S_i : B_i \to \mathbf{M}(C_i)\ (i = 1, 2)$ be morphisms with $\mathbf{M}$ as the functor underlying the Manes monad. Then these equalities hold for $\langle a_1, a_2 \rangle \in A_1 \times A_2$:

$$((S_1 \times_{\mathbf{M}} S_2)*(R_1 \times_{\mathbf{M}} R_2))(a_1, a_2) = \{\langle h_1h_2h_3h_4, c_1, c_2\rangle | \exists b_1, b_2 : \langle h_1, b_1\rangle \in R_1(a_1), \langle h_2, b_2\rangle \in R_2(a_2), \langle h_3, c_1\rangle \in S_1(b_1), \langle h_4, c_2\rangle \in S_2(b_2)\},$$

and

$$((S_1*R_1) \times_{\mathbf{M}} (S_2*R_2))(a_1, a_2) = \{\langle h_1h_2h_3h_4, c_1, c_2\rangle | \exists b_1, b_2 : \langle h_1, b_1\rangle \in R_1(a_1), \langle h_2, c_2\rangle \in S_1(b_1), \langle h_3, b_2\rangle \in R_2(a_2), \langle h_4, c_2\rangle \in S_2(b_2)\},$$

3. Let $K_i : A_i \to \mathbf{G}(B_i), L_i : B_i \to \mathbf{G}(C_i)\ (i = 1, 2)$ be morphisms with $\mathbf{G}$ as the functor underlying the Manes monad. Here A_i, B_i, C_i are measurable spaces, the monoid H is assumed to be measurable as well. Since

$$((L_1 \times_{\mathbf{G}} L_2)*(K_1 \times_{\mathbf{G}} K_2))(a_1, a_2)$$

is a finite measure on $H \times C_1 \times C_2$, and so is

$$((L_1*K_1) \times_{\mathbf{M}} (L_2*K_2))(a_1, a_2),$$

it is sufficient for establishing equality to show that the integrals for an arbitrary measurable and bounded function $\psi : H \times C_1 \times C_2 \to \mathbb{R}$ coincide. A calculation using Fubini's Theorem

on product integration establishes that

$$\int_{H\times C_1\times C_2} \psi \; d\left((L_1 \times_{\mathbf{G}} L_2)*(K_1 \times_{\mathbf{G}} K_2)\right)(a_1,a_2) =$$
$$\int_{H\times B_1}\int_{H\times B_2}\int_{H\times C_1}\int_{H\times C_2} \psi(h_1h_2h_3h_4,c_1,c_2)\; L_2(b_2)(d\langle h_4,c_2\rangle)\; L_1(b_1)(d\langle h_3,c_1\rangle)\times$$
$$\times\; K_2(a_2)(d\langle h_2,b_2\rangle)\; K_1(a_1)(d\langle h_1,b_1\rangle)$$

and

$$\int_{H\times C_1\times C_2} \psi \; d\left((L_1 \times_{\mathbf{G}} L_2)*(K_1 \times_{\mathbf{G}} K_2)\right)(a_1,a_2) =$$
$$\int_{H\times B_1}\int_{H\times C_1}\int_{H\times B_2}\int_{H\times C_2} \psi(h_1h_2h_3h_4,c_1,c_2)\; L_2(b_2)(d\langle h_4,c_2\rangle)\; K_2(a_2)(d\langle h_3,b_2\rangle)\times$$
$$\times\; L_1(b_2)(d\langle h_2,c_1\rangle)\; K_1(a_1)(d\langle h_1,b_1\rangle)$$

4. These equalities establish the claim. It is interesting to observe in which way in both cases the roles of h_2 and h_3 get interchanged, reflecting the way in which morphisms change positions. □

An easy induction using the second assertion in Lemma 1 establishes that the Kleisli identity on $a_1\times\cdots\times a_n$ can be calculated through the identities on the components. The $\sharp$-condition makes also sure that we may shift computations between products (the easy inductive proof is left to the reader):

Lemma 4 *Assume that the $\sharp$-condition holds. Then*

1. *The equality*
$$\eta_{a_1\times\cdots\times a_n} = \eta_{a_1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \eta_{a_n}$$
holds for all objects $a_1,\ldots,a_n$ in $\mathfrak{X}$,

2. *If $\sigma_i : a_i \to \mathbf{T}b_i$ and $\tau_i : b_i \to \mathbf{T}c_i$ are morphisms in $\mathfrak{X}$, then*

$$\begin{aligned}(\tau_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_n) * (\sigma_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \sigma_n) =& \\ (\tau_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_{j-1} \times_{\mathbf{T}} (\tau_j{*}\sigma_j) \times_{\mathbf{T}} \tau_{j+1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_n) *& \\ * \left(\sigma_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \sigma_{j-1} \times_{\mathbf{T}} \eta_{a_j} \times_{\mathbf{T}} \sigma_{j+1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \sigma_n\right) =& \\ \left(\tau_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_{j-1} \times_{\mathbf{T}} \eta_{b_j} \times_{\mathbf{T}} \tau_{j+1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_n\right) *& \\ * (\sigma_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \sigma_{j-1} \times_{\mathbf{T}} (\tau_j{*}\sigma_j) \times_{\mathbf{T}} \sigma_{j+1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \sigma_n)&\end{aligned}$$

□

The equations in part 2 of Lemma 4 are useful in our context: $\sigma_1\times_{\mathbf{T}}\ldots\times_{\mathbf{T}}\sigma_n$ and $\tau_1\times_{\mathbf{T}}\ldots\times_{\mathbf{T}}\tau_n$ represent the computations in consecutive blocks of a PF-system. Then we may shift the computation of a component out of a block into the next or the previous one without changing the result; shifting means among others replacing the morphism by the appropriate identity. We will use this observation in the proof of Prop. 8 for establishing the invariance result. From now on we assume that the $\sharp$-condition is satisfied.

