

# Structural Operational Semantics for Heterogeneously Typed Coalgebras



Harald König  

Fachhochschule für die Wirtschaft Hannover, Germany

Western Norway University of Applied Sciences, Bergen, Norway

Uwe Wolter  

University of Bergen, Norway

Tim Kräuter  

Western Norway University of Applied Sciences, Bergen, Norway

---

## Abstract

Concurrently interacting components of a modular software architecture are heterogeneously structured behavioural models. We consider them as coalgebras based on different endofunctors. We formalize the composition of these coalgebras as specially tailored segments of distributive laws of the bialgebraic approach of Turi and Plotkin. The resulting categorical rules for structural operational semantics involve many-sorted algebraic specifications, which leads to a description of the components together with the composed system as a single holistic behavioural system. We evaluate our approach by showing that observational equivalence is a congruence with respect to the algebraic composition operation.

**2012 ACM Subject Classification** Theory of computation → Semantics and reasoning

**Keywords and phrases** Coalgebra, Bialgebra, Structural operational semantics, Compositionality

**Digital Object Identifier** 10.4230/LIPIcs.CALCO.2023.7

**Funding** *Harald König*: The author thanks the University of Bergen for support of this project.

**Acknowledgements** The authors thank the anonymous referees for their helpful suggestions that have helped to improve this article.

## 1 Introduction

In a modular and component-based software architecture of a compound system the individual components interact concurrently. Categorically, these individual state-based components are modelled as *coalgebras*. However, in a landscape of multiple interacting systems these behavioural models are *heterogeneously* typed: There are deterministic or non-deterministic labelled transition systems as well as probabilistic systems, systems with or without termination, with or without output and so on, see [24], Chapter 3. Hence the coalgebras of the individual components are based on different endofunctors.

Reasoning about the correct behaviour of a compound system often requires establishing correctness of each local component and furthermore using theoretical means, which guarantee that global behaviour is determined by local behaviours. In [9], this modular method is called *compositionality* and a precise formulation of it requires the use of a framework, which captures the operational semantics of concurrent processes. Such a framework is given by transition rules of structural operational semantics (SOS). Conditional rules of the form

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{\text{op}(x, y) \xrightarrow{c} \text{op}(x', y')} \quad (1)$$



© Harald König, Uwe Wolter, and Tim Kräuter;

licensed under Creative Commons License CC-BY 4.0

10th Conference on Algebra and Coalgebra in Computer Science (CALCO 2023).

Editors: Paolo Baldan and Valeria de Paiva; Article No. 7; pp. 7:1–7:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

generate systems, whose states are closed terms over an *algebraic* signature [1]. Well-known rule formats are GSOS<sup>1</sup> [3] and tyft/tyxt [8]. For some of these rule formats one can prove compositionality to hold, whereas counterexamples can be provided for other formats [8].

In this paper, we propose a formal structure, which describes the composition of *heterogeneously typed* coalgebras with the help of structural operational semantics. For this, it is important to provide a suitable rule format, which guarantees compositionality (and hopefully other similar requirements) in heterogeneous environments. Since we deal with supposedly arbitrarily varying behavioural specifications, we need more general rule formats, which cannot be expected to be homogeneous like GSOS or tyft/tyxt. An adequate generalization of transition rules in the context of coalgebraic specifications are *natural transformations* between functors, whose domain and codomain reflect the transition from  $n (\geq 2)$  local systems to one compound system, i.e., functors of type  $\mathcal{SET}^n \rightarrow \mathcal{SET}$ . We will show that these so-called *interaction laws* (see Def. 12) can be embedded into distributive laws

$$\lambda : \vec{\Sigma} \vec{\mathcal{B}} \Rightarrow \vec{\mathcal{B}} \vec{\Sigma}$$

for suitable endofunctors  $\vec{\Sigma}$  and  $\vec{\mathcal{B}}$ . Distributive laws are part of a *bialgebraic* approach, which has been described in [14], but was originally proposed by Turi and Plotkin [27]. Here  $\vec{\mathcal{B}}$  (and also  $\vec{\Sigma}$ ) is a  $\mathcal{SET}^{n+1}$ -endofunctor, which simultaneously covers the behaviours of the  $n$  heterogeneously typed coalgebras *and* a specification of the compound system, which has to comprise the commonalities of the local system behaviours. The algebraic syntax functor  $\vec{\Sigma}$  contains the operation(s), which realize(s) the transition from the local components (input of the operation) to the global view of the composed system (output). We evaluate our approach by proving compositionality to hold for interaction laws.

Whereas in process algebras like CCS or CSP<sup>2</sup> this transfer of observational indistinguishability during syntactical build-up of process terms has to be guaranteed [8], we rather want compositionality, when individual software components are composed into a global compound network. Whereas [14] circumscribes compositionality as observational equivalence (w.r.t. final semantics) being a congruence (i.e. the coinductive extension is an algebra homomorphism), we propose a slightly adapted definition tailored to the specific situation of heterogeneously typed interacting systems.

Our work was inspired by practical scenarios, where the coupling of behavioural models with other executable models like test runners or event injectors is of crucial importance [19]. Furthermore, recently, systematic approaches to co-simulation for large-scale system assessment have gained popularity [20]. Here, a typical scenario is the interaction with probabilistic systems [2, 26], which requires a concrete language for their interaction [5, 19]. While these DSLs<sup>3</sup> are already well-established, they lack theoretical underpinning in the form of transition rules to reason about correctness properties.

Hence, we answer the main question

*How can we apply (parts of) the bialgebraic theory to understand the interaction of heterogeneously typed behavioural components?*

by providing the following contributions and novelties:

- A proof for the preservation of observational equivalence, when  $n$  local components are based on *different* behavioural specifications  $\mathcal{B}_1, \dots, \mathcal{B}_n$ , by embedding interaction laws into distributive laws.

<sup>1</sup> *General Structured Operational Semantics*

<sup>2</sup> Calculus of Communicating Systems [21], Communicating Sequential Processes [10]

<sup>3</sup> Domain Specific Languages

- An ensemble of  $n$  separated individual components together with the specification of its composition is formalised in one holistic *many-sorted* approach, i.e., as coalgebras for an endofunctor  $\vec{\mathcal{B}} : \mathcal{SET}^{n+1} \rightarrow \mathcal{SET}^{n+1}$ .

The paper is organized as follows: Sect. 2 clarifies notation, Sect. 3 presents the general setting based on practical scenarios as well as a motivating example, Sect. 4 recapitulates the survey [14] in some detail to make the content complete and comprehensible, and Sect. 5 presents the above mentioned novelties in detail: The linkage of the definitions of Interaction Law (in Def.12) and Congruence (adapted to the heterogeneous case in Def. 17) yield an adequate definition of compositionality and we can obtain our main statements: Theorem 21 proves compositionality to hold for interaction laws and Corollary 22 adapts the statement of the theorem to practical needs.

## 2 Basic Notation

We use the following notations:  $\mathcal{SET}$  is the category of sets and total mappings. For two sets  $A$  and  $X$  we write  $X^A$  for the set of all total maps from  $A$  to  $X$ . A special set is  $1$ , which denotes any singleton set, e.g.  $(1 + X)^A$  is the set of all partial maps from  $A$  to  $X$ .

For functors we will use calligraphic letters like  $\mathcal{F}$ ,  $\mathcal{G}$ , and, especially, letter  $\mathcal{B}$  for behavioural and greek letter  $\Sigma$  for algebraic specifications. Categories are denoted  $\mathbb{C}$  or  $\mathbb{D}$ , an application of a functor  $\mathcal{F} : \mathbb{C} \rightarrow \mathbb{D}$  will be written  $\mathcal{F}(X)$  for  $X \in |\mathbb{C}|$  (or short  $X \in \mathbb{C}$ ), the collection of objects of  $\mathbb{C}$ , whereas an application of  $\mathcal{F}$  to a morphism  $\alpha : X \rightarrow Y$  does not use parentheses:  $\mathcal{F}\alpha : \mathcal{F}(X) \rightarrow \mathcal{F}(Y)$ . Composition of functors  $\mathcal{F}$  and  $\mathcal{G}$  (if it is possible) is always written  $\mathcal{G}\mathcal{F}$  ( $\mathcal{G}$  applied after  $\mathcal{F}$ ). Special functors are  $\mathcal{ID}_{\mathbb{C}}$ , the identity functor on  $\mathbb{C}$ , and  $\wp_{fin}$ , the powerset functor assigning to a set the set of its finite subsets.

It is often convenient to give a definition (on objects) of a functor without explicitly naming its formal parameters, e.g. a functor  $\mathcal{B}$  mapping a set  $X$  to the set  $(1+X)^A$  (see above) is often denoted  $\mathcal{B} = (1 + \_)^A$ . Furthermore, when we give the complete definition of functors  $\mathcal{F}$ , we often combine object and morphism mapping by writing  $X \xrightarrow{f} Y \mapsto \mathcal{F}(X) \xrightarrow{\mathcal{F}f} \mathcal{F}(Y)$ .

