Integrating Observations and Computations in the Specification of State-Based, Dynamical Systems

Corina Cîrstea

Corpus Christi College and St. John's College

Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computation

Hilary Term 2000

Abstract

The overall goal of this work is to combine the complementary contributions of algebra and coalgebra to specification, in order to provide a formal framework for the specification of state-based, dynamical systems.

Algebraic specification methods benefit from the availability of *inductive techniques* for defining and reasoning about structures that involve *computation*; coalgebraic specification methods complement algebraic ones both in their objectives and in their means of achieving them, by employing *coinductive techniques* for defining and reasoning about structures that involve *observation*. State-based, dynamical systems comprise a computational aspect, concerned with the construction of (new) system states, and an observational aspect, concerned with the observation of (existing) system states, with the two aspects overlapping on features concerned with the evolution of system states.

Existing formalisms for the specification of such systems typically exploit the overlap between computational and observational features to employ either algebraic or coalgebraic techniques for specification and reasoning. However, such a choice limits the expressiveness of these formalisms w.r.t. either observational or computational features. Furthermore, the accounts given by such approaches to the concepts of indistinguishability by observations and respectively of reachability under computations are somewhat artificial, due to the failure to distinguish between computational and observational features.

The approach taken here is to clearly separate the two categories of features (by shifting the features concerned with the evolution of system states to the computational component), and to use algebra and respectively coalgebra in formalising them. In particular, such an approach yields a coalgebraically-defined notion of indistinguishability by observations, and an algebraically-defined notion of reachability under computations. The relationship between computing new states and observing the resulting states is specified by suitably lifting the coalgebraic structure of the semantic domains induced by the observational component to computations over these semantic domains. Such an approach automatically results in a compatibility between computational and observational features, with the observational indistinguishability of states being preserved by computations, and with the reachability of states under computations being preserved by observations.

Correctness properties of system behaviour are formalised using equational sentences. This is a standard technique in algebraic specification. A similar technique is used here for coalgebraic specification, with the resulting notion of sentence capturing system invariants quantified over state spaces. Moreover, a sound and complete calculus for reasoning about the specified behaviours is formulated in a concrete setting obtained by syntactically dualising the setting of many-sorted algebra. Equational sentences are then used to formalise the equivalence of computations as well as various system invariants, with the associated notions of satisfaction abstracting away observationally indistinguishable and respectively unreachable states, and with the associated proof techniques employing coinduction and respectively induction.

Suitably instantiating the resulting approach yields a formalism for the specification and verification of objects.

Acknowledgements

I would like to express my deepest gratitude towards my supervisors, Dr Grant Malcolm and Dr Luke Ong, for their support and guidance through various stages in the development of this thesis. Ever since my arrival in Oxford, Dr Malcolm has been a constant source of encouragement and technical guidance. Since taking up the rôle of co-supervisor, Dr Ong has also been very supportive of my work, and of great help in organising the final stages of the thesis.

Special thanks go to Professor Joseph Goguen and to the other members of the Declarative Group at Oxford, for stimulating discussions which made my initiation into the field of algebraic specification very enjoyable.

My involvement in the theory of coalgebras is largely due to Dr Bart Jacobs and Dr Jan Rutten, whose contributions to this relatively new field have made the present work possible. I would like to thank them for their interest in my work and for their valuable comments on various topics related to this thesis.

I would also like to thank my colleague, James Worrell, for being a reliable source for consultation and advice, and Dr Andrea Corradini and Professor Ugo Montanari, for brief but illuminating discussions on the subject of this thesis. The final version of the thesis has also benefited from the comments I received from my two examiners, Dr Jan Rutten and Dr Oege de Moor.

The research reported here would not have been possible without the financial support from the University of Oxford, support which I gratefully acknowledge. Most of the work for the thesis was carried out as a graduate student of Corpus Christi College Oxford. The thesis was completed while holding a Junior Research Fellowship at St. John's College Oxford.

Contents

1 Introduction

The use of algebra in specification goes back to [Gog75, GTW78] and the use of *many-sorted algebra* for the specification of data types (with *multiple sorts* being used to denote types, *many-sorted operations* being used to specify type functionality, and *many-sorted equations* being used to constrain this functionality). Since then many authors, including [Rei81, EKMP82, GM82, ONE89, Hen90, BB91, Gog91, BH95, Pad96, GM97] have contributed to the development of the field now known as algebraic specification.

As early as [Rei81, GM82], algebraic approaches to specification identified the need to abstract away certain implementation details in the specification of data types. This resulted in a distinction being made between *visible* and *hidden* sorts, and in *behavioural equivalence relations* on algebras being used to capture observational indistinguishability [Rei81, GM82]. In addition, notions of *initial* and *final realisation*[1] of a given behaviour were used to provide denotations for data type specifications [GM82].

Although the notion of behavioural equivalence was originally associated with final semantics, subsequent developments in the area of behavioural specification have paid increasingly less attention to the relevance of finality to the underlying semantics, being primarily concerned with the development of inductive proof techniques for behavioural satisfaction [NO88, Hen90, BB91]. However, despite the simplicity characterising induction principles, such techniques turned out to be surprisingly complex, as they employed induction over the *contexts* used for observation, with the number of contexts becoming increasingly unmanageable as larger specifications were considered (see e.g. [Hen90]). Consequently, attention has gradually turned to definition/proof techniques based on *coinduction* [MG94, BH95, BH96, GM97], with such techniques exploiting the maximality of behavioural equivalence amongst the congruences on a given algebra. In addition, the need to distinguish between *system constructors* and *system observers*, as well as to impose a certain compatibility between them (by requiring the behavioural equivalence relations induced by the observers to define congruences w.r.t. the constructors) has become increasingly apparent in recent approaches [Dia98, RG00, HB99].

New developments in the theory of coalgebras [Rut96] have suggested an alternative approach to system specification, where the primary interest is shifted from computations to observations. The use of coalgebraic methods in specification [Rei95, Rut96] stems from, and at the same time generalises the use of transition systems as operational models for dynamical systems [RT94, Rut95]. In particular, coalgebras have been used to describe the behaviour of classes in object-oriented programming [Jac96c, Jac96b, Jac96a, Jac97].

Rather than being concerned with the computations resulting in new states, coalgebraic approaches focus on the observable behaviour of system states, with the notion of *bisimulation* being used

[1] The terminology is drawn from automata theory. Initial realisations incorporate reachable behaviours, whereas final realisations incorporate observable behaviours.

to formalise observational indistinguishability. Initial states are occasionally used in coalgebraic specification formalisms to provide (basic) system constructors [Jac96c, Jac96a], this yielding a notion of reachability akin to that of transition systems. (The integration of initial states in these formalisms is facilitated by the fact that they employ a suitably-restricted algebraic syntax.)

Ongoing work in the areas of algebraic and coalgebraic specification concerns the integration of additional computational and observational features into existing formalisms for system specification. Examples include extensions [Dia98, RG00] of the *hidden algebra* formalism [GM97, GM00], as well as work on (Ω, Ξ)-logic [HK99], building on earlier work on observational specification logics [HB99]. However, this integration is hindered by the use of an algebraic syntax (which limits the expressiveness w.r.t. observational features) on the one hand, and by the decision to *simultaneously* specify computational and observational features (which results in compatibility restrictions having to be imposed in order to allow for an increased expressiveness w.r.t. computational features) on the other.

This thesis presents an alternative approach to integrating computational and observational features in the specification of state-based, dynamical systems, approach which uses algebra and respectively coalgebra to formalise such features, and which takes a layered view towards their integration. Further motivation for such an approach, an outline of the main contributions, as well as a more detailed account of the results presented are given in the following.

Motivation State-based, dynamical systems comprise a computational aspect, concerned with the construction of new system states and with the reachability of states under computations, and an observational aspect, concerned with the observation of existing system states and with the indistinguishability of states by observations. These two aspects overlap, in that features concerned with the evolution of system states (i.e. with the *system dynamics* [AM74b]) can be regarded both as a means to compute new states and as a means to observe existing states. There exist, however, system features whose nature is either purely computational or purely observational, with the construction of initial states and respectively the extraction of visible information from system states being instances of such features. Moreover, the notions of *reachable* and respectively *observable* system are dual to each other: while reachability of a system amounts to the mapping taking potential inputs to the states reachable from them being surjective, observability of a system amounts to the mapping taking system states to their observable behaviour being injective [AM74a].

Nevertheless, existing (co)algebraic approaches to the specification of state-based, dynamical systems tend to be primarily concerned *either* with the computations yielding (new) system states [GM97, HB99], *or* with the observations that can be made about (existing) system states [Rei95, Rut96, Jac96c, Jac96a, Cor98]. Depending on whether the emphasis is on computations or on observations, the underlying formalisms typically employ algebraic techniques with a semantics based on initial models, or coalgebraic techniques with a semantics based on final models. Some of these formalisms also accommodate purely observational and respectively purely computational features: observers yielding visible results together with observational equivalence relations are typically used in algebraic approaches [GM97, HB99], whereas initial states are occasionally used in coalgebraic approaches [Jac96c, Jac96a, Jac97]. In addition, some of these formalisms identify the need for a certain compatibility between the two categories of features [Dia98, RG00, HK99]. However, a

recurring problem in this work is the failure to *fully* and *naturally* capture either the observational or the computational aspect of systems. On the one hand, the choice to employ the same (usually algebraic) syntax in capturing both computational and observational features limits the expressiveness of the resulting formalisms w.r.t. at least one category of features (usually the observational ones). On the other hand, the decision to simultaneously specify computational and observational features in such approaches results in further constraints having to be imposed in order to guarantee that these features are compatible with each other. Such constraints involve either restrictions on the algebras used to model the specified systems [Dia98, HB99], or restrictions on the specifications used to describe system behaviour [RG00] (both aimed to ensure that observational equivalence relations are preserved by the system constructors). Furthermore, the resulting notions of indistinguishability by observations are often unnecessarily complex, as no distinction, either syntactic or semantic, is made between computational and observational features (and consequently observational equivalence relations take into account more features than it is actually needed).

The aim of this work is to fully exploit the expressive power of algebra and coalgebra when specifying purely computational and respectively purely observational structures, and to combine their complementary contributions when specifying structures that have both computational and observational features, in a manner which guarantees a certain compatibility between the two categories of features. This is achieved by clearly distinguishing between computational and observational features and using algebra and respectively coalgebra for their formalisation, and by taking a layered approach to their integration.

System features concerned with the evolution of systems are here regarded as part of the computational component only. This decision is justified partly by the ideas underlying the object-oriented paradigm, whereby a system's reaction to a request is fully determined by its current state, and partly by the layered nature of the approach, which requires all the computational features (including the features concerned with the evolution of systems) to be fully defined in terms of their effect on observational features (and hence also on the features concerned with the system structure)[2]. The resulting notion of reachability therefore takes into account both the construction and the evolution of system states, whereas the resulting notion of observational indistinguishability depends only on the structure of system states. (A different classification into observational and computational features, which regarded the evolution of system states as part of the observational component only, would have yielded an insufficiently rich notion of reachability, as well as an unnecessarily complex notion of observational indistinguishability.)