In fact, we can say more about representing $\times_{\mathbf{T}}$-products of morphisms: they can be written as Kleisli-products of a very special kind. The discerning reader will no doubt observe that the kind of representation derived from the discussion that follows will not be needed for the present constructions of PF-systems. It appears to be interesting, nevertheless.

Definition 9 *Assume $n > 1$, let $\tau_i : a_i \to \mathbf{T}b_i$ morphisms for $1 \leq i \leq n$, and let ξ be a permutation of $\{1, \ldots, n\}$. Then $\langle \sigma_1, \ldots, \sigma_n \rangle$ is* the ξ-expansion of $\langle \tau_1, \ldots, \tau_n \rangle$ *iff σ_j can be written as $\zeta_{j,1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \zeta_{j,n}$ such that*

1. *each $\zeta_{j,i}$ is either η_{a_i}, η_{b_i} or one of $\tau_1, \ldots, \tau_n$,*
2. *$\zeta_{j,k} \in \{\tau_1, \ldots, \tau_n\}$ iff $\xi(j) = k$,*
3. *if $\zeta_{j,k} = \tau_i$, then*

$$\zeta_{\ell,k} = \begin{cases} \eta_{a_i}, & \ell > j \\ \eta_{b_i}, & \ell < j. \end{cases}$$

For example, the permutation $(13)(2)$ of $\{1,2,3\}$ corresponds to

$$\begin{pmatrix} \zeta_{1,1} & \zeta_{1,2} & \zeta_{1,3} \\ \zeta_{2,1} & \zeta_{2,2} & \zeta_{2,3} \\ \zeta_{3,1} & \zeta_{3,2} & \zeta_{3,3} \end{pmatrix} = \begin{pmatrix} \eta_{b_1} & \eta_{b_2} & \tau_3 \\ \eta_{b_1} & \tau_2 & \eta_{a_3} \\ \tau_1 & \eta_{a_2} & \eta_{a_3} \end{pmatrix}$$

Thus if $\langle \sigma_1, \ldots, \sigma_n \rangle$ is a ξ-expansion of $\langle \tau_1, \ldots, \tau_n \rangle$, then assuming $\xi(j) = i$, σ_j can be written as

$$\eta_{b_1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \eta_{b_{i-1}} \times_{\mathbf{T}} \tau_i \times_{\mathbf{T}} \eta_{a_{i+1}} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \eta_{a_n}$$

indicating that τ_i is doing its work, whereas $\tau_1, \ldots, \tau_{i-1}$ did do their work already (thus the identity on the range is incorporated) and that $\tau_{i+1}, \ldots, \tau_n$ will still have to do their work (hence the identity of the respective domains are incorporated into the $\times_{\mathbf{T}}$-product).

Lemma 5 *Let under the assumptions of Def. 9 $\langle \sigma_1, \ldots, \sigma_n \rangle$ be an ξ-expansion of $\langle \tau_1, \ldots, \tau_n \rangle$, then*

$$\tau_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_n = \sigma_1 * \ldots * \sigma_n$$

holds.

Proof: 1. The proof proceeds by induction on n. For $n = 2$ the only ξ-expansions of $\langle \tau_1, \tau_2 \rangle$ are $\langle \eta_{b_1} \times_{\mathbf{T}} \tau_2, \tau_1 \times_{\mathbf{T}} \eta_{a_2} \rangle$ and $\langle \tau_1 \times_{\mathbf{T}} \eta_{b_2} \eta_{a_1} \times_{\mathbf{T}} \tau_2 \rangle$. The $\sharp$-conditions then permits directly establishing the claim.
2. The inductive step considers the ξ-expansion $\langle \sigma_1, \ldots, \sigma_{n+1} \rangle$ of $\langle \tau_1, \ldots, \tau_{n+1} \rangle$. Then $\xi(j_{n+1}) = n + 1$, and we can write

$$\sigma_i = \begin{cases} \sigma'_i \times_{\mathbf{T}} \eta_{b_{n+1}}, & i < j_{n+1} \\ \sigma'_i \times_{\mathbf{T}} \eta_{a_{n+1}}, & i > j_{n+1} \end{cases}$$

for some σ'_i. It is easy to see that

$$\langle \sigma'_1, \ldots, \sigma'_{j_{n+1}-1}, \sigma'_{j_{n+1}+1}, \ldots, \sigma_{n+1} \rangle$$

is an ξ'-expansion for $\langle \tau_1, \ldots, \tau_n \rangle$, where ξ' is the permutation of $\{1, \ldots, n\}$ derived from ξ. Now write $\sigma_{j_{n+1}} = \eta_{c_1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \eta_{c_n}$ where $c_i \in \{a_i, b_i\}$ is suitably chosen according to the

definition of the expansion, then we have by the induction hypothesis, by the $\sharp$-condition, and by Lemma 4