As usual, a natural transformation  $\nu$  between functors  $\mathcal{F}$  and  $\mathcal{G}$  with common domain and codomain, written  $\nu : \mathcal{F} \Rightarrow \mathcal{G}$ , is a family  $(\nu_X : \mathcal{F}(X) \rightarrow \mathcal{G}(X))_{X \in |\mathbb{C}|}$  compatible with morphism mapping. For appropriate functors  $\mathcal{H}$  and  $\mathcal{H}'$  we denote with  $\mathcal{H}\nu$  the family  $(\mathcal{H}\nu_X : \mathcal{H}\mathcal{F}(X) \rightarrow \mathcal{H}\mathcal{G}(X))_{X \in |\mathbb{C}|}$  and with  $\nu_{\mathcal{H}'}$  the family  $(\nu_{\mathcal{H}'(X)} : \mathcal{F}\mathcal{H}'(X) \rightarrow \mathcal{G}\mathcal{H}'(X))_{X \in |\mathbb{C}|}$ .

For an endofunctor  $\mathcal{B} : \mathbb{C} \rightarrow \mathbb{C}$  a  $\mathcal{B}$ -coalgebra is a  $\mathbb{C}$ -morphism  $X \xrightarrow{\alpha} \mathcal{B}(X)$ , called the structure map and written  $(X, \alpha)$  or - if  $X$  is clear from the context - just  $\alpha$ . A coalgebra morphism from  $(X, \alpha)$  to  $(Y, \beta)$  is a  $\mathbb{C}$ -morphism  $f : X \rightarrow Y$  such that  $\beta \circ f = \mathcal{B}f \circ \alpha$ . Instead of  $f$  we sometimes write  $(f, \mathcal{B}f)$  to stress the fact that commutativity involves  $\mathcal{B}f$ , as well. The resulting category of all coalgebras for  $\mathcal{B} : \mathbb{C} \rightarrow \mathbb{C}$  will be denoted  $\mathcal{B}\text{-Coalg}$ . If it admits a final object  $(Z, \zeta)$  and if  $(X, \alpha) \in \mathcal{B}\text{-Coalg}$ , we denote with  $u_\alpha : X \rightarrow Z$  the *coinductive extension* of  $\alpha$ , i.e. the unique  $\mathcal{B}\text{-Coalg}$ -morphism into the final object.

Likewise for an endofunctor  $\Sigma : \mathbb{C} \rightarrow \mathbb{C}$  a  $\Sigma$ -algebra is a  $\mathbb{C}$ -morphism  $\Sigma(X) \xrightarrow{a} X$  written  $(a, X)$ . An algebra morphism from  $(a, X)$  to  $(b, Y)$  is a  $\mathbb{C}$ -morphism  $f : X \rightarrow Y$  such that  $b \circ \Sigma f = f \circ a$ . Instead of  $f$  we sometimes write  $(\Sigma f, f)$ . The resulting category of all algebras will be denoted  $\Sigma\text{-Alg}$ .

For morphisms in combination with cartesian products, we use the following notations: If  $f : A \rightarrow B$  and  $g : A \rightarrow B'$ , then  $\langle f, g \rangle : A \rightarrow B \times B'$  denotes the uniquely determined resulting morphism. Likewise for  $g : A' \rightarrow B'$ ,  $f \times g : A \times A' \rightarrow B \times B'$ .

### 3 General Setting and Example

Multiple interacting components of software architectures collectively realize the requirements of business domains. Describing the interactions between these systems and checking their global behavioural consistency is a general, well-known challenge in software engineering [4]. To address this challenge, model-driven software engineering utilizes abstract representations of the constituting systems and their interactions. Such a setting thus consists of an ensemble of *heterogeneously structured* components, which must guarantee the desired global behaviour.

In the sequel, we will speak of *local* or *individual* components, which are assembled into a *global* or *compound* system. As in [24], "system" is also used as a superordinate term for all kinds of artifacts, whether they are composite or not.

Using a general and formal coordination language for the interaction of behavioural components in the form of transition rules requires agreement on key concepts of behavioural systems. It turns out that the concepts "State" and (observational) "State Change" are common to almost all behavioural specifications, cf. the introductory remarks of [12]. Coalgebras  $(X, \alpha)$  for some endofunctor  $\mathcal{B} : \mathbb{C} \rightarrow \mathbb{C}$  on some category  $\mathbb{C}$  comprise exactly these concepts: The structure map  $\alpha$  assigns to each  $x$  in the state space  $X$  the observable causality exhibited in state  $x$ . The different natures of causalities (behaviour) are specified by different endofunctors  $\mathcal{B}$ .

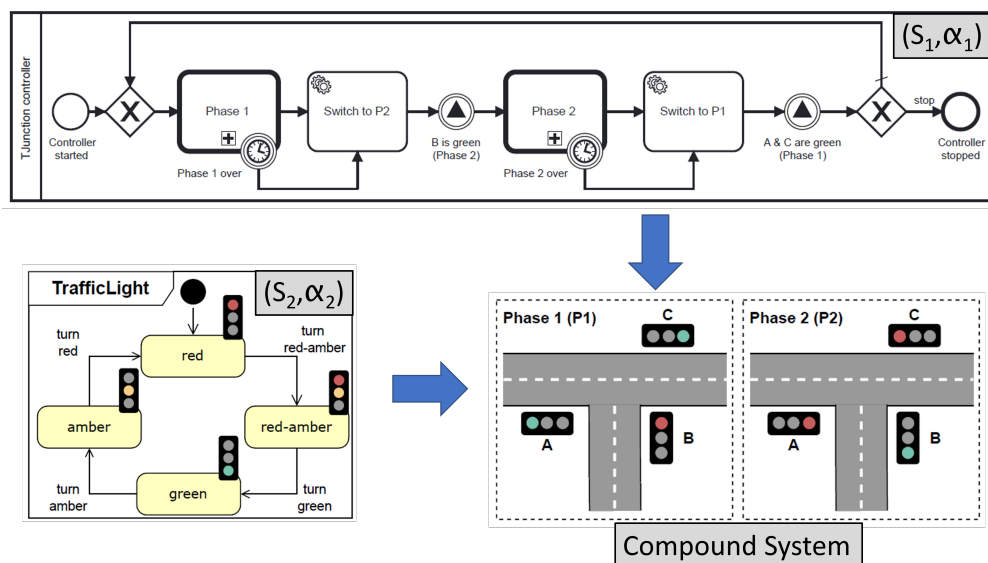
Towards a formal underpinning for the described setting, we need to understand how aligning individual components by specifying their interactions on the one hand, and automatic generation (computation) of global execution behaviour of the compound system, on the other hand, are carried out. For this, we assume  $n$  behavioural specifications  $\mathcal{B}_i : \mathcal{SET} \rightarrow \mathcal{SET}$  to be given for some  $n \geq 2$  and fix individual behavioural systems  $(S_i, \alpha_i) \in \mathcal{B}_i\text{-Coalg}$ .

As an example, we refer to the use case depicted in Fig. 1, where an instance of a **T-Junction-Controller** regulates the interaction of three **TrafficLights** A, B, and C. The T-junction controller (component  $(S_1, \alpha_1)$ ) and the behaviour of one traffic light (e.g., component  $(S_2, \alpha_2)$ ) are shown in the top and the bottom left part of Fig. 1. The resulting compound system is hinted at in the bottom right part. The operation, which takes as input the local components and "generates" the semantics of the compound system is visualized by arrows between the systems (blue in a colour display).

Whereas each traffic light is specified as a labelled transition system, the **TJunction** may be modelled as a BPMN<sup>4</sup>-model. The BPMN model specifies different phases to handle ( $P1$  and  $P2$ ). They are shown in the BPMN model and also in the two different snapshots of the compound system. The interaction with approaching vehicles may be modelled with a third formalism, e.g., a probabilistic transition system, which simulates exponentially distributed arrivals of buses or cars at one of the traffic lights. Aligning individual components by means of coordination languages, cf. [5], requires specifying coordination points (communication channels), e.g., if a request  $e$  of some approaching bus triggers the switch to phase 2 in the **TJunction** controller (an observation  $o$ ), this transition must synchronize with input  $i = \text{turnRed}$  of traffic light A and C. Moreover, B must simultaneously turn green. These synchronisations can be formalized with *synchronization algebras*, cf. [22], in this simple case, a partial map  $\varphi : O \times I \rightarrow \text{Act}$ , where  $O$  is the set of outputs of the controller like throw events or service calls in automatic tasks,  $I$  is the set of possible inputs to the respective traffic light, and  $\text{Act}$  is the set of observable actions of the compound system.