A consequence of the initial separation between observational and computational features is that the resulting framework employs an algebraically-defined notion of reachability under computations and a coalgebraically-defined notion of indistinguishability by observations (which are more natural and at the same time less complex than those employed by unilateral approaches). Thus, this framework benefits from the availability of inductive and coinductive techniques for proving properties up to reachability and respectively up to observability. Also, taking a layered approach to the integration of computational and observational features automatically ensures that observational indistinguishability is preserved by computations, whereas reachability is preserved by observations.

[2] As a result, observations also involving future states can be expressed in terms of observations only involving structural properties of states.

Equational sentences are used here to formalise correctness properties of system behaviour. The decision to focus on equational approaches is motivated by the intention to specify in a concise manner system properties quantified over state spaces, as well as by the existence of several equational specification formalisms for state-based, dynamical systems.

The use of equations in capturing the equivalence of computations is a standard technique in algebraic specification. This technique is supported by the existence of a sound and complete calculus of equational deduction, as well as of a characterisability result regarding the expressiveness of equational sentences w.r.t. suitably-closed classes of models (see e.g. [MT92]). Less is currently known about the suitability of equational approaches to coalgebraic specification. Moreover, all the existing equational coalgebraic specification formalisms [Rei95, HR95, Jac96c, Jac96a, Cor98] use suitably-restricted algebraic equations (as opposed to coalgebraically-defined equations) to constrain the specified behaviours. This not only restricts the expressiveness of such formalisms w.r.t. the structures specifiable within them (which are essentially algebraic structures), but at the same time prevents the use of standard algebraic techniques in correctness proofs (as the equations used for specification are required to contain precisely one variable denoting a system state).

An alternative, coalgebraic notion of equation is introduced here in an abstract setting. On the one hand, this notion generalises the kinds of algebraic equations typically used in system specification, by capturing system invariants that refer to the equality or similarity of observations on the system states. On the other hand, the definition of this notion also applies to situations where observations may have structured result types. In particular, an instance of this notion is used here to specify and reason about observational structures allowing for a choice in the result type of observers.

Contributions A first contribution of this thesis stands in investigating the existence of a coalgebraic, equational specification formalism whose expressiveness equals that of many-sorted algebra both w.r.t. the structures specifiable within it, and from the point of view of characterising classes of models by equational sentences. This investigation is carried out in two steps. First, a syntactic dual of many-sorted algebra, involving notions of *cosignature*, *covariable* and *coterm* is considered, and notions of *coequation* and *coequational satisfaction* are introduced in this setting. A sound and complete deduction calculus for coequations is also formulated. The resulting formalism does not, however, equal the expressiveness of many-sorted algebra w.r.t. the structures specifiable within it. This is also reflected in the fact that the notion of observational indistinguishability employed by this formalism is not sufficiently expressive. Consequently, a second step considers an enriched version of this formalism, which assumes the availability of a fixed data universe for specification and reasoning. The new formalism benefits from the existence of a sound and complete deduction calculus (obtained by suitably extending the deduction calculus previously formulated), as well as of an expressiveness similar to that of many-sorted algebra w.r.t. the structures specifiable within it. Coequations are still not sufficiently expressive to yield a characterisability result similar to that of many-sorted algebra. Nevertheless, they succeed in capturing, in a concise manner, system properties quantified over state spaces.

A second contribution of the thesis stands in developing an abstract coalgebraic framework for the specification of observational structures, framework which unifies some of the existing equational specification formalisms for state-based, dynamical systems, and of which the previously-described

formalism is also an instance. A framework for the specification of computational structures, of which the many-sorted algebraic approach to the specification of data types is an instance is also derived, essentially by dualising the coalgebraic framework.

Another contribution of the thesis stands in integrating the algebraic and coalgebraic frameworks thus obtained into a framework for the specification of systems having both an observational and a computational component. This integration is based on the work in [Tur96] on well-behaved operational semantics, and uses liftings of the coalgebraic structure of state spaces to computations over these state spaces to interpret computations on the state spaces induced by the observational component. The resulting interpretations are well-behaved, in that observational indistinguishability is preserved by computations, whereas reachability is preserved by observations. Abstract equations and coequations are then used to formalise the equivalence of computations and respectively invariants on the structure of systems. To account for an interest in observable behaviours and ground computations only, the associated notions of satisfaction abstract away observationally indistinguishable and respectively unreachable behaviours. The use of such abstractions also results in the availability of coinductive and respectively inductive techniques for correctness proofs. Specifically, proving the satisfaction of an equation up to observability can be reduced to exhibiting a bisimulation which relates the lhs and rhs of that equation, whereas proving the satisfaction of a coequation up to reachability can be reduced to exhibiting a submodel which satisfies that coequation.

A final contribution of the thesis stands in developing a specification formalism for objects, as an instance of the previously-described abstract framework. This formalism uses variants of many-sorted algebra and of its syntactic dual to specify object functionality and respectively object structure, and a notion of *constraint* to specify the changes in structure associated to a particular choice of functionality. In addition, many-sorted equations are used to capture the equivalence of object-oriented programs, while their syntactic duals are used to capture invariants on the object structure. Correctness proofs for such properties employ coinductive and respectively inductive techniques. Specifically, proving that two programs are observationally equivalent amounts to exhibiting a bisimulation which relates the states yielded by those programs, whereas proving that a state invariant holds in reachable states amounts to exhibiting a subspace of the state space which contains the reachable states and whose states satisfy the given invariant. The clear separation between structure and functionality, together with the use of algebra and respectively coalgebra in capturing them increase the expressiveness of the resulting formalism compared to unilateral, either algebraic or coalgebraic, specification formalisms for objects. In particular, a more comprehensive treatment of inheritance, which accounts both for its classification aspect and for its reusability aspect is provided by such an approach.

Synopsis Some familiarity with standard algebraic and coalgebraic specification techniques, as well as with the language of categories is required for reading this thesis. Chapter 2 provides an outline of all the relevant concepts, and at the same time establishes some notational conventions to be used in forthcoming chapters.

Chapter 3 introduces an abstract framework for the specification of structures involving both observational and computational features, based on a clear separation between the two categories of features.

In Section 3.1, attention is restricted to observational structures. Abstract notions of *cosignature*, *observer* and *coequation* are used to specify particular kinds of observational structures and to further constrain these structures. Coalgebras are used as models for the resulting specifications, with finality playing an important rôle in assigning suitable denotations to these specifications. Coequations are shown to induce subcoalgebras of given coalgebras on the one hand and *covarieties* on the other, while the resulting specification logic is shown to be an *institution*. Moreover, similar results are shown to hold for a notion of satisfaction of coequations up to observability. Other results in this section refer to the existence of suitable denotations for abstract cosignature morphisms, as well as of canonical ways of combining abstract coalgebraic specifications and respectively implementations. The expressiveness of coequations from the point of view of characterising covarieties is also briefly discussed.

In Section 3.2, an algebraic framework for the specification of structures involving computation is obtained essentially by dualising the previously-obtained coalgebraic framework.

In Section 3.3, the two frameworks are integrated in order to account for systems having both an observational and a computational component. Following [Tur96], liftings of the coalgebraic structure of semantic domains to computations over these semantic domains are used to interpret computations on the semantic domains induced by the observational component. (A dual approach, involving liftings of the algebraic structure of syntactic domains to observable behaviours over these syntactic domains is also described.) Abstract equations and coequations are used to formalise correctness properties of the behaviours specified using such liftings, with the associated notions of satisfaction being up to observability and respectively up to reachability. Both notions of satisfaction are shown to yield institutions. Suitable denotations for the resulting specifications are shown to be provided by the quotients of the unique homomorphisms from the initial to the final models. Two compositionality results, allowing specifications and respectively implementations to be combined in a canonical way are also formulated.

Chapter 4 develops a coalgebraic, equational formalism for the specification of structures allowing for a choice in the result type of observers. This formalism is, to a large extent, a syntactic dual of many-sorted algebra – notions of *covariable*, *coterm* and *coequation*, dual to those of *variable*, *term* and *equation* are used for specification, while coalgebras are used as models for the resulting specifications. A deduction calculus for coequations is formulated, and its soundness and completeness w.r.t. their satisfaction by coalgebras is proved. The approach is then extended to account for the availability of a fixed data universe w.r.t. which observational structures are specified. Such an extension is necessary in order to capture, via cosignatures, arbitrary polynomial endofunctors. By enriching the deduction calculus previously formulated with a rule which accounts for the data universe being fixed, the soundness and completeness results also generalise to the new setting. The resulting formalism is shown to be at least as expressive as many-sorted algebra w.r.t. the structures specifiable within it (which correspond to coalgebras of *extended* polynomial endofunctors), but less expressive as far as characterising classes of models by equational sentences is concerned.

Chapter 5 describes a formalism for the specification and verification of objects, obtained by suitably instantiating the specification framework introduced in Chapter 3. Specifically, many-sorted signatures and cosignatures are used to specify the data manipulated by objects, while *signatures*

of constructors and *cosignatures of observers* (defined as many-sorted signatures and respectively cosignatures whose sorts and operation symbols have been classified into visible and hidden ones, depending on whether they refer to data types or to object types) are used to specify object types. The liftings required by the abstract framework are then defined using suitably-restricted sets of *constraints*. Many-sorted equations and coequations are used to formalise correctness properties of the specified behaviours, with correctness proofs benefiting from the existence of a sound deduction calculus for constraints. An object specification of (bounded) stacks is used to illustrate the approach, as well as to compare it with existing approaches. A generic example involving the specification of inheritance relationships between classes is also given.

Chapter 6 summarises the results presented and briefly outlines possible directions for future research.

2 Preliminaries

This chapter contains some prerequisites for the material in the forthcoming chapters, including basic notions of category theory (Section 2.1) and general logics (Section 2.2), and an outline of the algebraic (Section 2.3) and coalgebraic (Section 2.4) approaches to specification.

2.1 Category Theory

This section only recalls the category-theoretic concepts and results that are directly relevant to the present work. For a comprehensive introduction to category theory, the reader is referred to [Bor94a, Bor94b].

The theory of categories provides an abstract setting for the study of universal properties. The categorical formulation of such properties involves notions of *object* (used to denote an abstract entity) and *arrow* (used to denote a particular relation between two such entities).

Definition 2.1.1. *A* **category** C *consists of:*

1. *a class*[1] $|\mathsf{C}|$ *of* **objects**
2. *for every pair* A, B *of objects, a set* $\mathsf{C}(A, B)$ *of* **arrows** *from* A *to* B
3. *for every triple* A, B, C *of objects, a* **composition law**

$$\circ : \mathsf{C}(B, C) \times \mathsf{C}(A, B) \to \mathsf{C}(A, C)$$

4. *for every object* A, *an* **identity** *arrow* $1_A \in \mathsf{C}(A, A)$

subject to the following constraints:

1. $h \circ (g \circ f) = (h \circ g) \circ f$ *for any* $f \in \mathsf{C}(A, B)$, $g \in \mathsf{C}(B, C)$ *and* $h \in \mathsf{C}(C, D)$
2. $f \circ 1_A = f = 1_B \circ f$ *for any* $f \in \mathsf{C}(A, B)$.