$$\begin{aligned}
\sigma_1 * \ldots * \sigma_{n+1} &= \left(\sigma'_1 \times_{\mathbf{T}} \eta_{b_{n+1}}\right) * \ldots * \left(\sigma'_{j_{n+1}-1} \times_{\mathbf{T}} \eta_{b_{n+1}}\right) * \\
&\quad * \sigma_{j_{n+1}} * \left(\sigma'_{j_{n+1}+1} \times_{\mathbf{T}} \eta_{a_{n+1}}\right) * \ldots * \left(\sigma'_{n+1} \times_{\mathbf{T}} \eta_{a_{n+1}}\right) \\
&= \left(\sigma'_1 * \ldots * \sigma'_{j_{n+1}-1} * \left(\eta_{c_1} \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \eta_{c_n}\right)\right) \times_{\mathbf{T}} \tau_{n+1} * \\
&\quad * \left(\sigma'_{j_{n+1}+1} * \ldots * \sigma'_{n+1}\right) \times_{\mathbf{T}} \eta_{a_{n+1}} \\
&= \left(\sigma'_1 * \ldots * \sigma'_{j_{n+1}-1} * \eta_{c_1 \times \cdots \times c_n} * \sigma'_{j_{n+1}+1} * \ldots * \sigma'_{n+1}\right) \times_{\mathbf{T}} \tau_{n+1} \\
&= \left(\sigma'_1 * \ldots * \sigma'_{j_{n+1}-1} * \sigma'_{j_{n+1}+1} * \ldots \sigma'_{n+1}\right) \times_{\mathbf{T}} \tau_{n+1} \\
&= \tau_1 \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \tau_{n+1}.
\end{aligned}$$

This establishes the claim. □
Now assume that n is an inner node in $\mathcal{G}$ with

$$\mathfrak{a}(\gamma, n) : \prod\{\gamma_{\langle k,n\rangle} | k \in \bullet n\} \to \mathbf{T}\left(\prod\{\gamma_{\langle n,k\rangle} | k \in n\bullet\}\right)$$

as its label, and assume that the edge $\langle k, n\rangle$ is replaced by the edges $\langle k, q\rangle, \langle q, n\rangle$ for some $q \in Q$. The new edges are labelled through the object $\gamma_{\langle k,n\rangle}$, and the new node n carries the label $\eta_{\gamma_{\langle k,n\rangle}}$. Other edges leading into node n are also replaced. The net effect of inserting a node just in front of node n is replacing $\mathfrak{a}(\gamma, n)$ by

$$\mathfrak{a}(\gamma, n) * \eta_{\prod\{\gamma_{\langle k,n\rangle} | k \in \bullet n\}}$$

which equals of course $\mathfrak{a}(\gamma, n)$. Similarly, replacing an edge $\langle n, k\rangle$ by edges $\langle n, q\rangle, \langle q, k\rangle$ and introducing labels on edges and on $q \in Q$ accordingly has the effect of replacing $\mathfrak{a}(\gamma, n)$ by

$$\eta_{\prod\{\gamma_{\langle n,k\rangle} | k \in n\bullet\}} * \mathfrak{a}(\gamma, n),$$

equalling $\mathfrak{a}(\gamma, n)$, too. This is a translation of the idea of inserting "neutral" nodes into the graph in order to render it stratified. In fact, two Q-extensions to $\mathcal{G}$ differ only by such neutral nodes on paths between nodes taken from $\mathcal{G}$.

Proposition 8 *Suppose $\langle \mathcal{G}, \gamma\rangle$ is a PF-system with $\langle \mathcal{G}_1, \gamma\rangle$ and $\langle \mathcal{G}_2, \gamma\rangle$ as Q-extensions. Then*

$$\mathfrak{P}(\gamma, \mathcal{G}_1) = \mathfrak{P}(\gamma, \mathcal{G}_2)$$

holds.
Proof: 1. The proof proceeds by induction on

$$N := \max\{\Lambda(\mathcal{G}_1), \Lambda(\mathcal{G}_2))\}.$$

The j$^{\text{th}}$ partition element of graph $\mathcal{G}_i$ will denoted by $S_j^{(i)}$.
2. The induction starts at $N = 2$. This step inspects each node n of $\mathcal{G}$ in turn. Suppose $n \in \left(S_1^{(1)} \setminus S_1^{(2)}\right) \cap V$, then, since graph $\mathcal{G}_2$ is an Q-extension to $\mathcal{G}$, for each predecessor w

of n in $\mathcal{G}_1$ there exists a node $q_w \in Q$ such that $\langle w, q_w\rangle, \langle q_w, n\rangle$ are edges in $\mathcal{G}_2$ which are labelled by the object $\gamma_{\langle w,n\rangle}$; node q_w itself carries the label $\eta_{\gamma_{\langle w,n\rangle}}$. Lemma 4 implies that the morphism $\mathfrak{a}(\gamma, n)$ which participates in defining $\mathfrak{P}(\gamma, \mathcal{G}_2)$ has a factor

$$\eta_{\gamma_{\langle w_1,n\rangle}\times\cdots\times\gamma_{\langle w_r,n\rangle}}$$

to the right, where $w_1, \ldots, w_r$ are in that order all predecessors of n in $\mathcal{G}_1$. A similar argument applies to $n \in \left(S_1^{(2)} \setminus S_1^{(1)}\right) \cap V$, so that $\mathfrak{P}(\gamma, \mathcal{G}_1)$ differs only by factors from $\mathfrak{P}(\gamma, \mathcal{G}_2)$ which are identity Kleisli morphisms. Hence the assertion holds for $N = 2$.