---

<sup>4</sup> Business Process Model and Notation



■ **Figure 1** TJunction traffic control system and its individual components.

The global execution behaviour can be described by *premises* (transitions of the local components) and *conclusions* (the resulting actions taken by the compound system). If, for instance,  $x \xrightarrow{e/o} x'$  in the BPMN-model and  $y \xrightarrow{i} y'$  are possible, then  $z \xrightarrow{\varphi(o,i)} z'$  is a global interaction of the components. Note that we obtain a respective conditional rule, but with different formats in its premises and conclusions: There is the Mealy-like notation in the first premise specifying output  $o$  in the BPMN process, when event  $e$  occurs, whereas the second premise specifies that the labelled transition system behaves like  $y'$ , if, in state  $y$ , input  $i$  occurred. Furthermore, the compound system may be non-deterministic, such that the conclusion reads, “The system *may* behave like  $z'$ , if in state  $z$ , action  $\varphi(o, i)$  was performed”.

Formally the interaction operation, which takes as input  $n$  states of the local components and outputs a state of the compound system, is based on an  $n$ -ary operation symbol  $\text{op} = \text{interact} : s_1 s_2 \dots s_n \rightarrow s_{n+1}$  of a suitable algebraic signature  $\Sigma$ , where sorts  $s_1, s_2, \dots$  reflect the structurally separated but interacting local components, and the compound system is based on a new behavioural specification  $\mathcal{B}$  and requires a new sort  $s_{n+1} \notin \{s_1, \dots, s_n\}$ .

We summarise the transfer from practical concepts to the bialgebraic formalism:

1. The individual components are based on behavioural endofunctors  $\mathcal{B}_1, \dots, \mathcal{B}_n$  and the compound system's behaviour is specified by another endofunctor  $\mathcal{B}$ . The individual components are coalgebras  $(S_i, \alpha_i) \in \mathcal{B}_i\text{-Coalg}$ , from which the states of the compound system  $(S, \alpha) \in \mathcal{B}\text{-Coalg}$  arise as output of an application of an  $n$ -ary algebraic operation  $\text{op}$  which has as input the states of the individual components.
2. The semantics of the compound system is formalized by SOS rules of the form

$$\frac{x_1 \xrightarrow{E_1} x'_1 \dots x_n \xrightarrow{E_n} x'_n}{\text{op}(x_1, \dots, x_n) \xrightarrow{F} \text{op}(x'_1, \dots, x'_n)}$$

where  $E_i$  and  $F$  are differently structured terms over the coordination points, and  $x_i, x'_i$  are states of the individual component  $(S_i, \alpha_i)$  for all  $i \in \{1, \dots, n\}$ .

#### 4 Background: Distributive Laws and Bialgebras

In this section, we recapitulate parts of [14] which are necessary to make the content of Sect. 5 complete and comprehensible. As in [18], we extend behavioural specifications  $\mathcal{B}$  only to copointed coalgebras (see Def. 2 below), i.e. we consider the assignment  $X \mapsto X \times \mathcal{B}(X)$  instead of  $\mathcal{B}(X)$  in order to be able to use current states in the formulation of the conclusion of SOS rules. However, we do not generalise it further, i.e. we neither make use of free extensions for the algebraic specification functor as in the original work [27] nor cofree extensions of the behaviour functor, cf. [14].

Let  $\mathbb{C}$  be an arbitrary category with finite products. The classical theory works with one fixed behavioural specification  $\mathcal{B} : \mathbb{C} \rightarrow \mathbb{C}$  and an algebraic specification  $\Sigma : \mathbb{C} \rightarrow \mathbb{C}$  which usually specifies the syntactical assembly of process terms (such as prefixing, alternative, and parallel, as well as interaction). In contrast to that, in Sect. 5, we will use  $\Sigma$  for the assembly of the compound system from the local components. Furthermore, let's define the functor

$$\mathcal{H} : \begin{cases} \mathbb{C} & \rightarrow & \mathbb{C} \\ X \xrightarrow{f} Y & \mapsto & X \times \mathcal{B}(X) \xrightarrow{f \times \mathcal{B}f} Y \times \mathcal{B}(Y). \end{cases} \quad (2)$$

Pairs  $(\mathcal{H} : \mathbb{C} \rightarrow \mathbb{C}, \varepsilon : \mathcal{H} \Rightarrow ID_{\mathbb{C}})$  are usually called *copointed* functors in the literature, e.g. [14], i.e.  $\mathcal{H}$  comes equipped with a comonadic "counit"  $\varepsilon$ . In our particular definition  $\mathcal{H}$  is accompanied with counit  $\pi_1 : \mathcal{H} \Rightarrow ID_{\mathbb{C}}$ , where  $\pi_1 = ((\pi_1)_X : X \times \mathcal{B}(X) \rightarrow X)$  is the componentwise first projection. We will use only these special copointed functors.

► **Definition 1** (Distributive Law over  $\mathcal{H}$ ). *A Distributive Law of  $\Sigma$  over  $\mathcal{H}$  is a natural transformation*

$$\lambda : \Sigma\mathcal{H} \Rightarrow \mathcal{H}\Sigma$$

*which is compatible with the counit, i.e. such that  $(\pi_1)_{\Sigma} \circ \lambda = \Sigma\pi_1 : \Sigma\mathcal{H} \Rightarrow \Sigma$ .* ┘

The extension from the original behavioural specification functor  $\mathcal{B}$  to  $\mathcal{H}$  also requires to consider special coalgebras for  $\mathcal{H}$  [18]:

► **Definition 2** (Copointed  $\mathcal{H}$ -Coalgebra). *Let  $\mathcal{H}$  be given as above. The category of copointed  $\mathcal{H}$ -coalgebras, written  $\mathcal{H}\text{-Coalg}_{co}$ , is the full subcategory of  $\mathcal{H}\text{-Coalg}$  with those objects  $(X, \alpha)$  satisfying  $(\pi_1)_X \circ \alpha = id_X$ .* ┘

► **Proposition 3** (Copointed  $\mathcal{H}$ -Coalgebras are  $\mathcal{B}$ -Coalgebras). *The assignment  $(X, \alpha) \mapsto (X, (\pi_2)_X \circ \alpha)$  extends to an isomorphism between categories  $\mathcal{H}\text{-Coalg}_{co}$  and  $\mathcal{B}\text{-Coalg}$ .* ┘

There is a canonical assignment from distributive laws over  $\mathcal{H}$  to natural transformations

$$\rho : \Sigma\mathcal{H} \Rightarrow \mathcal{B}\Sigma \quad (3)$$

given by  $\lambda \mapsto (\pi_2)_{\Sigma} \circ \lambda$ . Using counit compatibility from Def. 1, the assignment

$$\rho \mapsto \langle \Sigma\pi_1, \rho \rangle \quad (4)$$

turns out to be inverse to the former, see Theorem 10 in [18]. Thus

► **Proposition 4** (Equivalent Representation of Distributive Laws). *The assignments (3) and (4) yield a bijection between distributive laws over  $\mathcal{H}$  and natural transformations  $\rho : \Sigma\mathcal{H} \Rightarrow \mathcal{B}\Sigma$ .* ┘

Note that natural transformations as in (3) are special cases of GSOS laws, where the syntax functor  $\Sigma$  is replaced by its free extension  $\Sigma^*$  in the codomain of  $\rho$ , thus enabling arbitrary terms in the target of the SOS rule conclusion.

Because  $\lambda$  is a natural transformation,

$$\Sigma_\lambda : \begin{cases} \mathcal{H}\text{-Coalg}_{co} & \rightarrow & \mathcal{H}\text{-Coalg}_{co} \\ (X, h) & \mapsto & (\Sigma(X), \lambda_X \circ \Sigma h) \end{cases}$$

and its dual construction

$$\mathcal{H}^\lambda : \begin{cases} \Sigma\text{-Alg} & \rightarrow & \Sigma\text{-Alg} \\ (g, X) & \mapsto & (\mathcal{H}g \circ \lambda_X, \mathcal{H}(X)) \end{cases}$$

extend to endofunctors, where the first indeed maps to  $\mathcal{H}\text{-Coalg}_{co}$  by the compatibility of counits in Def. 1.  $\Sigma_\lambda$  applied to a coalgebra yields behaviour of algebraically composed states and will play a major role in Sect. 5.  $\mathcal{H}^\lambda$  will be used only in the present section.