An arrow $f \in \mathsf{C}(A, B)$ is said to have *domain* A and *codomain* B. One writes $f : A \to B$ or $A \xrightarrow{f} B$ for such an arrow.

The class of arrows of a category C is denoted $\|\mathsf{C}\|$.

Definition 2.1.2. *A category* C *is* **small** *if and only if* $|\mathsf{C}|$ *is a set; otherwise it is a* **large** *category. A category* C *is* **finite** *if and only if* $|\mathsf{C}|$ *and* $\|\mathsf{C}\|$ *are finite sets.*

[1] See [Bor94a, pp. 3-4] or [AHS90, Chapter 2] for a discussion on the distinction between sets and classes.

Definition 2.1.3. *A* **subcategory** B *of a category* A *consists of:*

1. *a subclass* $|\mathsf{B}|$ *of* $|\mathsf{A}|$
2. *for every* $A, B \in |\mathsf{B}|$*, a subset* $\mathsf{B}(A,B)$ *of* $\mathsf{A}(A,B)$

such that:

1. $f \in \mathsf{B}(A,B)$ *and* $g \in \mathsf{B}(B,C)$ *imply* $g \circ f \in \mathsf{B}(A,C)$
2. $1_A \in \mathsf{B}(A,A)$ *for any* $A \in |\mathsf{B}|$.

A subcategory B *of a category* A *is* **full** *if and only if* $\mathsf{B}(A,B) = \mathsf{A}(A,B)$ *for any* $A, B \in |\mathsf{B}|$.

Example 2.1.4. The canonical example of a category is Set, the category of sets and functions. This category is not small.

A slightly more structured category is Set^S, the category of S-indexed sets and S-indexed functions, with S a set. Its objects are *S-indexed sets* (i.e. families $(A_s)_{s\in S}$ with $A_s \in |\mathsf{Set}|$ for $s \in S$), its arrows are *S-indexed functions* (i.e. families $(f_s)_{s\in S}$ with $f_s \in \mathsf{Set}(A_s, B_s)$ for $s \in S$), while composition is defined component-wise. An *S-indexed subset* of an S-indexed set A is an S-indexed set B with $B_s \subseteq A_s$ for $s \in S$. Given S-indexed sets A and B, an *S-indexed relation* between A and B is an S-indexed set R with $R_s \subseteq A_s \times B_s$ for $s \in S$. An S-indexed relation R is an *equivalence relation* if and only if each R_s, with $s \in S$, is an equivalence relation.

Finally, given $V \subseteq S$ together with a V-indexed set D, the category Set^S_D has S-indexed sets A such that $A_v = D_v$ for $v \in V$ as objects, and S-indexed functions f such that $f_v = 1_{D_v}$ for $v \in V$ as arrows.

Example 2.1.5. Given a category C, its *dual* (or *opposite*) *category* C^{op} has $|\mathsf{C}^{\mathsf{op}}| = |\mathsf{C}|$ and $\mathsf{C}^{\mathsf{op}}(A,B) = \mathsf{C}(B,A)$ for $A, B \in |\mathsf{C}|$.

Example 2.1.6. Given a category C and a C-object C, the *slice category* C/C has pairs $\langle D, d\rangle$ with D a C-object and $d : D \to C$ a C-arrow as objects, and C-arrows $f : D \to D'$ satisfying $d' \circ f = d$ as arrows from $\langle D, d\rangle$ to $\langle D', d'\rangle$. Similarly, the category whose objects are given by pairs $\langle D, d\rangle$ with D a C-object and $d : C \to D$ a C-arrow, and whose arrows from $\langle D, d\rangle$ to $\langle D', d'\rangle$ are given by C-arrows $f : D \to D'$ satisfying $d' = f \circ d$ is denoted $\mathsf{C}\backslash C$.

Definition 2.1.7. *In a category* C*, an arrow* $f : A \to C$ *is said to* **factor** *through an arrow* $g : B \to C$ *(respectively* $g : A \to B$*) if and only if there exists an arrow* $h : A \to B$ *(*$h : B \to C$*) such that* $f = g \circ h$ *(*$f = h \circ g$*).*

Definition 2.1.8. *An arrow* $f : A \to B$ *in a category* C *is a* **monomorphism** *(respectively* **epimorphism***) if and only if for any pair of arrows* $g, h : C \to A$ *(*$g, h : B \to C$*) with* $C \in |\mathsf{C}|$*,* $f \circ g = f \circ h$ *(*$g \circ f = h \circ f$*) implies* $g = h$*. The arrow* $f : A \to B$ *is an* **isomorphism** *if and only if there exists an arrow* $g : B \to A$ *such that* $f \circ g = 1_B$ *and* $g \circ f = 1_A$*. If* $f : A \to B$ *and* $g : B \to A$ *are such that* $g \circ f = 1_A$*, then* f *is called a* **right inverse** *(or* **section***) for* g*,* g *is called a* **left inverse** *(or* **retraction***) for* f*, and* A *is called a* **retract** *of* B*.*

Monomorphisms are denoted $f : A \rightarrowtail B$, while epimorphisms are denoted $f : A \twoheadrightarrow B$. For a category C, the class of C-monomorphisms is denoted Mono(C), while the class of C-epimorphisms is denoted Epi(C).

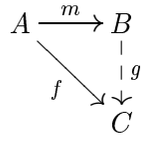
Proposition 2.1.9. *If $g \circ f$ is a monomorphism (respectively epimorphism), then f (g) is a monomorphism (epimorphism).*

Example 2.1.10. In Set (respectively Set^S or Set^S_D), monomorphisms coincide with (families of) injective functions, while epimorphisms coincide with (families of) surjective functions. Set-, Set^S- or Set^S_D-arrows which are both monomorphisms and epimorphisms are isomorphisms. In Set, Set^S and Set^S_D, all monomorphisms with non-empty domains have a retract, while all epimorphisms have a section.

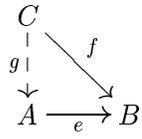
Definition 2.1.11. *Let* C *denote an arbitrary category, and let A, B denote* C*-objects. A* **span** *on A, B is a pair $\langle f, g \rangle$ of* C*-arrows, with $f : C \to A$ and $g : C \to B$ for some* C*-object C. Also, a* **relation** *on A, B is a tuple $\langle R, r_1, r_2 \rangle$ with R a* C*-object and $r_1 : R \to A$, $r_2 : R \to B$ some* C*-arrows, such that whenever $f, g : D \to R$ are* C*-arrows satisfying $r_i \circ f = r_i \circ g$ for $i = 1, 2$, it follows that $f = g$. (In this case, $\langle r_1, r_2 \rangle$ is called a* **monomorphic span**.*) For $C \in |\mathsf{C}|$, the relation given by $\langle C, 1_C, 1_C \rangle$ is called the* **equality relation** *on C. Given relations $\langle R, r_1, r_2 \rangle$ and $\langle R', r'_1, r'_2 \rangle$ on A, B, a* **morphism** *from $\langle R, r_1, r_2 \rangle$ to $\langle R', r'_1, r'_2 \rangle$ is a* C*-arrow $r : R \to R'$ such that $r_i = r'_i \circ r$ for $i = 1, 2$.*

Remark 2.1.12. For $A, B \in |\mathsf{C}|$, the category $\mathsf{Rel}(A, B)$ of relations on A, B is a preorder; that is, there exists at most one morphism between any two relations on A, B. One writes $\langle R, r_1, r_2 \rangle \leq \langle R', r'_1, r'_2 \rangle$ if there exists a morphism between $\langle R, r_1, r_2 \rangle$ and $\langle R', r'_1, r'_2 \rangle$.

Definition 2.1.13. *Let* C *denote an arbitrary category, and let $\mathcal{E}$ and $\mathcal{M}$ denote classes of* C*-arrows. A* C*-object C is $\mathcal{M}$-***injective** *if and only if for any $\mathcal{M}$-arrow $m : A \to B$ and* C*-arrow $f : A \to C$, there exists a* C*-arrow $g : B \to C$ satisfying $g \circ m = f$.*



*Also, C is $\mathcal{E}$-***projective** *if and only if for any $\mathcal{E}$-arrow $e : A \to B$ and* C*-arrow $f : C \to B$, there exists a* C*-arrow $g : C \to A$ satisfying $e \circ g = f$.*



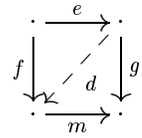
The category C **has enough** $\mathcal{M}$**-injectives** *(respectively* **enough** $\mathcal{E}$**-projectives**) *if and only if every* C*-object is the domain of an $\mathcal{M}$-arrow into an $\mathcal{M}$-injective object (the codomain of an $\mathcal{E}$-arrow from an $\mathcal{E}$-projective object).*

If $\mathcal{E}$ and $\mathcal{M}$ are taken to be Epi(C) and respectively Mono(C), then the prefixes $\mathcal{E}$- and $\mathcal{M}$- can be omitted.

Example 2.1.14. In Set, any object apart from the empty set is injective, while any object is projective.

Definition 2.1.15. *Let* C, $\mathcal{E}$ *and* $\mathcal{M}$ *be as in Definition 2.1.13.* $(\mathcal{E}, \mathcal{M})$ *is a* **factorisation system** *for* C *if and only if the following hold:*

- $\mathcal{E}$ *and* $\mathcal{M}$ *are closed under isomorphisms*
- C **has** $(\mathcal{E}, \mathcal{M})$**-factorisations**, *i.e. every* C*-arrow* f *has a factorisation of form* $f = m \circ e$ *with* $e \in \mathcal{E}$ *and* $m \in \mathcal{M}$
- C **has the unique diagonalisation property**, *i.e. whenever the* C *arrows* $e \in \mathcal{E}$, $m \in \mathcal{M}$, f *and* g *satisfy* $m \circ f = g \circ e$, *there exists a unique* C*-arrow* d *satisfying* $d \circ e = f$ *and* $m \circ d = g$.



In this case, C *is called an* $(\mathcal{E}, \mathcal{M})$**-category**.

Example 2.1.16. (Epi(Set), Mono(Set)) is a factorisation system for Set.

Arrows between categories are called *functors* and are required to preserve the categorical structure.

Definition 2.1.17. *Let* C *and* D *denote categories. A* **functor** F *from* C *to* D *consists of:*

1. *a mapping* $\mathsf{F} : |\mathsf{C}| \to |\mathsf{D}|$
2. *for every pair* A, B *of* C*-objects, a function* $\mathsf{F} : \mathsf{C}(A, B) \to \mathsf{D}(\mathsf{F}(A), \mathsf{F}(B))$

subject to the following constraints:

1. $\mathsf{F}(g \circ f) = \mathsf{F}(g) \circ \mathsf{F}(f)$ *for any* $f \in \mathsf{C}(A, B)$ *and* $g \in \mathsf{C}(B, C)$
2. $\mathsf{F}(1_C) = 1_{\mathsf{F}(C)}$ *for any* $C \in |\mathsf{C}|$.