3. Let $\max\{\Lambda(\mathcal{G}_1), \Lambda(\mathcal{G}_2)\} = N+1$. We may and do assume w.l.g. that $V \cap \left(S_1^{(2)} \cup S_1^{(1)}\right) \neq \emptyset$, for, otherwise no node of V is directly connected to a root in either extension, so we may construct new graphs by eliminating the respective sets S_1 without changing the work of either graph.

We construct from the PF-system $\langle \mathcal{G}_1, \gamma\rangle$ a PF-system $\langle \mathcal{G}_3, \gamma\rangle$ which is an Q-extension to $\langle \mathcal{G}, \gamma\rangle$ such that

$$S_1^{(3)} = \{n \in V | \exists w \in W : w \to_Q^* n \text{ in } \mathcal{G}_1\} \cup \{n \in V | \exists w \in W : w \to_Q^* n \text{ in } \mathcal{G}_2\},$$

where $\to_Q^*$ indicates that there exists a (unique) path of non-negative length that runs — with the exception of the endpoints — entirely through Q. Moreover,

$$\mathfrak{P}(\gamma, \mathcal{G}_1) = \mathfrak{P}(\gamma, \mathcal{G}_3)$$

will hold.

Initially, $\langle \mathcal{G}_3, \gamma\rangle := \langle \mathcal{G}_1, \gamma\rangle$. Assume $n \in V \cap S_1^{(1)}$ such that $n \in S_t^{(2)}$ for some $t > 1$. Let $w_1, \ldots, w_r$ be all predecessors to n in $\mathcal{G}_1$. Since $\mathcal{G}_2$ is an Q-extension, there exist nodes $q_{1,2}, \ldots, q_{1,t-1}, \ldots, q_{r,2}, \ldots, q_{r,t-1}$ in Q such that

$$\begin{array}{ccc} w_1 = q_{1,1} & \ldots & q_{1,t} = n \\ \vdots & & \vdots \\ w_r = q_{r,1} & \ldots & q_{r,t} = n \end{array}$$

form paths that run with the exception of their endpoints entirely through Q. The edges on the i^{th} path are labelled with the object $\gamma_{\langle w_i,n\rangle}$, and $\mathfrak{a}(\gamma, q_{i,j}) = \eta_{\gamma_{\langle w_i,n\rangle}}$.

Let $k_1, \ldots, k_s$ be all successors to n in $\mathcal{G}_2$. Remove the nodes $\{q_{i,j} | 1 \leq i \leq r, 2 \leq i \leq t-1\}$ and the edges, including $\langle w_i, q_{i,2}\rangle$ and $\langle q_{i,t-1}, n\rangle$ for $1 \leq i \leq r$ from $\mathcal{G}_3$, and add nodes $q'_{1,2}, \ldots, q'_{1,t-1}, \ldots, q'_{s,2}, \ldots, q'_{s,t-1}$ as well as edges so that we have the paths

$$\begin{array}{ccc} n = q'_{1,1} & \ldots & q'_{1,t} = k_1 \\ \vdots & & \vdots \\ n = q'_{s,1} & \ldots & q'_{s,t} = k_s. \end{array}$$

Put $\gamma_{\langle q'_{i,j}, q'_{i,j+1}\rangle} := \gamma_{\langle n,k_i\rangle}$, and set $\gamma(q'_{i,j}) := \eta_{\gamma_{\langle n,k_i\rangle}}$ $(i > 1, j < t)$.

Consequently, graph $\mathcal{G}_3$ remains an Q-extension to $\mathcal{G}$, and from Lemma 4, part 2, we see that $\mathfrak{P}(\gamma, \mathcal{G}_1) = \mathfrak{P}(\gamma, \mathcal{G}_3)$ holds. Working in this way through $V \cap \left(S_1^{(1)} \cup S_1^{(2)}\right)$ will eventually produce the desired graph.

In the same manner we construct a PF-system $\langle \mathcal{G}_4, \gamma \rangle$ with $S_1^{(4)} = S_1^{(2)}$ such that $\mathfrak{P}(\gamma, \mathcal{G}_2) = \mathfrak{P}(\gamma, \mathcal{G}_4)$ holds.
Remove the roots from $\mathcal{G}_3$; this yields the graph $\widetilde{\mathcal{G}_3}$ which is an Q-extension to

$$\widetilde{\mathcal{G}} := (V \setminus W, E \cap (V \setminus W \times V \setminus W)) .$$

Similarly, remove the roots from $\mathcal{G}_4$ yielding $\widetilde{\mathcal{G}_4}$, which is also an Q-extension to $\widetilde{\mathcal{G}}$. Since

$$\max\{\Lambda(\widetilde{\mathcal{G}_3}), \Lambda(\widetilde{\mathcal{G}_4})\} \leq N,$$

the induction hypothesis applies, so that

$$\begin{aligned} \mathfrak{P}(\gamma, \mathcal{G}_1) &= \mathfrak{P}(\gamma, \mathcal{G}_3) \\ &= \mathfrak{P}\left(\gamma, \widetilde{\mathcal{G}_3}\right) *\mathfrak{A}\left(S_1^{(3)}, \gamma\right) \\ &= \mathfrak{P}\left(\gamma, \widetilde{\mathcal{G}_4}\right) *\mathfrak{A}\left(S_1^{(4)}, \gamma\right) \\ &= \mathfrak{P}(\gamma, \mathcal{G}_4) \\ &= \mathfrak{P}(\gamma, \mathcal{G}_2) \end{aligned}$$

holds. □
This proposition shows that the work described by an Q-extension of a PF-system does only depend on the underlying PF-system, so that we are now in a position to define the work of such a system — which need not be stratified — through its stratified step-twins.