For a distributive law  $\lambda$ , there is a new category, which yields a combination of operational and denotational models w.r.t. functors  $\mathcal{B}$  (and thus  $\mathcal{H}$ ) and  $\Sigma$ :

► **Definition 5** (Category of  $\lambda$ -Bialgebras). *Let  $\lambda : \Sigma\mathcal{H} \Rightarrow \mathcal{H}\Sigma$  be a distributive law according to Def.1. The category  $\lambda$ -Bialg has objects arrow-pairs  $\Sigma(X) \xrightarrow{g} X \xrightarrow{h} \mathcal{H}(X)$  with copointed  $(X, h)$  and for which*

$$\mathcal{H}g \circ \lambda_X \circ \Sigma h = h \circ g \tag{5}$$

Morphisms are those  $f : X \rightarrow Y$ , which are simultaneously  $\Sigma\text{-Alg}$ - and  $\mathcal{H}\text{-Coalg}_{co}$ -morphisms.

Using (5), one obtains

► **Proposition 6** ([14], Prop. 12). *There are the isomorphisms*

$$\Sigma_\lambda\text{-Alg} \cong \lambda\text{-Bialg} \cong \mathcal{H}^\lambda\text{-Coalg}$$

where e.g. the second one is based on the assignment

$$(\Sigma(X) \xrightarrow{g} X \xrightarrow{h} \mathcal{H}(X)) \mapsto ((g, X) \xrightarrow{(\Sigma h, h)} \mathcal{H}^\lambda(g, X))$$

on objects of the respective categories. ┘

A consequence of this fact is the following proposition, for which we include a proof, because we need parts of it in Sect. 5:

► **Proposition 7** (Initial and Final Bialgebras, [14], Sect. 4.3). *If  $\Sigma$  admits an initial algebra  $(a, A)$  and if  $\mathcal{H}\text{-Coalg}_{co}$  has the final (copointed) coalgebra  $(Z, \zeta)$ , then the former uniquely extends to an initial object  $\Sigma(A) \xrightarrow{a} A \xrightarrow{h_\lambda} \mathcal{H}(A)$  of  $\lambda$ -Bialg and the latter uniquely extends to a final object  $\Sigma(Z) \xrightarrow{g^\lambda} Z \xrightarrow{\zeta} \mathcal{H}(Z)$  of  $\lambda$ -Bialg.*

**Proof.** By Prop. 6 we can look for an initial object in  $\mathcal{H}^\lambda\text{-Coalg}$ . But for any endofunctor  $\overline{\mathcal{H}} : \mathbb{D} \rightarrow \mathbb{D}$  the carrier of the initial object in  $\overline{\mathcal{H}}\text{-Coalg}$  is just the initial object in  $\mathbb{D}$ , if it exists. Hence for  $\overline{\mathcal{H}} = \mathcal{H}^\lambda$  and  $\mathbb{D} = \Sigma\text{-Alg}$ , we obtain the initial  $\mathcal{H}^\lambda\text{-Coalg}$ -object  $a \xrightarrow{(\Sigma h_\lambda, h_\lambda)} \mathcal{H}^\lambda a$ , where  $h_\lambda : A \rightarrow \mathcal{H}(A)$  is the unique  $\Sigma\text{-Alg}$ -morphism out of the initial object. By the definition of  $\mathcal{H}^\lambda$  this yields the commutative diagram

$$\begin{array}{ccc} \Sigma(A) & \xrightarrow{a} & A \\ \Sigma h_\lambda \downarrow & & \downarrow h_\lambda \\ \Sigma \mathcal{H}(A) & \xrightarrow{\mathcal{H}a \circ \lambda_A} & \mathcal{H}(A) \end{array} \tag{6}$$

## 7:8 SOS for Heterogeneously Typed Coalgebras

turning  $\Sigma(A) \xrightarrow{a} A \xrightarrow{h_\lambda} \mathcal{H}(A)$  into a  $\lambda$ -*Bialg*-object (because  $h_\lambda$  is copointed by the following Prop. 8) but also the *initial*  $\lambda$ -*Bialg*-object due to the assignment given in Prop. 6. The unique extension of the final object is dually obtained yielding the final  $\lambda$ -bialgebra  $\Sigma(Z) \xrightarrow{g^\lambda} Z \xrightarrow{\zeta} \mathcal{H}(Z)$  for the unique  $\mathcal{H}$ -*Coalg*<sub>co</sub>-morphism  $g^\lambda$  to the final object. ◀

Using counit compatibility of  $\pi_1$  in Def. 1 and initiality of  $(a, A)$ , one also obtains

► **Proposition 8** (Copointedness of  $h_\lambda$ ).  *$h_\lambda$  is a copointed  $\mathcal{H}$ -coalgebra.* ◻

The initial and final bialgebras from the proof of Prop. 7 yield a unique arrow  $f : A \rightarrow Z$  in the commutative diagram

$$\begin{array}{ccccc} \Sigma(A) & \xrightarrow{a} & A & \xrightarrow{h_\lambda} & \mathcal{H}(A) \\ \Sigma f \downarrow & & \downarrow f & & \downarrow \mathcal{H}f \\ \Sigma(Z) & \xrightarrow{g^\lambda} & Z & \xrightarrow{\zeta} & \mathcal{H}(Z) \end{array} \quad (7)$$

$f$  is simultaneously the coinductive extension of  $h_\lambda$ , i.e. its behavioural semantics, and the inductive extension of  $g^\lambda$ , i.e. the evaluation of  $\Sigma$ -terms in  $(g_\lambda, Z)$ . The former statement is the important one. It gives the key statement of this section:

► **Observation 9.** The coinductive extension of copointed  $h_\lambda$  is an algebra homomorphism from the initial  $\Sigma$ -algebra. ◻

## 5 From Local Components to Compound Systems Coalgebraically

### 5.1 The Theoretical Setting

Let  $(S_1, \alpha_1) \in \mathcal{B}_1\text{-Coalg}, \dots, (S_n, \alpha_n) \in \mathcal{B}_n\text{-Coalg}$  be  $n$  individual, local components and  $\mathcal{B}$  a behavioural specification for the compound system, see item 1 in the summary on page 5. Related local components' interactions are based on an  $n$ -ary operation symbol  $\text{op} := \text{interact}$  and coordination points of  $\mathcal{B}_1, \dots, \mathcal{B}_n$  together with a synchronisation algebra<sup>5</sup>  $\varphi$  establish transition rules, cf. item 2 of the summary.

► **Example 10** (See Sect. 3). A BPMN model can be encoded with the functor

$$\mathcal{B}_1 = (1 + O \times \_)^E,$$

where  $E$  are state-changing events like a timer event in the Tjunction Controller, cf. Fig. 1. The set  $O$  defines outputs, e.g.,  $\text{SwitchToP2} \in O$ , which must be synchronized with a call to a traffic light to turn red. Let  $(S_1, \alpha_1) \in \mathcal{B}_1\text{-Coalg}$  be such a component. Traffic lights are deterministic labelled transition systems based on

$$\mathcal{B}_2 = (1 + \_)^I,$$

where, for instance,  $I = \{\mathbf{A}.\text{turnRed}, \mathbf{A}.\text{turnGreen}, \dots\}$  is the input set  $I$  of traffic light  $\mathbf{A}$ . Let  $(S_2, \alpha_2) \in \mathcal{B}_2\text{-Coalg}$  be such a component.

<sup>5</sup> For arbitrary  $n$ , these synchronisation descriptions will no longer be binary.



We define  $Act := O + I + \{\tau\}$  and expect the compound system to change state depending on the used coordination points<sup>6</sup>. For this, we extend the involved sets by an idle action  $*$ , i.e.  $X_* := X + \{*\}$  for  $X \in \{O, I, Act\}$ , cf. [22]. As usual, a transition with  $*$  from state  $x$  to  $x'$  is possible, if and only if  $x = x'$ . We assume a synchronisation algebra  $\varphi : O_* \times I_* \rightarrow Act_*$  for the synchronisation of a BPMN model with one traffic light. In the above example, we might have  $\varphi(\text{SwitchToP2}, \text{A.turnRed}) = \tau$  (modelling a silent synchronisation), whereas some other value combinations like  $\varphi(\text{SwitchToP2}, \text{A.turnGreen})$  are undefined. Whenever an output  $o$  is *uncoupled*, i.e., whenever the first component can evolve independently from the second for a transition with  $o$ , we let  $\varphi$  be undefined for pairs  $(o, i)$  for all  $i \in I$ , and define  $\varphi(o, *) := o$ . This is true, for example, if  $o$  is an outgoing signal like "B is green", which does not change the state of any traffic light, cf. Fig. 1. Similarly  $\varphi$  is undefined for  $(o, i)$  for all  $o \in O$  and  $\varphi(*, i) = i$  for uncoupled  $i$ . Finally,  $\varphi(o, i) = * \iff o = i = *$ , cf. [22].