A functor F *is* **faithful** *(respectively* **full***) if and only if the function* $\mathsf{F} : \mathsf{C}(A, B) \to \mathsf{D}(\mathsf{F}(A), \mathsf{F}(B))$ *is injective (surjective) for any* $A, B \in |\mathsf{C}|$. *An* **endofunctor** *on a category* C *is a functor* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$.

One writes $\mathsf{F} : \mathsf{C} \to \mathsf{D}$ for a functor F from C to D. Also, given $\mathsf{F} : \mathsf{C} \to \mathsf{D}$ together with $A \in |\mathsf{C}|$ and $f \in \|\mathsf{C}\|$, $\mathsf{F}(A)$ and $\mathsf{F}(f)$ are alternatively denoted $\mathsf{F}A$ and respectively $\mathsf{F}f$.

For a category C, the *identity functor* $\mathsf{Id}_\mathsf{C} : \mathsf{C} \to \mathsf{C}$ takes C-objects and arrows to themselves.

Functor composition is defined by: $\mathsf{G} \circ \mathsf{F} : \mathsf{C} \to \mathsf{E}$, $(\mathsf{G} \circ \mathsf{F})(A) = \mathsf{G}(\mathsf{F}A)$ for $A \in |\mathsf{C}|$, and $(\mathsf{G} \circ \mathsf{F})(f) = \mathsf{G}(\mathsf{F}f)$ for $f \in \|\mathsf{C}\|$, where $\mathsf{F} : \mathsf{C} \to \mathsf{D}$ and $\mathsf{G} : \mathsf{D} \to \mathsf{E}$. $\mathsf{G} \circ \mathsf{F}$ is alternatively denoted GF.

Small categories and functors between them constitute a category, denoted Cat.

The functors between two arbitrary categories do not, in general, constitute a set. Consequently, arbitrary categories and functors between them do not constitute a category. The structures obtained by dropping the requirements that the objects constitute a class and respectively that the arrows between two objects constitute a set in the definition of categories are called *quasi-categories* (see e.g. [AHS90]). (In the case of quasi-categories, the objects and respectively the arrows between two objects form so-called *conglomerates*, which formalise collections of classes.)

Example 2.1.18. Classes and functions between them constitute a quasi-category, denoted SET.

Example 2.1.19. Categories and functors between them constitute a quasi-category, denoted CAT.

Any category is also a quasi-category. The notion of functor generalises to quasi-categories, with the generalised notion still being called a functor.

Definition 2.1.20. *Two categories* C *and* D *are* **isomorphic** *if and only if there exist functors* $\mathsf{F} : \mathsf{C} \to \mathsf{D}$ *and* $\mathsf{G} : \mathsf{D} \to \mathsf{C}$ *such that* $\mathsf{GF} = \mathsf{Id}_{\mathsf{C}}$ *and* $\mathsf{FG} = \mathsf{Id}_{\mathsf{D}}$.

Definition 2.1.21. *A functor* $\mathsf{F} : \mathsf{C} \to \mathsf{C}'$ **preserves** *a property* P *of arrows if whenever a* C*-arrow* f *has property* P*, so does* $\mathsf{F}f$. F **reflects** *the property* P *if whenever* $\mathsf{F}f$ *has property* P*, so does* f. F **creates** *the property* P *if, for any* C′*-arrow* f' *with property* P*, there exists precisely one* C*-arrow* f *such that* $\mathsf{F}f = f'$*, and moreover,* f *has property* P*. Finally,* F **lifts** *the property* P *if, for any* C′*-arrow* f' *with property* P*, there exists a* C*-arrow* f *with property* P *such that* $\mathsf{F}f = f'$.

It follows immediately that if a functor creates a certain property, then it also lifts that property.

Arrows between functors are captured by *natural transformations*.

Definition 2.1.22. *Let* $\mathsf{F}, \mathsf{G} : \mathsf{C} \to \mathsf{D}$ *denote arbitrary functors. A* **natural transformation** α *from* F *to* G *(denoted* $\alpha : \mathsf{F} \Rightarrow \mathsf{G}$*, or* $\mathsf{F} \overset{\alpha}{\Longrightarrow} \mathsf{G}$*) is given by a class of* D*-arrows* $\alpha_A : \mathsf{F}A \to \mathsf{G}A$ *for* $A \in |\mathsf{C}|$*, such that* $\alpha_B \circ \mathsf{F}f = \mathsf{G}f \circ \alpha_A$ *for any* C*-arrow* $f : A \to B$.

$$\begin{array}{ccc} \mathsf{F}A & \xrightarrow{\alpha_A} & \mathsf{G}A \\ {\scriptstyle \mathsf{F}f}\downarrow & & \downarrow{\scriptstyle \mathsf{G}f} \\ \mathsf{F}B & \xrightarrow[\alpha_B]{} & \mathsf{G}B \end{array}$$

A natural transformation $\alpha : \mathsf{F} \Rightarrow \mathsf{G}$ *is a* **natural monomorphism** *(respectively a* **natural epimorphism** *or a* **natural isomorphism***) if and only if* α_A *is a monomorphism (respectively an epimorphism or an isomorphism) for any* $A \in |\mathsf{C}|$.

Given functors $\mathsf{F}, \mathsf{G}, \mathsf{H} : \mathsf{C} \to \mathsf{D}$ together with natural transformations $\alpha : \mathsf{F} \Rightarrow \mathsf{G}$ and $\beta : \mathsf{G} \Rightarrow \mathsf{H}$, letting $(\beta \circ \alpha)_A = \beta_A \circ \alpha_A$ for $A \in |\mathsf{C}|$ yields a natural transformation $\beta \circ \alpha : \mathsf{F} \Rightarrow \mathsf{H}$. Moreover, $\circ$ is associative and has the natural transformations $1_{\mathsf{F}} : \mathsf{F} \Rightarrow \mathsf{F}$ given by $(1_{\mathsf{F}})_A = 1_{\mathsf{F}A}$ for $A \in |\mathsf{C}|$, with $\mathsf{F} : \mathsf{C} \to \mathsf{D}$, as identities.

Example 2.1.23. If C denotes a small category and D denotes an arbitrary category, the functors from C to D and the natural transformations between them constitute a category [C, D], called the *category of functors* from C to D. Moreover, if D is small then so is [C, D].

Remark 2.1.24. If C and D denote arbitrary categories, the functors from C to D and the natural transformations between them only constitute a quasi-category. (In this case, the natural transformations between two functors $\mathsf{F}, \mathsf{G} : \mathsf{C} \to \mathsf{D}$ do not, in general, constitute a set.) This quasi-category is also denoted [C, D].

Amongst the objects of a category, of particular interest are *initial* and *final* ones.

Definition 2.1.25. *Let* C *denote an arbitrary category. A* C*-object* C *is* **initial** *(respectively* **final***) if and only if for any* C*-object* D*, there exists a unique* C*-arrow* $c : C \to D$ *(*$c : D \to C$*).*

The only arrow from an initial or final object to itself is therefore the identity.

Initial and final objects are instances of more general categorical concepts, called *colimits* and respectively *limits*.

Definition 2.1.26. *Let* D *denote an arbitrary category. A* **diagram of shape** D *in a category* C *is a functor* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$.

Example 2.1.27. Diagrams of shape $\cdot \longrightarrow \cdot \longrightarrow \cdots$ (respectively $\cdot \longleftarrow \cdot \longleftarrow \cdots$) are called *ω-chains* (*ω^{op}-chains*).

Definition 2.1.28. *Let* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$ *denote a diagram of shape* D*, with* D *small. A* **cone** *on* d *is a tuple* $(L, (l_D)_{D \in |\mathsf{D}|})$*, with* $L \in |\mathsf{C}|$ *and* $(l_D : L \to \mathsf{d}D) \in \| \mathsf{C} \|$ *for* $D \in |\mathsf{D}|$*, such that* $\mathsf{d}d \circ l_D = l_{D'}$ *for each* $(d : D \to D') \in \| \mathsf{D} \|$*. A* **limit** *for* d *in* C *is a cone* $(L, (l_D)_{D \in |\mathsf{D}|})$ *on* d*, having the property that for any cone* $(C, (c_D)_{D \in |\mathsf{D}|})$ *on* d*, there exists a unique* C*-arrow* $c : C \to L$ *such that* $l_D \circ c = c_D$ *for each* $D \in |\mathsf{D}|$.

Final objects are obtained as limits of diagrams of form $\mathsf{d} : \Phi \to \mathsf{C}$, with Φ a category with no objects. Other kinds of limits include: *(finite) products* (obtained by taking D to be a (finite) category with no arrows other than the identities), *pullbacks* (obtained by taking D to be of shape $\cdot \longrightarrow \cdot \longleftarrow \cdot$), and *equalisers* (obtained by taking D to be of shape $\cdot \rightrightarrows \cdot$). The pullback of a pair of equal arrows is called a *kernel pair*.

Notions of *cocone* and *colimit* can be defined by reversing the direction of arrows in the definitions of cones and respectively limits. Instantiating the general definition of colimit yields notions of *initial object*, *(finite) coproduct* (obtained by taking D to be a (finite) category with no arrows other than the identities), *pushout* (obtained by taking D to be of shape $\cdot \longleftarrow \cdot \longrightarrow \cdot$), and *coequaliser* (obtained by taking D to be of shape $\cdot \rightrightarrows \cdot$). The pushout of a pair of equal arrows is called a *cokernel pair*.

Finally, a coequaliser of a kernel pair of a C-arrow f is called a *quotient of* f, while an equaliser of a cokernel pair of f is called an *image of* f.

Given a family $(C_i)_{i\in I}$ of C-objects, their product (respectively coproduct) is denoted $(\prod_{i\in I} C_i, (\pi_j : \prod_{i\in I} C_i \to C_j)_{j\in I})$ ($(\coprod_{i\in I} C_i, (\iota_j : C_j \to \coprod_{i\in I} C_i)_{j\in I})$). The product arrows π_j (coproduct arrows ι_j) are called *product projections* (*coproduct injections*). And given a family $(f_i : C_i \to D_i)_{i\in I}$ of C-arrows, the unique C-arrow induced by the cone $(\prod_{i\in I} C_i, (f_i \circ \pi_i)_{i\in I})$ on $(D_i)_{i\in I}$ (respectively by the cocone $(\coprod_{i\in I} D_i, (\iota_i \circ f_i)_{i\in I})$ on $(C_i)_{i\in I}$) is denoted $\prod_{i\in I} f_i$ ($\coprod_{i\in I} f_i$). If $I = \{1, \ldots, n\}$, $\prod_{i\in I} C_i$ (respectively $\coprod_{i\in I} C_i$) and $\prod_{i\in I} f_i$ (respectively $\coprod_{i\in I} f_i$) are alternatively denoted $C_1 \times \ldots \times C_n$ ($C_1 + \ldots + C_n$) and $f_1 \times \ldots \times f_n$ ($f_1 + \ldots + f_n$), while the unique C-arrow induced by a cone $(C, (c_i : C \to C_i)_{i\in\{1,\ldots,n\}})$ on $(C_i)_{i\in\{1,\ldots,n\}}$ (respectively by a cocone $(D, (d_i : D_i \to D)_{i\in\{1,\ldots,n\}})$ on $(D_i)_{i\in\{1,\ldots,n\}}$) is denoted $\langle c_1, \ldots, c_n\rangle$ ($[d_1, \ldots, d_n]$).