Definition 10 *We define the work $\mathfrak{P}(\gamma, \mathcal{G})$ being done by the PF-system $\langle \mathcal{G}, \gamma \rangle$ as the work $\mathfrak{P}(\gamma, \mathcal{G}_1)$ of one of its Q-extensions $\langle \mathcal{G}_1, \gamma \rangle$.*

Bisimilarity is investigated in Sect. 5 for stratified systems, and Prop. 5 shows that bisimilarity is preserved: when the nodes of two systems are bisimilar, then the work being done by these systems is bisimilar. This result is readily extended to general PF-systems, as will be shown now:

Proposition 9 *Let $\langle \mathcal{G}, \gamma_1 \rangle$ and $\langle \mathcal{G}, \gamma_2 \rangle$ be bisimilar PF-systems, and let the overall work be performed by the functor $\mathfrak{P}(\gamma_|, \mathcal{G}) : \mathfrak{i}_j \to \mathfrak{o}_j$ for $j = 1, 2$. Then the work performed by both systems is bisimilar:*

$$\langle \mathfrak{i}_1, \mathfrak{o}_1, \mathfrak{P}(\gamma_1, \mathcal{G}) \sim \langle \mathfrak{i}_2, \mathfrak{o}_2, \mathfrak{P}(\gamma_2, \mathcal{G}) .$$

Proof: Let $\langle \mathcal{G}', \gamma_j \rangle$ be Q-extensions to $\langle \mathcal{G}, \gamma_j \rangle$ for $j = 1, 2$. The the construction implies that $\langle \mathcal{G}', \gamma_1 \rangle$ is bisimilar to $\mathcal{G}', \gamma_2 \rangle$, because by Lemma 4 $\langle a, a, \eta_a \rangle \sim \langle b, b, \eta_b \rangle$ holds for objects a, b in $\mathfrak{X}$. Thus the assertion follows from Prop. 5 in conjunction with Prop. 8. □
The usefulness of this construction is indicated by modelling two operations on PF-systems, viz., concatenation and substitution. Concatenation takes two PF-systems and connects them, substitution (a.k.a. *refinement*) takes a PF-system, and replaces a node in it by another PF-system. Both operations are vital in composing systems from smaller ones, so that larger systems can be built up through a suitable sequence of them.

Concatenation Let $\langle \mathcal{G}_1, \gamma \rangle$ and $\langle \mathcal{G}_2, \chi \rangle$ be two PF-systems, $\mathcal{G}_i = \langle V_i, E_i \rangle$. The idea in concatenating both is to pipe the output from the first system to the input of the second one, hence

$$V_1 \cap V_2 = B_1 = W_2$$

should hold: the output nodes from the first system should coincide with the input nodes for the second one; otherwise, these systems do not share nodes. Neither an input node nor an output node carries any functionality in our model, but by lumping them together, we may wish to perform some work (combining pipes often requires some transformation, e.g., of formats, between input and output). Hence we assume for each node $n \in B_1 = W_2$ the existence of a Kleisli morphism

$$\mathfrak{a}\,(\tau, n) : \prod \{\gamma_{\langle k,n \rangle} | \langle k, n \rangle \in E_1\} \to \mathbf{T}\left(\prod \{\chi_{\langle n,j \rangle} | \langle n, j \rangle \in E_2\}\right).$$

This permits defining the *τ-concatenation* $\langle \mathcal{G}_1, \gamma \rangle +_\tau \langle \mathcal{G}_2, \chi \rangle$ as $\langle \mathcal{H}, \kappa \rangle$ with

$$\begin{aligned}
\mathcal{H} &:= \langle V_1 \cup V_2, E_1 \cup E_2 \rangle, \\
\kappa_{\langle k,n \rangle} &:= \begin{cases} \gamma_{\langle k,n \rangle}, & \langle k, n \rangle \in E_1, \\ \chi_{\langle k,n \rangle}, & \text{otherwise}, \end{cases} \\
\mathfrak{a}\,(\kappa, n) &:= \begin{cases} \mathfrak{a}\,(\gamma, n)\,, & n \in V_1 \setminus (W_1 \cup B_1), \\ \mathfrak{a}\,(\tau, n)\,, & n \in B_1, \\ \mathfrak{a}\,(\chi, n)\,, & n \in V_2 \setminus (W_2 \cup B_2). \end{cases}
\end{aligned}$$

We get as a corollary from Prop. 8:

Corollary 3 *Under the conditions above, $\langle \mathcal{H}, \kappa \rangle := \langle \mathcal{G}_1, \gamma \rangle +_\tau \langle \mathcal{G}_2, \chi \rangle$ is a PF-system, and*

$$\mathfrak{P}\,(\kappa, \mathcal{H}) = \mathfrak{P}\,(\chi, \mathcal{G}_2) * (\mathfrak{a}\,(\tau, n_1) \times_{\mathbf{T}} \ldots \times_{\mathbf{T}} \mathfrak{a}\,(\tau, n_k)) * \mathfrak{P}\,(\gamma, \mathcal{G}_1)\,,$$

where $B_1 = W_2 = \{n_1, \ldots, n_k\}$. □

Thus τ provides the glue for composing the PF-systems, and the work being done exhibits the work performed when combining both systems. The glue alluded at here is different from but similar in function to the glue introduced in [30].