The underlying algebraic signature will have three sorts:  $s_1$  and  $s_2$  for the states of the two local components and  $s_3$  for the compound system. Because resulting transitions can be silent for different coordinations and hence result in non-determinism of the compound system, we define

$$\mathcal{B} := \wp_{fin}(Act \times \_) \quad \lrcorner$$

The original work of [27] shows a one-to-one correspondence between sets of GSOS laws and natural transformations. We will show how we can follow this approach along the above-stated example. When we mention SOS rules, we exemplarily use notations in the context of our examples. Furthermore, whenever we write down  $\varphi(o, i)$ , we automatically assume this value to be defined.

The family of SOS-rules

$$\left( \frac{x \xrightarrow{e/o} x' \quad y \xrightarrow{i} y'}{op(x, y) \xrightarrow{\varphi(o, i)} op(x', y')} \right)_{o \in O_*, i \in I_*} \quad (8)$$

describes the operational semantics of the compound system as a *heterogenous* interaction law. E.g., the controller's command to make traffic light A change to red is the law for  $b := \text{SwitchToP2}$  and  $i = \text{A.turnRed}$ .

In the example, the state space  $S$  of the compound system must take into consideration the original state spaces by pairing  $S_1$  and  $S_2$ , i.e., in this example

$$S = S_1 \times S_2. \quad (9)$$

Of course, in the general case,  $S$  can depend arbitrarily on the state spaces  $S_1, \dots, S_n$  of the local components.

## 5.2 Interaction Laws and Induced Coalgebra

In this section, we formalize the construction of the compound system from the local components, if its operational semantics is given as an SOS rule like in (8). This rule does not depend on concrete state spaces, hence it can be seen as an interaction law between

<sup>6</sup> As usual,  $\tau$  models silent (unobservable) transitions.

## 7:10 SOS for Heterogeneously Typed Coalgebras

systems of arbitrary state spaces  $X$  and  $Y$ . We claim that it can be encoded as a map, which decomposes into two factors, the first reflecting the premises given by the transitions of  $\alpha_1$  and  $\alpha_2$ , and a second factor  $\rho_{X,Y}$ , which reflects the conclusions:

$$X \times Y \xrightarrow{\langle id, \alpha_1 \rangle \times \langle id, \alpha_2 \rangle} X \times \mathcal{B}_1(X) \times Y \times \mathcal{B}_2(Y) \xrightarrow{\rho_{X,Y}} \mathcal{B}(X \times Y) \quad (10)$$

We first define  $\rho_{X,Y}$  in the context of our example:

► **Example 11** (Example 10 ctd). Recall that we call  $o \in O$  *coupled*, if there is  $i \in I$  such that  $\varphi(o, i)$  is defined and vice versa for  $i \in I$ . Otherwise, it is called *uncoupled*. Then, for  $\mathcal{B}_1 = (1 + O \times \_)^E$ ,  $\mathcal{B}_2 = (1 + \_)^I$ , and  $\mathcal{B} = \wp_{fin}(Act \times \_)$ , we can define

$$\begin{aligned} \rho_{X,Y}(x, f_1, y, f_2) &= \{(\varphi(o, i), (x', y')) \mid o \neq * \neq i, (o, x') \in f_1(E), y' = f_2(i)\} \\ &\cup \{(o, (x', y)) \mid (o, x') \in f_1(E), o \text{ uncoupled}\} \\ &\cup \{(i, (x, y')) \mid y' = f_2(i), i \text{ uncoupled}\} \end{aligned}$$

where  $f_1 : E \rightarrow 1 + O \times X$  and  $f_2 : I \rightarrow 1 + Y$ . It is easy to see that  $(\rho_{X,Y})_{(X,Y) \in |\mathcal{SET}|^2}$  is natural in its parameters  $X$  and  $Y$ .  $\lrcorner$

As in the classical theory, natural transformations as in Example 10 can now be used to *define* SOS-rules. For this let's define the copointed versions  $\mathcal{H}_i := \mathcal{SET} \rightarrow \mathcal{SET}$  of the functors  $\mathcal{B}_i$  as in (2) for  $i \in \{1, \dots, n\}$ . The special assignment  $(S_1, S_2) \mapsto S_1 \times S_2$  from (9) extends to a functor  $\Sigma : \mathcal{SET}^2 \rightarrow \mathcal{SET}$  which yields natural transformation  $\rho : \Sigma(\mathcal{H}_1 \times \mathcal{H}_2) \Rightarrow \mathcal{B}\Sigma : \mathcal{SET}^2 \rightarrow \mathcal{SET}$  in Example 11. However, we don't want to exclude additional dependencies, when constructing states of the compound system. E.g. additional supervising components or intermediate components like message queues may let the overall state space differ from the pure cartesian product of the local state spaces. Hence, we are interested in an arbitrary functor  $\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}$  for some  $n \geq 2$  and correspondingly adapted natural transformations. Thus the appropriate definition in our context is

► **Definition 12** (Interaction Law). *Let  $\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}$  be an arbitrary functor,  $\mathcal{B}_1, \dots, \mathcal{B}_n$ , and  $\mathcal{B}$  be  $\mathcal{SET}$ -endofunctors, and functors  $\mathcal{H}_i : \mathcal{SET} \rightarrow \mathcal{SET}$  be defined as in (2), i.e.  $\mathcal{H}_i(X) = X \times \mathcal{B}_i(X)$  for all  $X \in \mathcal{SET}$  and all  $i \in \{1, \dots, n\}$ . An interaction law is a natural transformation*

$$\rho : \Sigma(\mathcal{H}_1 \times \dots \times \mathcal{H}_n) \Rightarrow \mathcal{B}\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}. \quad \lrcorner$$

Similarly to the definition of  $\Sigma_\lambda$  in Sect. 4, this yields an assignment

$$\Sigma_\rho : \left\{ \begin{array}{l} \mathcal{B}_1\text{-Coalg} \times \dots \times \mathcal{B}_n\text{-Coalg} \longrightarrow \mathcal{B}\text{-Coalg} \\ ((S_1, \alpha_1), \dots, (S_n, \alpha_n)) \mapsto (\Sigma(S_1, \dots, S_n), \rho_{S_1, \dots, S_n} \circ \Sigma(\langle id_{S_1}, \alpha_1 \rangle, \dots, \langle id_{S_n}, \alpha_n \rangle)) \end{array} \right. \quad (11)$$

which becomes a functor, because  $\rho$  is a natural transformation. Any  $n$ -tuple  $(f_1, \dots, f_n)$  with  $f_i$  a  $\mathcal{B}_i$ -Coalg-morphism is mapped by  $\Sigma_\rho$  to the  $\mathcal{B}$ -Coalg-morphism  $\Sigma(f_1, \dots, f_n)$ .

► **Definition 13** ( $\rho$ -Induced Coalgebra). *Given  $(S_1, \alpha_1) \in \mathcal{B}_1\text{-Coalg}, \dots, (S_n, \alpha_n) \in \mathcal{B}_n\text{-Coalg}$  and an interaction law  $\rho$ , the  $\mathcal{B}$ -coalgebra  $\Sigma_\rho((S_1, \alpha_1), \dots, (S_n, \alpha_n))$  is called the  $\rho$ -induced coalgebra of  $(S_1, \alpha_1), \dots, (S_n, \alpha_n)$ . If the carrier sets are clear from the context, we just write  $\Sigma_\rho(\alpha_1, \dots, \alpha_n)$  for the  $\rho$ -induced coalgebra.*  $\lrcorner$

Hence, the  $\rho$ -induced coalgebra is the compound system arising from the local components, when an SOS rule like (8), which is reflected in interaction law  $\rho$ , is applied.

► **Example 14** (Example 10 ctd). In the example, we obtain the desired compound system, a  $\mathcal{B}$ -coalgebra with state space  $S_1 \times S_2$  behaving as specified by the local components and the SOS-laws from (8).  $\lrcorner$

### 5.3 Compositionality

Verification of correctness of composed systems should be guaranteed, if its components are already correct. Moreover, semantics preserving refactorings of local components should also preserve the semantics of the compound system. These behavioural correctness issues are often based on observational equivalence, hence we want observational equivalence to be preserved after the construction of the compound system from the local components. In this section, we formally define these aspects in the context of our setting.