Definition 2.1.29. *Let* D *denote a small category. A category* C **has** D-**limits**, *or* **is** D-**complete** *if and only if any diagram of shape* D *in* C *has a limit in* C. *A category* C **has (finite) limits**, *or* **is (finitely) complete** *if and only if any diagram of shape* D *in* C, *with* D *a small (respectively finite) category has a limit in* C.

Standard results state that finite completeness and cocompleteness of categories follow from the existence of certain kinds of limits and respectively colimits.

Proposition 2.1.30. *For a category* C, *the following are equivalent:*

1. C *is finitely complete*
2. C *has a final object and pullbacks*
3. C *has binary products and equalisers.*

Example 2.1.31. The categories Set, Set^S and Set^S_D are complete and cocomplete, with limits and colimits in Set^S and Set^S_D being computed component-wise. In Set, an initial object is given by the empty set, while a final object is given by any one-element set; also, products are given by cartesian products, while coproducts are given by disjoint unions.

Example 2.1.32. The category Cat is both complete and cocomplete. Limits and colimits of diagrams in Cat are constructed using the limits and respectively colimits in Set of the diagrams obtained by post-composing the original diagrams with the functors Obj : Cat → Set (taking a small category to its set of objects) and Arr : Cat → Set (taking a small category to its set of arrows).

Remark 2.1.33. Replacing categories with quasi-categories (as target categories) in the definition of limits and colimits yields notions of *small limit* and *small colimit* in a quasi-category, as well as of *small completeness* and *small cocompleteness* of quasi-categories. Then, the quasi-category CAT is both small complete and small cocomplete, with small limits and small colimits in CAT being computed similarly to limits and colimits in Cat (see e.g. [Sch72]).

Instantiating[2] Definition 2.1.21 yields notions of *limit preservation*, *limit reflection*, *limit creation* and *limit lifting* by a functor.

[2] See also Example 2.1.50.

Definition 2.1.34. *Let* D *denote a small category. A functor* $\mathsf{F} : \mathsf{C} \to \mathsf{E}$ **preserves** D-**limits**, *or* **is** D-**continuous** *if and only if for any* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$, *if* $(L, (l_D)_{D \in |\mathsf{D}|})$ *is a limit for* d *in* C, *then* $(\mathsf{F}L, (\mathsf{F}l_D)_{D \in |\mathsf{D}|})$ *is a limit for* Fd *in* E. F **reflects** D-**limits** *if and only if for any* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$, *if* $(\mathsf{F}L, (\mathsf{F}l_D)_{D \in |\mathsf{D}|})$ *is a limit for* Fd *in* E, *then* $(L, (l_D)_{D \in |\mathsf{D}|})$ *is a limit for* d *in* C. F **creates** D-**limits** *if and only if for any* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$, *if* $(K, (k_D)_{D \in |\mathsf{D}|})$ *is a limit for* Fd *in* E, *then there exists precisely one cone* $(L, (l_D)_{D \in |\mathsf{D}|})$ *on* d *in* C *such that* $\mathsf{F}L = K$ *and* $\mathsf{F}l_D = k_D$ *for any* $D \in |\mathsf{D}|$, *and moreover,* $(L, (l_D)_{D \in |\mathsf{D}|})$ *is a limit for* d *in* C. *Finally,* F **lifts** D-**limits** *if and only if for any* $\mathsf{d} : \mathsf{D} \to \mathsf{C}$, *if* $(K, (k_D)_{D \in |\mathsf{D}|})$ *is a limit for* Fd *in* E, *then there exists a limit* $(L, (l_D)_{D \in |\mathsf{D}|})$ *for* d *in* C *such that* $\mathsf{F}L = K$ *and* $\mathsf{F}l_D = k_D$ *for any* $D \in |\mathsf{D}|$.

Consequently, if a functor creates certain limits, then it also lifts those limits.

Proposition 2.1.35 ([AHS90]). *Let* D *denote a small category, and let* $\mathsf{F} : \mathsf{C} \to \mathsf{E}$ *denote an arbitrary functor. If* E *has* D*-limits and* F *lifts them, then* C *has* D*-limits and* F *preserves them.*

Corollary 2.1.36. *Let* D *denote a small category, and let* $\mathsf{F} : \mathsf{C} \to \mathsf{E}$ *denote an arbitrary functor. If* E *has* D*-limits and* F *creates them, then* C *has* D*-limits and* F *preserves them.*

Example 2.1.37. The class of *extended polynomial functors* from Set^S to Set is the least class containing:

1. the constant functor A with $A \in |\mathsf{Set}|$
2. the projection functor Π_s with $s \in S$
3. the product functor $\mathsf{F}_1 \times \ldots \times \mathsf{F}_n$, with $\mathsf{F}_1, \ldots, \mathsf{F}_n$ already in the class
4. the coproduct functor $\mathsf{F}_1 + \ldots + \mathsf{F}_n$, with $\mathsf{F}_1, \ldots, \mathsf{F}_n$ already in the class
5. the exponent functor F^A, with $A \in |\mathsf{Set}|$ and F already in the class.

The class of *polynomial functors* from Set^S to Set is the least class containing 1-4 above.

The class of *(extended) polynomial endofunctors* on Set^S consists of endofunctors $\mathsf{F} : \mathsf{Set}^S \to \mathsf{Set}^S$ such that F_s is an (extended) polynomial functor for each $s \in S$.

Extended polynomial (endo)functors preserve pullbacks, equalisers and limits of ω^{op}-chains. Also, polynomial (endo)functors preserve coequalisers of kernel pairs and colimits of ω-chains, but do not, in general, preserve pushouts or arbitrary coequalisers. Finally, polynomial (endo)functors preserve monomorphisms as well as epimorphisms.

Example 2.1.38. If $\phi : S \to S'$ is an arbitrary function, then the functor $\mathsf{U}_\phi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ taking $A' \in |\mathsf{Set}^{S'}|$ to $(A'_{\phi(s)})_{s \in S} \in |\mathsf{Set}^S|$ preserves limits and colimits. And if, in addition, ϕ is injective, then U_ϕ lifts limits and colimits. (Both statements follow from limits and colimits in Set^S and $\mathsf{Set}^{S'}$ being computed component-wise.)

Similarly, if $V \subseteq S$ and $V \subseteq S'$, if $\phi : S \to S'$ is such that $\phi|_V = \iota_V : V \to S'$, and if D is a V-indexed set, then the functor $\mathsf{U}_\phi : \mathsf{Set}^{S'}_D \to \mathsf{Set}^S_D$ taking $A' \in |\mathsf{Set}^{S'}_D|$ to $(A'_{\phi(s)})_{s \in S} \in |\mathsf{Set}^S_D|$ preserves limits and colimits. And if, in addition, ϕ is injective, then U_ϕ lifts limits and colimits.

The construction of pullbacks in Cat can be used to obtain the following result.

Proposition 2.1.39. *Let* C, C_1 *and* C_2 *denote small categories, let* $\mathsf{U}_i : \mathsf{C}_i \to \mathsf{C}$ *with* $i = 1, 2$ *denote arbitrary functors, and let* $\mathsf{U}'_i : \mathsf{C}' \to \mathsf{C}_i$ *with* $i = 1, 2$ *denote the pullback of* U_1 *and* U_2 *in* Cat. *If each of* C, C_1, C_2 *is* D*-complete (*D*-cocomplete) for some small category* D*, and if* U_1 *and* U_2 *lift* D*-limits (*D*-colimits), then* C' *is* D*-complete (*D*-cocomplete), while* U'_1 *and* U'_2 *lift* D*-limits (*D*-colimits).*

Proof (sketch). For a diagram $\mathsf{d} : \mathsf{D} \to \mathsf{C}'$ in C', the existence of D-limits (D-colimits) in C_1 together with the preservation of D-limits (D-colimits) by U_1 (see Proposition 2.1.35) and the lifting of D-limits (D-colimits) by U_2 ensure that there exist limits (colimits) for $\mathsf{U}'_1\mathsf{d}$ in C_1 and for $\mathsf{U}'_2\mathsf{d}$ in C_2 whose images under U_1 and respectively U_2 coincide. These limits (colimits) uniquely determine a limit (colimit) for d in C'. □

Remark 2.1.40. Proposition 2.1.39 generalises to arbitrary categories C, C_1 and C_2, with the pullback being taken in the quasi-category CAT.

The following result concerns the existence of limits in categories of functors.

Proposition 2.1.41. *Let* C *and* D *denote small categories, and let* E *denote a* D*-complete (respectively* D*-cocomplete) category. Then, any* D*-shaped diagram* $\mathsf{d} : \mathsf{D} \to [\mathsf{C}, \mathsf{E}]$ *has a limit (colimit), and this limit (colimit) is computed pointwise. Furthermore, if each of* $\mathsf{d}D$ *with* $D \in |\mathsf{D}|$ *is* I*-continuous (*I*-cocontinuous), for some small category* I*, then so is the limit (colimit) of* d*. Finally, if each of* $\mathsf{d}D$ *with* $D \in |\mathsf{D}|$ *preserves monomorphisms (epimorphisms), then so does the limit (colimit) of* d*.*

Remark 2.1.42. Proposition 2.1.41 generalises to the case when C is an arbitrary category. (In this case, $[\mathsf{C}, \mathsf{E}]$ is a quasi-category.)

The next result summarises some properties of monomorphisms, pullbacks, equalisers and kernel pairs which will be used in the following chapters.

Proposition 2.1.43. *The following hold:*

1. *Any equaliser is a monomorphism.*
2. *The pullback of a monomorphism along an arbitrary arrow, if it exists, is itself a monomorphism.*
3. *An arrow* $f : C \to D$ *in a category* C *is a monomorphism if and only if the following is a pullback in* C*:*

$$\begin{array}{ccc} C & \xrightarrow{1_C} & C \\ {\scriptstyle 1_C}\downarrow & & \downarrow{\scriptstyle f} \\ C & \xrightarrow[f]{} & D \end{array}$$

4. *If a functor preserves kernel pairs, then it also preserves monomorphisms.*

5. *The kernel pair of an arrow defines a relation on the domain of that arrow.*
6. *If a kernel pair has a coequaliser, it is the kernel pair of its coequaliser.*
7. *If a coequaliser has a kernel pair, it is the coequaliser of its kernel pair.*
8. *If a category has binary products and pullbacks, then it also has equalisers.*

(Dual results hold for epimorphisms, pushouts, coequalisers and cokernel pairs.)

A concept which will prove particularly useful in the following is that of a *regular epimorphism* (respectively *regular monomorphism*), given by an epimorphism (monomorphism) which is also a coequaliser (equaliser).