Substitution Systems are often built through successive stages of refinements, where a part of a system is first represented as a node, and this node is then replaced in subsequent steps by an entire subsystem. This is formalized in a graph-theoretic context through graph grammars [22, Ch. 3]. We will focus on a special case.
Let $\mathcal{G}_i = \langle V_i, E_i \rangle$ be dags with respective roots W_i and leaves B_i, and let $n \in V_1$ be a node such that (dots taken in $\mathcal{G}_1$)

$$\bullet n = W_2, n\bullet = B_2.$$

Thus an incoming edge for n comes from a root in $\mathcal{G}_2$, and an outgoing edge goes to a leaf in $\mathcal{G}_2$. For technically simplifying the representation, we assume that only the nodes in $W_2 \cup B_2$ are common to V_1 and V_2. We assume further that we have a selection map

$$\psi : W_2 \cup B_2 \to V_2 \setminus (W_2 \cup B_2)$$

which will help constructing new edges when absorbing $\mathcal{G}_2$ into $\mathcal{G}_1$ by associating with each root or leaf an inner node as source or target of an edge, as we will see. We require that $\psi[W_2] \cap \psi[B_2] = \emptyset$, since otherwise cycles in the replacement graph would result. Define the *ψ-replacement*

$$\mathcal{G}_1[\mathcal{G}_2 \backslash_\psi n]$$

of node n through graph $\mathcal{G}_2$ as the graph $\langle U, D \rangle$ by

$$\begin{aligned} U &:= (V_1 \setminus \{n\}) \cup V_2, \\ D &:= \left(E_1 \cap (V_1 \setminus \{n\})^2\right) \cup E_2 \cup \\ &\quad \{\langle w, \psi(w)\rangle | w \in W_2\} \cup \{\langle \psi(b), b\rangle | b \in B_2\}. \end{aligned}$$

Thus we build the new graph by combining all nodes with the exception of n, the node to be replaced. All edges leading into n or out of it are removed, and replaced by edges into $\mathcal{G}_2$: if $\langle w, n\rangle \in E_1$ is an edge in $\mathcal{G}_1$, the node w must be a root in $\mathcal{G}_2$, then this edge will be replaced in the replacement graph by the edge $\langle w, \psi(w)\rangle$, similarly for edges $\langle n, b\rangle \in E_1$. Since the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ do not have cycles, and since ψ assigns by assumption different nodes to roots and to leaves, $\langle U, D\rangle$ does not have any cycles either.
We apply this construction to PF-systems now. Suppose that in addition to the assumptions made so far $\langle \mathcal{G}_2, \gamma\rangle$ and $\langle \mathcal{G}_2, \chi\rangle$ are PF-systems. For getting our machinery going, the new edges need labels from $\mathfrak{X}$; these edges should not violate the typing constraints imposed on node n. Call the selection map ψ *viable* iff

$$\forall w \in W_2 : \gamma_{\langle w,n\rangle} = \chi_{\langle w,\psi(w)\rangle} \wedge \forall b \in B_2 : \gamma_{\langle n,b\rangle} = \chi_{\langle \psi(b),b\rangle}$$

holds. Hence the Kleisli morphisms $\mathfrak{a}(\gamma, n)$ and $\mathfrak{P}(\chi, \mathcal{G}_2)$ have the same signatures.
We define the PF-system

$$\langle \mathcal{G}_1[\mathcal{G}_2 \backslash_\psi n], \gamma[\chi \backslash_\psi n]\rangle$$

in the obvious way by taking the values γ, and χ for edges in E_1 or in E_2, resp., depending on where they come from, and by setting for the new edges

$$\gamma[\chi \backslash_\psi n]_{\langle w,\psi(w)\rangle} := \gamma_{\langle w,n\rangle},$$

similarly for $\gamma[\chi \backslash_\psi n]_{\langle \psi(b),b\rangle}$. The labels for the nodes are left unchanged, coming either from γ or from χ. Then Prop. 8 implies that we may compute the work for the composed system in these steps:

- compute $\mathfrak{P}(\chi, \mathcal{G}_2)$, hence the work of the system which is to refine node n,
- substitute for $\mathfrak{a}(\gamma, n)$ the morphism $\mathfrak{P}(\chi, \mathcal{G}_2)$, leaving the rest of γ alone; technically: form $\gamma[\mathfrak{P}(\chi, \mathcal{G}_2) \backslash n]$,
- compute the work done by the PF-system based on graph $\mathcal{G}_1$ with the modified value for γ.