► **Definition 15** (Observational Equivalence). *Let  $\mathcal{F} : \mathcal{SET} \rightarrow \mathcal{SET}$ , such that  $\mathcal{F}$ -Coalg admits a final object  $(Z, \zeta)$ . Let  $(X, \alpha) \in \mathcal{F}$ -Coalg and  $u_\alpha : X \rightarrow Z$  be its coinductive extension. Two states  $x, x' \in X$  are said to be observationally equivalent, written  $x \sim_\alpha x'$ , if  $(x, x')$  is contained in the kernel relation  $\ker(u_\alpha)$ , i.e. if  $u_\alpha(x) = u_\alpha(x')$ . ◻*

For future use, we state the following proposition, which easily follows, because for any  $\mathcal{F}$ -Coalg-morphism  $f : (X, \alpha) \rightarrow (Y, \beta)$ , the coinductive extension satisfies  $u_\alpha = u_\beta \circ f$ :

► **Proposition 16** (Observational Equivalence). *With the same ingredients as in Def. 15, two states  $x_1, x_2 \in X$  are observationally equivalent, if there is an  $\mathcal{F}$ -Coalg-morphism  $f : (X, \alpha) \rightarrow (Y, \beta)$  such that  $f(x_1) = f(x_2)$ .<sup>7</sup> ◻*

Let  $(u_{\alpha_i})_{i \in \{1, \dots, n\}}$  be the coinductive extensions of our local components,

$$\sim_i = \ker(u_{\alpha_i}), \quad (12)$$

and some operation  $\text{op} : S_1 \times \dots \times S_n \rightarrow A$  be given for some state set  $A$  of some  $\mathcal{B}$ -coalgebra. Furthermore, let  $\sim$  be the kernel relation of its coinductive extension, then preservation of observational equivalence under  $\text{op}$  means

$$\forall i \in \{1, \dots, n\} : x_i \sim_i x'_i \Rightarrow \text{op}(x_1, \dots, x_n) \sim \text{op}(x'_1, \dots, x'_n), \quad (13)$$

i.e. observational equivalence is a compatible w.r.t. operation  $\text{op}$ . It is well-known that a general definition of congruence on an algebra  $a : \mathcal{F}(A) \rightarrow A$  for an endofunctor  $\mathcal{F} : \mathbb{C} \rightarrow \mathbb{C}$  is as follows: A monomorphism  $R \xrightarrow{\langle \pi_1, \pi_2 \rangle} A \times A$  is a congruence on  $a$ , if there is an algebra  $r : \mathcal{F}(R) \rightarrow R$ , for which the diagram

$$\begin{array}{ccccc} \mathcal{F}(A) & \xleftarrow{\mathcal{F}(\pi_1)} & \mathcal{F}(R) & \xrightarrow{\mathcal{F}(\pi_2)} & \mathcal{F}(A) \\ a \downarrow & & r \downarrow & & \downarrow a \\ A & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & A \end{array} \quad (14)$$

commutes, cf. Theorem 3.3.5. in [12].

However, in the case of separated heterogeneously typed state sets of the local systems, a general definition of congruence must be based on the above-defined functor  $\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}$ .

<sup>7</sup> Note that this proposition can even better be taken as the definition for observational equivalence, because it does not depend on the existence of a final object. We found it, however, more demonstrative to use Def. 15 for it.

► **Definition 17** (*a-compatibility*). Let  $A_1, \dots, A_n, A$  be sets and

$$a : \Sigma(A_1, \dots, A_n) \rightarrow A$$

be a map. Furthermore let  $(R_i \subseteq A_i \times A_i)_{i \in \{1, \dots, n\}}$  and  $R \subseteq A \times A$  be a collection of  $n + 1$  binary relations with projections  $\pi_1^i, \pi_2^i : R_i \rightarrow A_i$  for all  $i$  and  $\pi_1, \pi_2 : R \rightarrow A$ . The relation tuple  $(R_1, \dots, R_n, R)$  is said to be *a-compatible*, if there is a map  $r$ , such that the following diagram commutes:

$$\begin{array}{ccccc} \Sigma(A_1, \dots, A_n) & \xleftarrow{\Sigma(\pi_1^1, \dots, \pi_1^n)} & \Sigma(R_1, \dots, R_n) & \xrightarrow{\Sigma(\pi_2^1, \dots, \pi_2^n)} & \Sigma(A_1, \dots, A_n) \\ \downarrow a & & \downarrow r & & \downarrow a \\ A & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & A \end{array}$$

┘

► **Example 18** (*op-compatibility*). Let  $R_i = \sim_i$  and  $R = \sim$ , cf. (12), then it is easy to see that in the case  $\Sigma(X_1, \dots, X_n) = X_1 \times \dots \times X_n$  *op-compatibility* yields (13). ┘

► **Observation 19.** *a-compatibility* of  $(R_1, \dots, R_n, R)$  can thus be read as an implication: If pairs  $(a_i, a'_i)$  are related via  $R_i$ , then *a-images* of corresponding elements of the set  $\Sigma(A_1, \dots, A_n)$  are related as well. ┘

Of course, the meaning of the term "corresponding" depends on the action of  $\Sigma$ .

It is not self-evident that observational equivalence is compatible with the syntactic structure of process terms in transition rules, see the counterexamples in [8] or violations of compositionality in the context of the  $\pi$ -calculus [21], Chapt. 12.4. However, in our setting, we can prove that interaction laws preserve observational equivalence. Note that this is almost evident in the above example, where  $n = 2$  and  $\Sigma(X, Y) = X \times Y$ , because the image of the pair of coinductive extensions  $u_{\alpha_1} : S_1 \rightarrow \dots$  and  $u_{\alpha_2} : S_2 \rightarrow \dots$  of functor  $\Sigma_\rho$  is the  $\mathcal{B}$ -coalgebra-morphism  $u = u_{\alpha_1} \times u_{\alpha_2}$ , for which  $((x_1, x_2), (x'_1, x'_2)) \in \ker(u_{\alpha_1} \times u_{\alpha_2})$ , if  $(x_1, x'_1) \in \ker(u_{\alpha_1})$  and  $(x_2, x'_2) \in \ker(u_{\alpha_2})$ , which yields the desired result by Prop. 16.

The proof idea for the general case is to keep the local state spaces and the state space for the compound system separated as systems in their own right by assigning different sorts of the underlying algebraic specification to them and then apply (7) (i.e. observation 9). For this, we must formalise the whole setting of system components and their interaction in one holistic *many-sorted* approach as follows. Recall that we assume  $(S_1, \alpha_1) \in \mathcal{B}_1\text{-Coalg}, \dots, (S_n, \alpha_n) \in \mathcal{B}_n\text{-Coalg}$  to be  $n$  individual, local components, then we define the endofunctor

$$\vec{\Sigma} : \begin{cases} \mathcal{SET}^{n+1} & \rightarrow & \mathcal{SET}^{n+1} \\ (X_1, \dots, X_n, X_{n+1}) & \mapsto & (S_1, \dots, S_n, \Sigma(X_1, \dots, X_n)) \end{cases}$$

with  $\vec{\Sigma}(h_1, \dots, h_n, h_{n+1}) := (\underbrace{id, \dots, id}_{n \text{ times}}, \Sigma(h_1, \dots, h_n))$  on function tuples. Intuitively, we define

an algebraic signature with sorts  $s_1, \dots, s_n, s_{n+1}$  and "constants" of sort  $s_i$  the elements of  $S_i$  (for  $1 \leq i \leq n$ ), as well as operation symbols with codomain  $s_{n+1}$ . Thus the term algebra has carrier sets  $S_1, \dots, S_n$ , whereas the carrier of sort  $s_{n+1}$  comprises all terms arising from a single application of an operation symbol. We obtain

► **Proposition 20** (Initial Object of  $\vec{\Sigma}$ ).  $\vec{\Sigma}\text{-Alg}$  possesses an initial object with carrier  $\mathbf{0} := (S_1, \dots, S_n, \Sigma(S_1, \dots, S_n)) = \vec{\Sigma}(\mathbf{0})$  and structure map  $id_{\mathbf{0}} : \vec{\Sigma}(\mathbf{0}) \rightarrow \mathbf{0}$ .

**Proof.** Given a  $\vec{\Sigma}$ -algebra  $(f_1, \dots, f_n, f_{n+1}) : \vec{\Sigma}(X_1, \dots, X_n, X_{n+1}) \rightarrow (X_1, \dots, X_n, X_{n+1})$ , it is easy to see that

$$\mathbf{0} \xrightarrow{(f_1, \dots, f_n, f_{n+1} \circ \Sigma(f_1, \dots, f_n))} (X_1, \dots, X_n, X_{n+1})$$

establishes the unique algebra homomorphism from  $id_{\mathbf{0}}$  to the given algebra.  $\blacktriangleleft$

► **Theorem 21** (Interaction Laws preserve Observational Equivalence). *Let  $\mathcal{B}_1, \dots, \mathcal{B}_n$ , and  $\mathcal{B}$  be  $n+1$   $\mathcal{SET}$ -endofunctors, such that all corresponding categories of coalgebras admit a final coalgebra. Let  $(S_1, \alpha_1) \in \mathcal{B}_1\text{-Coalg}, \dots, (S_n, \alpha_n) \in \mathcal{B}_n\text{-Coalg}$  and  $u_{\alpha_1}, \dots, u_{\alpha_n}$  be their coinductive extensions. For functor  $\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}$  let an interaction law  $\rho$  be given as in Def. 12, and  $\Sigma_\rho(\alpha_1, \dots, \alpha_n)$  be the  $\rho$ -induced  $\mathcal{B}$ -coalgebra, cf. (11) and Def. 13, together with its coinductive extension  $u$ . Then the family  $(\ker(u_{\alpha_1}), \dots, \ker(u_{\alpha_n}), \ker(u))$  of kernel relations is  $id_{\Sigma(S_1, \dots, S_n)}$ -compatible.*

Thus by observation 19: If pairs  $(x_i, x'_i)$  are observationally equivalent w.r.t.  $\alpha_i$ , then corresponding elements in the set  $\Sigma(S_1, \dots, S_n)$  are observationally equivalent w.r.t. the  $\rho$ -induced coalgebra  $\Sigma_\rho(\alpha_1, \dots, \alpha_n)$ . Thus observational equivalence carries over from the local components to the compound system.