Example 2.1.44. In Set, all epimorphisms and monomorphisms are regular – every epimorphism is the coequaliser of its kernel pair, while every monomorphism is the equaliser of its characteristic mapping and of a constant mapping (see e.g. [Bor94a, p. 143]).

Definition 2.1.45. *A category* C *is called* **regular** *if and only if the following hold:*

1. *every arrow has a kernel pair*
2. *every kernel pair has a coequaliser*
3. *the pullback of a regular epimorphism along an arbitrary arrow exists and is a regular epimorphism.*

Proposition 2.1.46. *In a regular category, every arrow f has a factorisation of form $f = m \circ e$ with m a monomorphism and e a regular epimorphism, and moreover, this factorisation is unique up to isomorphism. (Such a factorisation is called a* **regular-epi-mono factorisation**.*)*

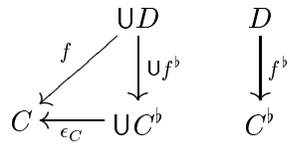
Proof (sketch). The C-arrows e and m defining the regular-epi-mono factorisation of $f : A \to C$ are given by the quotient $e : A \to B$ of f and respectively by the unique C-arrow $m : B \to C$ satisfying $f = m \circ e$ resulting from e being a coequaliser. □

Example 2.1.47. Set is regular (as all epimorphisms are stable under pullbacks), and so are Set^S and Set^S_D, with S an index set and D a V-sorted set for some $V \subseteq S$.

Remark 2.1.48. The notion of regular category used here, taken from [Bor94b], does not appear to be the standard one. An alternative definition of regular categories can for instance be found in [AHS90, FS90], where it is required that the category in question is finitely complete and admits regular-epi-mono factorisations, and that regular epimorphisms are stable under pullbacks. One can, however, show that these conditions imply those in Definition 2.1.45, and moreover, provided that the category in question in finitely complete, the converse also holds. All the regular categories considered in the following are finitely complete.

General universal properties are expressed using the notion of *(co)universal arrow*.

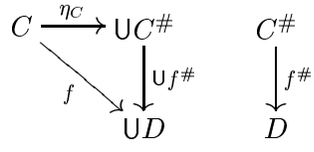
Definition 2.1.49. *Let* U : D → C *denote a functor, and let* $C \in |\mathsf{C}|$. *A* **couniversal arrow** *from* U *to* C *is a* C*-arrow* $\epsilon_C : \mathsf{U}C^\flat \to C$, *with* $C^\flat$ *a* D*-object, such that for any* D*-object* D *and* C*-arrow* $f : \mathsf{U}D \to C$, *there exists a unique* D*-arrow* $f^\flat : D \to C^\flat$ *satisfying* $f = \epsilon_C \circ \mathsf{U}f^\flat$.



$$\mathsf{C} \underset{\mathsf{U}}{\longleftarrow} \mathsf{D}$$

The D*-object* $C^\flat$ *is said to be* **cofree** *over* C *w.r.t.* U.

Given U *and* C *as before, a* **universal arrow** *from* C *to* U *is a* C*-arrow* $\eta_C : C \to \mathsf{U}C^\#$, *with* $C^\#$ *a* D*-object, such that for any* D*-object* D *and* C*-arrow* $f : C \to \mathsf{U}D$, *there exists a unique* D*-arrow* $f^\# : C^\# \to D$ *satisfying* $f = \mathsf{U}f^\# \circ \eta_C$.



$$\mathsf{C} \underset{\mathsf{U}}{\longleftarrow} \mathsf{D}$$

The D*-object* $C^\#$ *is said to be* **free** *over* C *w.r.t.* U.

Example 2.1.50. Let D denote a small category, and let $\Delta : \mathsf{C} \to [\mathsf{D}, \mathsf{C}]$ denote the *diagonal functor*, defined by:

- $\Delta(C)(D) = C$ for $D \in |\mathsf{D}|$ and $\Delta(C)(d) = 1_C$ for $d \in \| \mathsf{D} \|$, for $C \in |\mathsf{C}|$
- $\Delta(c)_D = c$ for $D \in |\mathsf{D}|$ and $c \in \| \mathsf{C} \|$.

Then, a limit for a diagram d : D → C is the same as a couniversal arrow from Δ to d, while a colimit for d is the same as a universal arrow from d to Δ.

Remark 2.1.51. Let U : D → C denote an arbitrary functor. If, for any C-object C, there exists a couniversal arrow $\epsilon_C : \mathsf{U}C^\flat \to C$ from U to C, then the correspondence $C \mapsto C^\flat$ extends uniquely to a functor R : C → D, called a *right adjoint to* U. The action of R on arrows is given by:

$$(c : C \to C') \in \| C \| \quad \mapsto \quad ((c \circ \epsilon_C)^\flat : C^\flat \to C'^\flat) \in \| D \|$$

Moreover, the couniversal arrows $\epsilon_C : \mathsf{U}C^\flat \to C$, $C \in |\mathsf{C}|$ define a natural transformation $\epsilon : \mathsf{UR} \Rightarrow \mathsf{Id}_\mathsf{C}$, called the *counit* of the adjunction. If, in addition, ϵ is the identity natural transformation $1_{\mathsf{Id}_\mathsf{C}}$, then R is called a *right adjoint, right inverse* to U.

Similarly, any family of universal arrows $\eta_C : C \to \mathsf{U}C^\#$, $C \in |\mathsf{C}|$ yields a functor L : C → D, called a *left adjoint to* U, mapping $C \in |\mathsf{C}|$ to $C^\# \in |\mathsf{D}|$ and $(c : C \to C') \in \| \mathsf{C} \|$ to $(\eta_{C'} \circ c)^\# \in \| \mathsf{D} \|$,

in such a way that the universal arrows $\eta_C : C \to \mathsf{U}C^{\#}$, $C \in |\mathsf{C}|$ define a natural transformation $\eta : \mathsf{Id}_{\mathsf{C}} \Rightarrow \mathsf{UL}$, called the *unit* of the adjunction. If η is the identity natural transformation $1_{\mathsf{Id}_{\mathsf{C}}}$, then L is called a *left adjoint, right inverse* to U.

Example 2.1.52. If $\phi : S \to S'$ is an arbitrary function, then the functor $\mathsf{U}_\phi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ taking $A' \in |\mathsf{Set}^{S'}|$ to $(A'_{\phi(s)})_{s \in S} \in |\mathsf{Set}^S|$ has both a right adjoint $\mathsf{R} : \mathsf{Set}^S \to \mathsf{Set}^{S'}$ and a left adjoint $\mathsf{L} : \mathsf{Set}^S \to \mathsf{Set}^{S'}$. (R takes $A \in |\mathsf{Set}^S|$ to $(\prod_{\phi(s)=s'} A_s)_{s' \in S'} \in |\mathsf{Set}^{S'}|$, while L takes $A \in |\mathsf{Set}^S|$ to $(\coprod_{\phi(s)=s'} A_s)_{s' \in S'} \in |\mathsf{Set}^{S'}|$.) And if, in addition, ϕ is injective, then R and L are also right inverses to U_ϕ.

Similarly, if $V \subseteq S$ and $V \subseteq S'$, if $\phi : S \to S'$ is such that $\phi{\restriction_V} = \iota_V : V \to S'$, and if D is a V-indexed set, then the functor $\mathsf{U}_\phi : \mathsf{Set}_D^{S'} \to \mathsf{Set}_D^S$ taking $A' \in |\mathsf{Set}_D^{S'}|$ to $(A'_{\phi(s)})_{s \in S} \in |\mathsf{Set}_D^S|$ has both a right adjoint and a left adjoint. And if, in addition, ϕ is injective, then these adjoints are also right inverses to U_ϕ.

Proposition 2.1.53. *Let* $\mathsf{L} : \mathsf{C} \to \mathsf{D}$ *and* $\mathsf{R} : \mathsf{D} \to \mathsf{C}$ *be functors. If* L *is a left adjoint to* R *then* R *is a right adjoint to* L*, and conversely.*

One writes $\mathsf{L} \dashv \mathsf{R}$ whenever L is a left adjoint to R.

Proposition 2.1.54. *Right adjoints preserve limits, while left adjoints preserve colimits.*

Then, 3 of Proposition 2.1.43 yields the following.

Corollary 2.1.55. *Right adjoints preserve monomorphisms, while left adjoints preserve epimorphisms.*

2.2 General Logics

Formal specification and verification techniques employ a large variety of logics. Common features of these logics include notions of *sentence*, *inference* of sentences from collections of sentences, *model*, and *satisfaction* of sentences by models. The notions of *entailment system* [Mes89] and *institution* [GB92] capture the essence of the syntactical and respectively semantical aspects of a specification logic, while abstracting away the syntactic and semantic details specific to particular logics.

We begin with the notion of institution. Its main ingredients are: a collection of *signatures* (providing a syntax for constructing sentences) and *signature morphisms* (providing translations between syntaxes), together with, for each signature Σ, a collection of Σ-*models*, a collection of Σ-*sentences*, and a *satisfaction relation* between Σ-models and Σ-sentences. These are subject to a constraint formalising the statement that *truth is invariant under changes of notation* [GB92].

Definition 2.2.1 ([GB92]). *A* **logical system** *is a tuple* $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$*, where:*

1. Sign *is a (quasi-)category whose objects are called* **signatures**

2. $\mathsf{Mod} : \mathsf{Sign} \to \mathsf{CAT}^{\mathsf{op}}$ *is a functor giving, for each signature* Σ*, a category* $\mathsf{Mod}(\Sigma)$ *whose objects are called* Σ-**models** *and whose arrows are called* Σ-**homomorphisms**
3. $\mathsf{Sen} : \mathsf{Sign} \to \mathsf{SET}$ *is a functor giving, for each signature, a class of* **sentences** *over that signature*
4. $\models$ *is a signature-indexed family of relations* $(\models_\Sigma)_{\Sigma \in |\mathsf{Sign}|}$ *with, for* $\Sigma \in |\mathsf{Sign}|$*, the relation* $\models_\Sigma \subseteq |\mathsf{Mod}(\Sigma)| \times \mathsf{Sen}(\Sigma)$ *being called* Σ-**satisfaction**.

An **institution** *is a logical system* $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ *additionally satisfying:*

$$M' \models_{\Sigma'} \mathsf{Sen}(\phi)(e) \quad \textit{if and only if} \quad \mathsf{Mod}(\phi)(M') \models_\Sigma e$$

for any $(\phi : \Sigma \to \Sigma') \in \|\mathsf{Sign}\|$, $M' \in |\mathsf{Mod}(\Sigma')|$ *and* $e \in \mathsf{Sen}(\Sigma)$.

The defining condition of institutions is called the *satisfaction condition*.