Formally:

Corollary 4 *Let the selection map ψ be viable, then*

$$\mathfrak{P}(\gamma[\chi \backslash_\psi n], \mathcal{G}_1[\mathcal{G}_2 \backslash_\psi n]) = \mathfrak{P}(\gamma[\mathfrak{P}(\chi, \mathcal{G}_2) \backslash n], \mathcal{G}_1).$$

□

8 Comparison And Conclusion

We will briefly have a look at modelling PF-systems through the specification language Z, and we will see what points architectural modelling for `Mobile UNITY` through categories are stressed. Then we will give some hints at further work in this area.
The modelling of pipes and filters through the specification language Z summarized and discussed e.g. in [28] is evidently much closer to an implementation than the approach proposed in the present paper. Thus a person intending to implement a system with such an architecture is probably better off looking at a Z-specification, using well-known refinement techniques like the ones discussed by Spivey [29, Ch. 5] for coming even closer to a realization as a running system.
The difference to the present approach, however, lies deeper: Shaw at al. emphasize in discussing software architectures the first class rank of architectural connectors [28, 1, 25, 26, 27]. This implies that filters and pipes are treated on the same eye-level. The scenario here marks a contrast: connectors are represented through objects in a category, components through morphisms of a rather special kind, putting these two kinds of entities on different levels. It may much be said in favor of dealing uniformly with connectors and with components, but it seems that an asymmetric treatment helps the intuition: computations are conceptually different from data transport, however complex the latter may be. The present approach reflects an approach like "Tell me, what your data are, then we will talk about computations on them", much like in object-oriented software construction.
The approach proposed by Fiadeiro et al, see e.g. [30, 13], using categories for modelling architectures shows how different kinds of functors, in particular interface functors, may be put to use for constructing systems. This is illustrated in [30] where e diagram is "compiled" through computing its colimit, leading to the early version of a program. Moreover, fundamental kinds of interactions of program components are studied using the patterns constructed in that paper. The focus lies on modelling just the interactions for a particular class of mobile programs, emphasizing the importance of connectors: "Software Architecture has put forward the concept of connector to express complex relationships between system components, thus facilitating the separation of coordination from computation." is the very first sentence in the abstract. The computation proper, however, has not been addressed, and this is what we propose in the present paper. Reflecting the mobile nature of the programs discussed in [30], and taking into account that no fixed topology is available for the computing nodes in such a scenario, another difference becomes visible: the topology of the communication and the direction of data flow remains fixed here but may be subject to change dynamically in a mobile context. But this is a completely different story, since PF-systems exhibit a fixed structure by their very nature.
Program evolution is supported by concatenating, and by hierarchically composing PF-systems. While the first operation is easily modelled in the Z-approach, only a hint at supporting the hierarchical composition is given in [1], making it difficult to compare both approaches in this respect.

Further Work The present paper excludes those PF-systems that are cyclic; further work should remove this restriction. It is challenging to see how other architectural styles are tackled, and how to model dynamically changing communication topologies. On the relational side, we have narrowed down monads which represent the two major kinds of relations, albeit a gap remains for measurable set-valued relations, as indicated at the end of Sect. 3. The

natural transformations θ and the $\sharp$-condition seem to be ingredients to a monad which models relations, and further work will address this common area of relational types.

References

[1] G. Abowd, R. Allen, and D. Garlan. Using style to understand descriptions of software architecture. In D. Notkin, editor, *SIGSOFT'93*, volume 18 of *Software Engineering Notes*, pages 9 – 20. ACM, Dec 1993.

[2] S. Abramsky, R. Blute, and P. Panangaden. Nuclear and trace ideal in tensored *-categories. *Journal of Pure and Applied Algebra*, 143(1 – 3):3 – 47, 1999.

[3] P. Aczel and N. Mendler. A final coalgebra theorem. In H. H. Pitt, A. Poigne, and D. E. Rydeheard, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357 – 365, 1989.

[4] M. Barr and C. Wells. *Category Theory for Computing Science*. Les Publications CRM, Montreal, third edition, 1999.

[5] C. Brink, W. Kahl, and G. Schmidt, editors. *Relational Methods in Computer Science*. Advances in Computing Science. Springer-Verlag, Wien, New York, 1997.

[6] D. Cantone, E. G. Omodeo, and A. Policriti. *Set Theory for Computing*. Springer-Verlag, 2001. In print.

[7] J. Demongeot and F. Leitner. Compact set-valued flows I: Applications in medical imaging. *C. R. Acad. Sci., Paris, Ser. II*, b 323(11):747 – 754, 1996.

[8] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation of labelled markov-processes. Technical report, School of Computer Science, McGill University, Montreal, 1998.

[9] E.-E. Doberkat. Good state transition policies for nondeterministic and stochastic automata. *Information and Control*, 46:135 – 155, 1980.

[10] E.-E. Doberkat. The converse of a probabilistic relation. Technical Report 113, Chair for Software Technology, University of Dortmund, June 2001.

[11] E.-E. Doberkat. The demonic product of probabilistic relations. In Mogens Nielsen and Uffe Engberg, editors, *Proc. Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 113 – 127, Berlin, 2002. Springer-Verlag.

[12] E.-E. Doberkat, F. Schmidt, and C. Veltmann. Re-engineering the German integrated system for measuring and assessing environmental radioactivity. *Environmental Modelling & Software*, 15:267 – 278, 2000.

[13] J. L. Fiadeiro and T. Maibaum. A mathematical toolbox for the software architect. In J. Kramer and A. Wolf, editors, *Proc. 8th Intl. Workshop on Software Specification and Design*. IEEE Computer Society Press, 1996.

[14] M. Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, volume 915 of *Lecture Notes in Mathematics*, pages 68 – 85, Berlin, 1981. Springer-Verlag.

[15] C. J. Himmelberg. Measurable relations. *Fund. Math.*, 87:53 – 72, 1975.

[16] A. S. Kechris. *Classical Descriptive Set Theory*. Number 156 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, Heidelberg, New York, 1994.