**Proof.** To make reading easier, we give the proof of Theorem 21 for the special case  $n = 2$ . It easily carries over to the general case. Let  $\mathcal{H}_1, \mathcal{H}_2$ , and  $\mathcal{H}$  be the copointed versions of  $\mathcal{B}_1, \mathcal{B}_2$ , and  $\mathcal{B}$  as in (2) and with this

$$\vec{\mathcal{H}} := \mathcal{H}_1 \times \mathcal{H}_2 \times \mathcal{H} : \mathcal{SET}^3 \rightarrow \mathcal{SET}^3$$

Let  $\vec{\mathcal{B}} := \mathcal{B}_1 \times \mathcal{B}_2 \times \mathcal{B} : \mathcal{SET}^3 \rightarrow \mathcal{SET}^3$ , then we have obtained the setting of Sect. 4 with  $\mathbb{C} = \mathcal{SET}^3$ ,  $\mathcal{H} := \vec{\mathcal{H}}$ , and  $\mathcal{B} := \vec{\mathcal{B}}$ . Furthermore, we define

$$\vec{\rho} := (\alpha_1, \alpha_2, \rho) : \vec{\Sigma}\vec{\mathcal{H}} \Rightarrow \vec{\mathcal{B}}\vec{\Sigma} : \mathcal{SET}^3 \rightarrow \mathcal{SET}^3. \quad (15)$$

which is a natural transformation, because  $(\alpha_1, X_1 = \alpha_1)_{X_1 \in |\mathcal{SET}|}$  and  $(\alpha_2, X_2 = \alpha_2)_{X_2 \in |\mathcal{SET}|}$  are independent of their parameters  $X_1, X_2$ , resp. By Prop. 4 it corresponds to a distributive law  $\vec{\lambda} : \vec{\Sigma}\vec{\mathcal{H}} \Rightarrow \vec{\mathcal{H}}\vec{\Sigma}$  of  $\vec{\Sigma}$  over  $\vec{\mathcal{H}}$ , where by (4)

$$\vec{\lambda} = \langle \vec{\Sigma}\pi_1, \vec{\rho} \rangle. \quad (16)$$

Following the notation of Prop. 20, (6) becomes

$$\begin{array}{ccc} \vec{\Sigma}(\mathbf{0}) & \xlongequal{\quad} & \mathbf{0} \\ \vec{\Sigma}h_{\vec{\lambda}} \downarrow & & \downarrow h_{\vec{\lambda}} \\ \vec{\Sigma}\vec{\mathcal{H}}(\mathbf{0}) & \xrightarrow{\vec{\lambda}_0} & \vec{\mathcal{H}}(\mathbf{0}) \end{array} \quad (17)$$

The first component  $h_{\vec{\lambda}}^1$  of  $h_{\vec{\lambda}}$  is the composition of the first components of the left and bottom arrow:  $h_{\vec{\lambda}}^1 = \vec{\lambda}_0^1 \circ (\vec{\Sigma}h_{\vec{\lambda}})^1 = \langle id_{S_1}, \alpha_1 \rangle \circ id_{S_1}$ , because  $\vec{\Sigma}$  is constant in the first component and similarly for the second component, hence

$$(h_{\vec{\lambda}}^1, h_{\vec{\lambda}}^2) = (\langle id_{S_1}, \alpha_1 \rangle, \langle id_{S_1}, \alpha_2 \rangle). \quad (18)$$

Thus, the third component of  $\vec{\Sigma}h_{\vec{\lambda}}$  equals  $\Sigma(\langle id_{S_1}, \alpha_1 \rangle, \langle id_{S_2}, \alpha_2 \rangle)$ . By (16) the third component of  $\vec{\lambda}_0$  is the pair of the third component of  $(\vec{\Sigma}\pi_1)_{\mathbf{0}}$  and  $\rho_{(S_1, S_2)}$ , hence

$$h_{\vec{\lambda}}^3 = \langle id_{\Sigma(S_1, S_2)}, \rho_{S_1, S_2} \circ \Sigma(\langle id_{S_1}, \alpha_1 \rangle, \langle id_{S_2}, \alpha_2 \rangle) \rangle = \langle id_{\Sigma(S_1, S_2)}, \Sigma_\rho(\alpha_1, \alpha_2) \rangle \quad (19)$$

by Def. 13. Let  $(\mathbf{1}, \zeta)$  be the final  $\vec{B}$ -coalgebra (which exists, because it is taken componentwise), then by Prop. 3  $(\mathbf{1}, \langle id, \zeta \rangle)$  is final in  $\vec{\mathcal{H}}\text{-Coalg}_{co}$  and (7) is reflected in the left two squares in

$$\begin{array}{ccccc}
 \vec{\Sigma}(\mathbf{0}) & \xlongequal{\quad} & \mathbf{0} & \xrightarrow{h_{\vec{x}}} & \vec{\mathcal{H}}(\mathbf{0}) & \xrightarrow{(\pi_2)_0} & \vec{B}(\mathbf{0}) \\
 \vec{\Sigma}\vec{u} \downarrow & & \downarrow \vec{u} & & \downarrow \vec{\mathcal{H}}\vec{u} & & \downarrow \vec{B}\vec{u} \\
 \vec{\Sigma}(\mathbf{1}) & \xrightarrow{g_{\vec{x}}} & \mathbf{1} & \xrightarrow{\langle id, \zeta \rangle} & \vec{\mathcal{H}}(\mathbf{1}) & \xrightarrow{(\pi_2)_1} & \vec{B}(\mathbf{1}) \\
 & & & \searrow \zeta & & & 
 \end{array} \tag{20}$$

with  $\vec{u} = (u_{\alpha_1}, u_{\alpha_2}, u)$ , see observation 9. By (15), (18), and (19) the triangle in the top right commutes.

Thus the coinductive extension  $\vec{u}$  of the  $\vec{B}$ -coalgebra  $(\alpha_1, \alpha_2, \Sigma_\rho(\alpha_1, \alpha_2))$  is a  $\vec{\Sigma}$ -algebra homomorphism and it is well-known that this makes  $\vec{u}$ 's kernel relation a congruence in the sense of (14) for  $\mathcal{F} := \vec{\Sigma}$ , see [12], Sect. 3.2., where  $A = (S_1, S_2, \Sigma(S_1, S_2)) = \mathcal{F}(A)$ ,  $R = (\ker(u_{\alpha_1}), \ker(u_{\alpha_2}), \ker(u))$ , hence  $\mathcal{F}(R) = (S_1, S_2, \Sigma(\ker(u_{\alpha_1}), \ker(u_{\alpha_2}))$ . Considering the third components only, shows that the family of kernel relations  $(R_1 := \ker(u_{\alpha_1}), R_2 := \ker(u_{\alpha_2}), R := \ker(u))$  is  $id_{\Sigma(S_1, S_2)}$ -compatible as desired.  $\blacktriangleleft$

From Theorem 21 we also obtain

► **Corollary 22** (Sufficient Criterion for Compositionality). *Let  $\mathcal{B}_1, \dots, \mathcal{B}_n, \mathcal{B}$  and  $\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}$  be given as above. Let for all  $i \in \{1, \dots, n\}$  the  $\mathcal{SET}$ -endofunctors  $\mathcal{H}_i$  and  $\mathcal{H}$  be given as in (2), then compositionality holds for the heterogeneous scenario, if the computation of the compound system can be described by a natural transformation*

$$\rho : \Sigma(\mathcal{H}_1 \times \dots \times \mathcal{H}_n) \Rightarrow \mathcal{B}\Sigma : \mathcal{SET}^n \rightarrow \mathcal{SET}. \quad \lrcorner$$

## 6 Related Work

**Practical approaches.** The general idea of transforming different behavioural formalisms to a single semantic domain in order to reason about crosscutting concerns is nothing new [6]. We mention only a few approaches: [17] developed consistency checking for sequence diagrams and statecharts based on CSP, while Petri nets were used for the same scenario in [29]. Nevertheless, all approaches utilize fixed types of transition systems and no common framework, which can capture all possible types of transition structures. In recent years, *co-simulation* of coupled heterogeneous systems has become popular and there is already a plethora of work on that topic [7]. In particular [5] tackles the problem of coordinating different models using a dedicated coordination language. However, the majority of these approaches lack theoretical underpinnings, and, to the best of our knowledge, co-simulated comprehensive behaviour has not been formulated coalgebraically.