Remark 2.2.2. The definition of institutions in [GB92] did not involve defining logical systems beforehand. However, in order to also account for logics in which the satisfaction condition does not necessarily hold, here the original definition has been split accordingly. Also, the definition of institutions in [GB92] involved functors $\mathsf{Mod} : \mathsf{Sig} \to \mathsf{Cat}^{\mathsf{op}}$ and $\mathsf{Sen} : \mathsf{Sig} \to \mathsf{Set}$, as opposed to functors $\mathsf{Mod} : \mathsf{Sig} \to \mathsf{CAT}^{\mathsf{op}}$ and $\mathsf{Sen} : \mathsf{Sig} \to \mathsf{SET}$. The generalisation considered here is necessary in order to give an institutional account of logics with abstract notions of signature and sentence (see Sections 3.1.3, 3.2.3 and 3.3.3).

In a logical system $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ one writes U_ϕ for $\mathsf{Mod}(\phi)$, and ϕ for $\mathsf{Sen}(\phi)$, with $(\phi : \Sigma \to \Sigma') \in \| \mathsf{Sign} \|$. The functors U_ϕ are called *reduct functors*. Also, one writes $M' \restriction_\Sigma$ (respectively $f' \restriction_\Sigma$) as a shorthand for $\mathsf{U}_\phi M'$ ($\mathsf{U}_\phi f'$), with $(\phi : \Sigma \to \Sigma') \in \| \mathsf{Sign} \|$, $M' \in |\mathsf{Mod}(\Sigma')|$ and $f' \in \| \mathsf{Mod}(\Sigma') \|$.

Definition 2.2.3. *Let* $\mathcal{L} = (\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ *denote a logical system. A* Σ'*-model* M' *is a* Σ'-**extension** *(respectively* **coextension***) of a* Σ*-model* M *along a signature morphism* $\phi : \Sigma \to \Sigma'$ *if and only if there exists a* Σ*-homomorphism* $f : M \to M' \restriction_\Sigma$ *(*$f : M' \restriction_\Sigma \to M$*).*

Definition 2.2.4. *Let* $\mathcal{L} = (\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ *denote an institution.*

1. *An* **(**$\mathcal{L}$**-)specification** *is a pair* (Σ, E) *with* $\Sigma \in |\mathsf{Sign}|$ *and* $E \subseteq \mathsf{Sen}(\Sigma)$.
2. *A* Σ*-model* M **satisfies** *a specification* (Σ, E) *(written* $M \models_\Sigma E$*) if and only if* $M \models_\Sigma e$ *for each* $e \in E$.
3. *An* $\mathcal{L}$*-specification is* **consistent** *if and only if it has at least one model.*
4. *A* Σ*-sentence* e *is* **semantically entailed** *by a class* E *of* Σ*-sentences (written* $E \models_\Sigma e$*) if and only if* $M \models_\Sigma E$ *implies* $M \models_\Sigma e$ *for any* $M \in |\mathsf{Mod}(\Sigma)|$.
5. *An* **(**$\mathcal{L}$**-)theory** *is an* $\mathcal{L}$*-specification* (Σ, E) *such that* E *is closed w.r.t.* $\models_\Sigma$*. The closure of a class* E *of* Σ*-sentences under* $\models_\Sigma$ *is denoted* E^*.

6. *A signature morphism* $\phi : \Sigma \to \Sigma'$ *defines an* **($\mathcal{L}$-)specification morphism** $\phi : (\Sigma, E) \to (\Sigma', E')$ *if and only if* $\mathsf{Sen}(\phi)(E) \subseteq E'^*$.
7. *An* **($\mathcal{L}$-)theory morphism** *is an $\mathcal{L}$-specification morphism between $\mathcal{L}$-theories.*

In an institution $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$, one writes $\mathsf{Mod}(\Sigma, E)$ for the full subcategory of $\mathsf{Mod}(\Sigma)$ whose objects satisfy the specification (Σ, E). Then, specification morphisms $\phi : (\Sigma, E) \to (\Sigma', E')$ induce theory morphisms $\phi : (\Sigma, E^*) \to (\Sigma', E'^*)$ on the one hand (see [GB92, Lemma 8]), and *reduct functors* $\mathsf{U}_\phi : \mathsf{Mod}(\Sigma', E') \to \mathsf{Mod}(\Sigma, E)$ on the other.

If Spec denotes the category of specifications and specification morphisms of an institution, then diagrams in Spec can be used to formalise the relationships that exist between the components of the system being specified, while colimits in Spec yield a canonical way to combine existing specifications of the system components into a specification of the system as a whole. The next two results concern the existence of colimits in Spec.

Theorem 2.2.5 ([GB92]). *Let* $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ *denote an institution, let* Spec *denote its category of specifications, and let* $\mathsf{Sig} : \mathsf{Spec} \to \mathsf{Sign}$ *denote the functor taking specifications to their underlying signatures. Then,* Sig *lifts colimits.*

Corollary 2.2.6 ([GB92]). *If the category of signatures of an institution is finitely cocomplete, then so is its category of specifications.*

The notion of entailment employed by institutions is based on the satisfaction of sentences by models. A different notion of entailment, based on the inference of sentences from collections of sentences according to specified rules is captured by *entailment systems*.

Definition 2.2.7 ([Mes89]). *An* **entailment system** *is a triple* $(\mathsf{Sign}, \mathsf{Sen}, \vdash)$*, where:*

1. Sign *is a (quasi-)category whose objects are called* **signatures**
2. $\mathsf{Sen} : \mathsf{Sign} \to \mathsf{SET}$ *is a functor giving, for each signature, a class of* **sentences** *over that signature*
3. $\vdash$ *is a signature-indexed family of relations* $(\vdash_\Sigma)_{\Sigma \in |\mathsf{Sign}|}$ *with, for* $\Sigma \in |\mathsf{Sign}|$*, the relation* $\vdash_\Sigma \ \subseteq \ \mathcal{P}(\mathsf{Sen}(\Sigma)) \times \mathsf{Sen}(\Sigma)$ *being called* Σ-**entailment**

such that the following hold:

1. $\{e\} \vdash_\Sigma e$*, for* $e \in \mathsf{Sen}(\Sigma)$ *(reflexivity)*
2. $E \vdash_\Sigma e$ *and* $E \subseteq E'$ *imply* $E' \vdash_\Sigma e$ *(monotonicity)*
3. $E \vdash_\Sigma e_i$ *for* $i \in I$ *and* $\{e_i \mid i \in I\} \vdash_\Sigma e$ *imply* $E \vdash_\Sigma e$ *(transitivity)*
4. $E \vdash_\Sigma e$ *implies* $\mathsf{Sen}(\phi)(E) \vdash_{\Sigma'} \mathsf{Sen}(\phi)(e)$*, for* $\phi : \Sigma \to \Sigma'$ *($\vdash$-translation)*

A desirable property of any specification logic which involves both an institution and an entailment system is the existence of a certain compatibility between its two notions of entailment, in a sense made precise below.

Definition 2.2.8. *Let* Sign, Mod, Sen, $\models$, *and* $\vdash$ *be such that* $(\mathsf{Sign}, \mathsf{Mod}, \mathsf{Sen}, \models)$ *is a logical system and* $(\mathsf{Sign}, \mathsf{Sen}, \vdash)$ *is an entailment system. Then,* $\vdash$ *is* **sound** *(respectively* **complete***) for* $\models$ *if and only if* $E \vdash_\Sigma e$ *implies* $E \models_\Sigma e$ *(*$E \models_\Sigma e$ *implies* $E \vdash_\Sigma e$*) for any* $\Sigma \in |\mathsf{Sign}|$, $E \subseteq \mathsf{Sen}(\Sigma)$ *and* $e \in \mathsf{Sen}(\Sigma)$.

2.3 Algebraic Specification

2.3.1 Many-Sorted Algebra

The theory of *many-sorted algebras* [GTW78] is a generalisation of the theory of universal algebras that uses multiple sorts to specify collections of types (as opposed to single types). A brief account of the use of many-sorted algebras for the specification of data types is given in the following.

Definition 2.3.1. *A* **(many-sorted) signature** *is a pair* (S, Σ)*, with* S *a* **sort set** *and* Σ *an* $S^* \times S$*-sorted set of* **operation symbols** *(with* S^* *denoting the set of finite sequences of elements of* S*). Operation symbols* $\sigma \in \Sigma_{\epsilon,s}$ *(with* ϵ *denoting the empty sequence) are called* **constant symbols**.

The sorts in a many-sorted signature denote types, while the operation symbols denote type constructors.

Many-sorted signatures (S, Σ) are abbreviated Σ whenever the context allows it. If Σ denotes a many-sorted signature with sort set S, one writes $\sigma : s_1 \dots s_n \to s$ for $\sigma \in \Sigma_{s_1 \dots s_n, s}$, and Σ_s for the subsignature $\{\sigma : s_1 \dots s_n \to s \mid s_1, \dots, s_n \in S\}$ with $s \in S$.

Example 2.3.2. The many-sorted signature $\Sigma_{\mathtt{NAT}}$ consisting of a sort `Nat`, a constant symbol $0 : \to \mathtt{Nat}$ and a unary operation symbol $\mathtt{succ} : \mathtt{Nat} \to \mathtt{Nat}$ specifies natural numbers.

Particular values of the types specified by a many-sorted signature can be referred to by means of *algebraic terms*.

Definition 2.3.3. *Let* Σ *denote a many-sorted signature with sort set* S*. The* S*-sorted set* T_Σ *of* **ground** Σ**-terms** *is defined inductively as follows:*

1. $\sigma \in T_{\Sigma,s}$ *for* $\sigma \in \Sigma_{\epsilon,s}$
2. $\sigma(t_1, \dots, t_n) \in T_{\Sigma,s}$ *for* $\sigma \in \Sigma_{s_1 \dots s_n, s}$ *and* $t_i \in T_{\Sigma,s_i}$, $i = 1, \dots, n$.

If $\mathcal{V}$ *denotes an* S*-sorted set of* **variables**, *the* S*-sorted set* $T_\Sigma(\mathcal{V})$ *of* Σ**-terms with variables in** $\mathcal{V}$ *is the* S*-sorted set of* $\Sigma \cup \{V : \to s \mid V \in \mathcal{V}_s,\ s \in S\}$*-terms.*

That is, Σ-terms are generated by the operation symbols of Σ, while Σ-terms with variables in $\mathcal{V}$ are generated by the operation symbols of Σ together with the variables in $\mathcal{V}$ (regarded as constant symbols of an enriched signature).

Example 2.3.4. `succ(0)`, `succ(succ(0))`, ... are ground $\Sigma_{\mathtt{NAT}}$-terms of sort `Nat`.

One writes $V : s$ for $V \in \mathcal{V}_s$ and $t : s$ for $t \in T_\Sigma(\mathcal{V})_s$, with $s \in S$.