[17] S. Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, 2 edition, 1997.

[18] T. Lorenz. Set-valued maps for image segmentation. In *ALGORITMY 2000*, Conference on Scientific Computing, pages 65 – 173, 2000.

[19] N. Medvidovics, D. S. Rosenblum, D. F. Redmiles, and J. E. Robbins. Modeling software architectures in the Unified Modeling Language. *ACM Trans. Softw. Eng. Method.*, 11(1):2 – 57, January 2002.

[20] E. Moggi. An abstract view of programming languages. Lecture Notes, Stanford University, June 1989.

[21] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55 – 92, 1991.

[22] M. Nagl, editor. *Building Tightly Integrated Software Development Environments: The IPSEN Approach*, volume 1170 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[23] P. Panangaden. Probabilistic relations. In C. Baier, M. Huth, M. Kwiatkowska, and M. Ryan, editors, *Proc. PROBMIV*, pages 59 – 74, 1998. Also available from the School of Computer Science, McGill University, Montreal.

[24] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3 – 80, 2000. Special issue on modern algebra and its applications.

[25] M. Shaw. The coming-of-age of software architecture research. In *Proc. 23rd International Conference on Software Engineering*, pages 656 – 664, 2001.

[26] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik. Abstractions for software architecture and tools to support them. *IEEE Trans. Softw. Eng.*, 21(4):314 – 335, April 1995.

[27] M. Shaw and D. Garlan. Formulations and formalisms in software architecture. In J. van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[28] M. Shaw and D. Garlan. *Software Architecture — Perspectives on an Emerging Discipline*. Prentice-Hall, 1996.

[29] J. M. Spivey. *The Z Notation — A Reference Manual*. Prentice-Hall, 1989.

[30] M. Wermelinger and J. L. Fiadeiro. Connectors for mobile programs. *IEEE Trans. Softw. Eng.*, 24(5):331 – 341, May 1998.

/99/ T. Bühren, M. Cakir, E. Can, A. Dombrowski, G. Geist, V. Gruhn, M. Gürgrn, S. Handschumacher, M. Heller, C. Lüer, D. Peters, G. Vollmer, U. Wellen, J. von Werne
Endbericht der Projektgruppe eCCo (PG 315)
Electronic Commerce in der Versicherungsbranche
Beispielhafte Unterstützung verteilter Geschäftsprozesse
Februar 1999

/100/ A. Fronk, J. Pleumann,
Der DoDL-Compiler
August 1999

/101/ K. Alfert, E.-E. Doberkat, C. Kopka
Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art
September 1999

/102/ E.-E. Doberkat
An Note on a Categorial Semantics for ER-Models
November 1999

/103/ Christoph Begall, Matthias Dorka, Adil Kassabi, Wilhelm Leibel, Sebastian Linz, Sascha Lüdecke, Andreas Schröder, Jens Schröder, Sebastian Schütte, Thomas Sparenberg, Christian Stücke, Martin Uebing, Klaus Alfert, Alexander Fronk, Ernst-Erich Doberkat
Abschlußbericht der Projektgruppe PG-HEU (326)
Oktober 1999

/104/ Corina Kopka
Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme
März 2000

/105/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe IPSI
April 2000

/106/ Ernst-Erich Doberkat
Die Hofzwerge — Ein kurzes Tutorium zur objektorientierten Modellierung
September 2000

/107/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
November 2000

/108/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe IPSI
Februar 2001

/109/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
Februar 2001

/110/ Eugenio G. Omodeo, Ernst-Erich Doberkat
Algebraic semantics of ER-models from the standpoint of map calculus.
Part I: Static view
März 2001

/111/ Ernst-Erich Doberkat
An Architecture for a System of Mobile Agents
März 2001

/112/ Corina Kopka, Ursula Wellen
Development of a Software Production Process Model for Multimedia CAL Systems by Applying Process Landscaping
April 2001

/113/ Ernst-Erich Doberkat
The Converse of a Probabilistic Relation
June 2001

/114/ Ernst-Erich Doberkat, Eugenio G. Omodeo
Algebraic semantics of ER-models in the context of the calculus of relations.
Part II: Dynamic view
Juli 2001

/115/ Volker Gruhn, Lothar Schöpe (Eds.)
Unterstützung von verteilten Softwareentwicklungsprozessen durch integrierte Planungs-, Workflow- und Groupware-Ansätze
September 2001

/116/ Ernst-Erich Doberkat
The Demonic Product of Probabilistic Relations
September 2001

/117/ Klaus Alfert, Alexander Fronk, Frank Engelen
Experiences in 3-Dimensional Visualization of Java Class Relations
September 2001

/118/ Ernst-Erich Doberkat
The Hierarchical Refinement of Probabilistic Relations
November 2001

/119/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer, Ingo Röpling,
Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Intermediate Report
November 2001

/120/ Volker Gruhn, Ursula Wellen
Autonomies in a Software Process Landscape
Januar 2002

/121/ Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2001
des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
Februrar 2002

/122/ Ernst-Erich Doberkat, Gregor Engels, Jan Hendrik Hausmann, Mark Lohmann, Christof Veltmann
Anforderungen an eine eLearning-Plattform — Innovation und Integration —
April 2002

/123/ Ernst-Erich Doberkat
Pipes and Filters: Modelling a Software Architecture Through Relations
Juni 2002

/124/ Volker Gruhn, Lothar Schöpe
Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen
Juni 2002