**SOS Framework, Distributive Laws and Compositionality.** All important variants of SOS rules are described in [1] and we took most of its coalgebraic abstraction from the original work [27], further elaborated in [14], especially for copointed functors in [18], and probably formulated in the most general way in [12]. All important variations of distributive laws and connected aspects of compositionality are surveyed in Chapter 8 of [14]. Moreover, compositionality in the bialgebraic approach is a facet of the microcosm principle: The

behavior of a composed system involves an outer operator on  $\mathcal{B}\text{-Coalg}$ , the composition of behaviors is an inner operator on the final object of  $\mathcal{B}\text{-Coalg}$ , see [9], where the compositionality property is derived from a formalization of the microcosm principle for Lawvere theories.

*Heterogeneity* appears whenever different behavioral paradigms shall be combined. One of the first examples are hybrid systems, which combine discrete and continuous dynamics [11]. However, reasoning about operational semantics of arbitrary heterogeneously typed transition structures is usually treated by common abstractions of the different systems: E.g. the coordination of a Mealy machine and a probabilistic system can be investigated by reducing both systems to labelled transition systems and formulating interactions with LTS-based SOS rules. A different approach, which is closer to ours, is described in [13], where the combination of two distributive laws based on different behavioral specifications is investigated: So-called *heterogeneous* transition systems simultaneously carry two different coalgebraic structures  $\mathcal{B}$  and  $\mathcal{B}'$  and behavioural descriptions are based on natural transformations of the form  $\Sigma(\mathcal{B} \times \mathcal{B}') \Rightarrow (\mathcal{B} \times \mathcal{B}')\Sigma$ . However, the authors do not pick up the holistic view of our approach and do not investigate compositionality.

*Categorically*, heterogeneity leads to the general theory of *(co-)institutions*. [23] proves three different types of logics for coalgebras to be institutions. Another approach are parametrized endofunctors as comprehensive behavioural specifications, where the overall structure can be studied in terms of cofibrations [16]. [28] investigates co-institutions purely dual to classical institutions [25].

## 7 Future Work

We investigated the synchronisation of  $n$  local components to obtain a compound system. The idea was to introduce  $n + 1$  sorts, which reflects the fact that the resulting compound system is obtained *in one step* from the locals. That excludes step-by-step synchronisation, i.e. the assembly of some components to an intermediate composed system, which in a later step is combined with other components, before the resulting global operational semantics is reached. The challenge in future work is to cope with an unsteady number of sorts for the arising intermediate systems. Similarly, our approach cannot directly be applied to asynchronous communications via intermediate components like message queues, object spaces, etc. It is a goal to derive formal underpinnings also in these cases.

Moreover, it is worth thinking about other types of extensions or refinements of local components and how they cause an impact on the composed system. If, for instance, a local system is conservatively extended [1], then we can ask the question whether the compound system is also conservatively extended. Furthermore, it is an open question, whether extensive refinements of the local systems and their interaction specifications can still be handled with interaction laws.

Finally, if additional system properties are imposed on the local behavioural models by modal logic formulae, the question arises, whether the use of co-forgetful functors in the translation of these formulae to the compound system [15] matches the framework proposed in the present paper. Altogether, the goal is to extend the first iteration of our work and, in future steps, develop more insight into the topic.

## References

- 1 Luca Aceto, Wan J. Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. North-Holland / Elsevier, 2001. doi:10.1016/b978-044482830-9/50021-7.
- 2 Falk Bartels. On generalised coinduction and probabilistic specification formats: distributive laws in coalgebraic modelling. *Academisch Proefschrift, Vrije Universiteit Amsterdam*, 2004.
- 3 Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can’t be traced. In Jeanne Ferrante and Peter Mager, editors, *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*, pages 229–239. ACM Press, 1988. doi:10.1145/73560.73580.
- 4 Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-driven software engineering in practice. *Synthesis lectures on software engineering*, 3(1):1–207, 2017.
- 5 Julien Deantoni. Modeling the behavioral semantics of heterogeneous languages and their coordination. In *2016 Architecture-Centric Virtual Integration (ACVI)*, pages 12–18. IEEE, 2016.
- 6 Gregor Engels, Jochen Malte Küster, Reiko Heckel, and Luuk Groenewegen. A methodology for specifying and analyzing consistency of object-oriented behavioral models. In A Min Tjoa and Volker Gruhn, editors, *Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering 2001, Vienna, Austria, September 10-14, 2001*, pages 186–195. ACM, 2001. doi:10.1145/503209.503235.
- 7 Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: a survey. *ACM Computing Surveys (CSUR)*, 51(3):1–33, 2018.
- 8 Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and computation*, 100(2):202–260, 1992.
- 9 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. The microcosm principle and concurrency in coalgebra. In *International Conference on Foundations of Software Science and Computational Structures*, pages 246–260. Springer, 2008.
- 10 Charles Antony Richard Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.
- 11 Bart Jacobs. Object-oriented hybrid systems of coalgebras plus monoid actions. *Theoretical Computer Science*, 239(1):41–95, 2000.
- 12 Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. doi:10.1017/CB09781316823187.
- 13 Marco Kick, John Power, and Alex Simpson. Coalgebraic semantics for timed processes. *Information and Computation*, 204(4):588–609, 2006.
- 14 Bartek Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comput. Sci.*, 412(38):5043–5069, 2011. doi:10.1016/j.tcs.2011.03.023.
- 15 Harald König and Uwe Wolter. Consistency of heterogeneously typed behavioural models: A coalgebraic approach. In Yamine Aït Ameur and Florin Craciun, editors, *Theoretical Aspects of Software Engineering - 16th International Symposium, TASE 2022, Cluj-Napoca, Romania, July 8-10, 2022, Proceedings*, volume 13299 of *Lecture Notes in Computer Science*, pages 308–325. Springer, 2022. doi:10.1007/978-3-031-10363-6\_21.
- 16 Alexander Kurz and Dirk Pattinson. *Coalgebras and modal logic for parameterised endofunctors*. Centrum voor Wiskunde en Informatica, 2000.
- 17 Jochen Malte Küster. Towards Inconsistency Handling of Object-Oriented Behavioral Models. *Electron. Notes Theor. Comput. Sci.*, 109:57–69, 2004. doi:10.1016/j.entcs.2004.02.056.
- 18 Marina Lenisa, John Power, and Hiroshi Watanabe. Category theory for operational semantics. *Theor. Comput. Sci.*, 327(1-2):135–154, 2004. doi:10.1016/j.tcs.2004.07.024.



- 19 Dorian Leroy, Erwan Bousse, Manuel Wimmer, Tanja Mayerhofer, Benoit Combemale, and Wieland Schwinger. Behavioral interfaces for executable DSLs. *Software and Systems Modeling*, 19(4):1015–1043, 2020.
- 20 Giovanni Liboni and Julien Deantoni. Cosim20: An integrated development environment for accurate and efficient distributed co-simulations. In *Proceedings of the 2020 International Conference on Big Data in Management*, pages 76–83, 2020.
- 21 Robin Milner. *Communicating and Mobile Systems: The  $\pi$ -Calculus*. Cambridge University Press, 1999.
- 22 Mogens Nielsen and Glynn Winskel. Models for concurrency. In *Handbook of Logic in Computer Science, Volume 4*. Oxford Science Publications, 1995.
- 23 Dirk Pattinson. Translating logics for coalgebras. In *International Workshop on Algebraic Development Techniques*, pages 393–408. Springer, 2002.
- 24 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.
- 25 Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012. doi:10.1007/978-3-642-17336-3.
- 26 Ana Sokolova. Probabilistic systems coalgebraically: A survey. *Theoretical Computer Science*, 412(38):5095–5110, 2011.
- 27 Daniele Turi and Gordon Plotkin. Towards a mathematical operational semantics. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 280–291. IEEE, 1997.
- 28 U. Wolter. (Co)Institutions for Coalgebras. Reports in Informatics 415, Dep. of Informatics, University of Bergen, 2016.
- 29 Shuzhen Yao and Sol M Shatz. Consistency checking of UML dynamic models based on Petri net techniques. In *2006 15th International Conference on Computing*, pages 289–297. IEEE, 2006.