Definition 2.3.5. *Let Σ denote a many-sorted signature, let $t \in T_\Sigma(\{V_1, \ldots, V_n\})$ with $V_i : s_i$ for $i = 1, \ldots, n$, and let $t_i \in T_\Sigma(\mathcal{V})_{s_i}$ for $i = 1, \ldots, n$. The* **substitution** *of $t_1 \ldots, t_n$ for $V_1, \ldots, V_n$ in t, denoted $t(t_1/V_1, \ldots, t_n/V_n)$ ($t(\overline{t}/\overline{V})$ for short), is defined inductively as follows:*

1. $V_i(\overline{t}/\overline{V}) = t_i$ *for* $i \in \{1, \ldots, n\}$
2. $(\sigma(t'_1, \ldots, t'_m))(\overline{t}/\overline{V}) = \sigma(t'_1(\overline{t}/\overline{V}), \ldots, t'_m(\overline{t}/\overline{V}))$
   for $\sigma \in \Sigma_{s'_1 \ldots s'_m, s}$ *and* $t'_i \in T_\Sigma(\{V_1, \ldots, V_n\})_{s'_i}$, $i = 1, \ldots, n$.

Many-sorted algebras interpret the sorts and operation symbols in many-sorted signatures as sets and respectively functions on these sets, while *algebra homomorphisms* provide structure-preserving mappings from one algebra to another.

Definition 2.3.6. *Let Σ denote a many-sorted signature with sort set S. A* **(many-sorted) Σ-algebra** *is an S-sorted set A together with, for each $\sigma \in \Sigma_{w,s}$ with $w \in S^*$ and $s \in S$, a function $\sigma_A : A_w \to A_s$ (with $A_\epsilon = \{*\}$ and $A_{s_1 \ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$ for $n \geq 1$ and $s_1, \ldots, s_n \in S$). The S-sorted set A is called the* **carrier** *of the algebra. A* **(many-sorted) Σ-homomorphism** *between Σ-algebras A and B is an S-sorted function $f : A \to B$ additionally satisfying:*

1. $f_s(\sigma_A) = \sigma_B$ *for* $\sigma \in \Sigma_{\epsilon,s}$
2. $f_s(\sigma_A(a_1, \ldots, a_n)) = \sigma_B(f_{s_1}(a_1), \ldots, f_{s_n}(a_n))$ *for* $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ *and* $a_i \in A_{s_i}$, $i = 1, \ldots, n$.

For a many-sorted signature Σ, the category of Σ-algebras and Σ-homomorphisms is denoted $\mathrm{Alg}(\Sigma)$.

The interpretation of Σ-operation symbols by Σ-algebras extends to an interpretation of Σ-terms by Σ-algebras.

Definition 2.3.7. *Let Σ denote a many-sorted signature, and let $\mathcal{V}$ denote a set of variables. The* **interpretation** *of a Σ-term $t \in T_\Sigma(\mathcal{V})_s$, with $s \in S$, in a Σ-algebra A is a function $t_A : \prod_{X \in \mathcal{V}, X : s'} A_{s'} \to A_s$ defined as follows:*

1. $X_A = \pi_X$ *for* $X \in \mathcal{V}_s$
2. $(\sigma(t_1, \ldots, t_n))_A = \sigma_A \circ \langle (t_1)_A, \ldots, (t_n)_A \rangle$ *for* $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ *and* $t_i \in T_\Sigma(\mathcal{V})_{s_i}$, $i = 1, \ldots, n$

with $\langle (t_1)_A, \ldots, (t_n)_A \rangle : \prod_{X \in \mathcal{V}, X : s'} A_{s'} \to A_{s_1} \times \ldots \times A_{s_n}$ denoting the unique Set*-arrow induced by $(t_i)_A : \prod_{X \in \mathcal{V}, X : s'} A_{s'} \to A_{s_i}$ with $i = 1, \ldots, n$.*

Theorem 2.3.8 ([GTW78]). *Let Σ denote a many-sorted signature with sort set S. Then, the S-sorted set T_Σ carries a Σ-algebra structure. Moreover, this Σ-algebra, known as the* **term Σ-algebra**, *is initial in* $\mathrm{Alg}(\Sigma)$.

Remark 2.3.9. The unique Σ-homomorphism from the term Σ-algebra to an arbitrary Σ-algebra A takes a Σ-term $t \in T_{\Sigma,s}$ to its interpretation $t_A \in A_s$.

Corollary 2.3.10. *Let Σ denote a many-sorted signature with sort set S, and let $\mathcal{V}$ denote an S-sorted set of variables. Then, the S-sorted set $T_\Sigma(\mathcal{V})$ carries a Σ-algebra structure.*

Proof. The conclusion follows from Theorem 2.3.8, by taking Σ to be the signature $\Sigma \cup \{V : \rightarrow s \mid V \in \mathcal{V}_s,\ s \in S\}$. □

Definition 2.3.11. *Let Σ denote a many-sorted signature. A Σ-algebra A is* **reachable** *if and only if $!_{A,s}$ is surjective for each $s \in S$, where $!_A : T_\Sigma \rightarrow A$ denotes the unique Σ-homomorphism from T_Σ to A.*

Example 2.3.12. An initial $\Sigma_{\mathtt{NAT}}$-algebra is the algebra I interpreting:

- the sort `Nat` as the set $I_{\mathtt{Nat}} = \{\mathtt{0},\ \mathtt{succ(0)},\ \mathtt{succ(succ(0))},\ \ldots\}$
- the constant symbol `0` as the element $\mathtt{0}_I = \mathtt{0}$ of $I_{\mathtt{Nat}}$
- the unary operation symbol `succ` as the function $\mathtt{succ}_I : I_{\mathtt{Nat}} \rightarrow I_{\mathtt{Nat}}$ defined by: $\mathtt{succ}_I(\mathtt{0}) = \mathtt{succ(0)}$, $\mathtt{succ}_I(\mathtt{succ(0)}) = \mathtt{succ(succ(0))}$, ...

Initial algebras satisfy an *induction principle*, which exploits the existence and uniqueness of homomorphisms into any other algebra. Specifically, the existence of such homomorphisms yields a *definition principle* for functions on the initial algebra, while the uniqueness of such homomorphisms yields a *proof principle* for inductively defined functions.

Example 2.3.13. Given the many-sorted signature $\Sigma_{\mathtt{NAT}}$ in Example 2.3.2, one can use induction to define a function $\mathtt{even} : T_{\Sigma_{\mathtt{NAT}}} \rightarrow \mathtt{Bool} = \{\mathtt{true},\ \mathtt{false}\}$, which takes a $\Sigma_{\mathtt{NAT}}$-term t to the boolean value `true`, respectively `false`, depending on whether the natural number denoted by t is even or not. After appropriately defining a $\Sigma_{\mathtt{NAT}}$-structure on `Bool`, the $\Sigma_{\mathtt{NAT}}$-homomorphism from $T_{\Sigma_{\mathtt{NAT}}}$ to `Bool` resulting from the initiality of T_Σ provides a definition for `even`. The desired $\Sigma_{\mathtt{NAT}}$-structure on `Bool` is given by:

- $\mathtt{0}_{\mathtt{Bool}} : \{*\} \rightarrow \mathtt{Bool},\ \ \mathtt{0}_{\mathtt{Bool}} = \mathtt{true}$
- $\mathtt{succ}_{\mathtt{Bool}} : \mathtt{Bool} \rightarrow \mathtt{Bool},\ \ \mathtt{succ}_{\mathtt{Bool}}(\mathtt{true}) = \mathtt{false},\ \ \mathtt{succ}_{\mathtt{Bool}}(\mathtt{false}) = \mathtt{true}$

Furthermore, one can use induction to prove that $\mathtt{even}(\mathtt{succ}(\mathtt{succ}(t))) = \mathtt{even}(t)$ holds for any $\Sigma_{\mathtt{NAT}}$-term t. This is a consequence of the uniqueness of a $\Sigma_{\mathtt{NAT}}$-homomorphism from $T_{\Sigma_{\mathtt{NAT}}}$ to `Bool`, together with the observation that the function taking a $\Sigma_{\mathtt{NAT}}$-term t to the boolean value $\mathtt{even}(\mathtt{succ}(\mathtt{succ}(t)))$ defines such a homomorphism:

- $\mathtt{even}(\mathtt{succ}(\mathtt{succ}(\mathtt{0}))) = \mathtt{succ}_{\mathtt{Bool}}(\mathtt{succ}_{\mathtt{Bool}}(\mathtt{0}_{\mathtt{Bool}})) = \mathtt{true} = \mathtt{0}_{\mathtt{Bool}}$
- $\mathtt{even}(\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(t)))) = \mathtt{succ}_{\mathtt{Bool}}(\mathtt{even}(\mathtt{succ}(\mathtt{succ}(t))))$

For a many-sorted signature with sort set S, there exists a canonical way of extending an S-sorted set to a Σ-algebra.

Theorem 2.3.14 ([GTW78]). *Let Σ denote a many-sorted signature with sort set S, and let $\mathsf{U} : \mathsf{Alg}(\Sigma) \to \mathsf{Set}^S$ denote the functor taking Σ-algebras to their carrier. Then, for any $A \in |\mathsf{Set}^S|$, there exists $A^\# \in |\mathsf{Alg}(\Sigma)|$ free over A w.r.t. U.*

Proof (sketch). For $A \in |\mathsf{Set}^S|$, $A^\#$ is given by an initial algebra of the signature $\Sigma_A = \Sigma \cup \{a : \to s \mid a \in A_s,\ s \in S\}$. □

Remark 2.3.15. The free Σ-algebra over the empty S-sorted set is isomorphic to T_Σ, while the free Σ-algebra over an S-sorted set $\mathcal{V}$ of variables is isomorphic to $T_\Sigma(\mathcal{V})$.

Equations are used to constrain the values associated to data types, namely by identifying values constructed in different ways.

Definition 2.3.16. *Let Σ denote a many-sorted signature with sort set S. A* **(many-sorted) Σ-equation** *is a tuple $(\mathcal{V}, l, r)$ (also denoted $(\forall\, \mathcal{V})\ l = r$) with $\mathcal{V}$ an S-sorted set (of variables) and $l, r \in T_\Sigma(\mathcal{V})$. If $\mathcal{V} = \emptyset$, the equation is called* **ground**. *A Σ-algebra A* **satisfies** *a Σ-equation $(\forall\, \mathcal{V})\ l = r$ (written $A \models_\Sigma (\forall\, \mathcal{V})\ l = r$) if and only if for any* **assignment** *of values in A to the variables in $\mathcal{V}$, in the form of an S-sorted function $\theta : \mathcal{V} \to \mathsf{U}A$, $\theta^\#(l) = \theta^\#(r)$ (with $\theta^\# : T_\Sigma(\mathcal{V}) \to A$ denoting the unique extension of $\theta : \mathcal{V} \to \mathsf{U}A$ to a Σ-homomorphism).*

Remark 2.3.17. A variant of the notion of Σ-equation is that of a *conditional Σ-equation*, given by a tuple $(\mathcal{V}, (l, r), (l_1, r_1), \ldots, (l_n, r_n))$ (also denoted $(\forall\, \mathcal{V})\ l = r$ if $l_1 = r_1, \ldots, l_n = r_n$). A Σ-algebra A is said to *satisfy* a conditional equation of the above form if and only if, for any assignment $\theta : \mathcal{V} \to \mathsf{U}A$, $\theta^\#(l) = \theta^\#(r)$ holds whenever each of $\theta^\#(l_i) = \theta^\#(r_i)$ with $i = 1, \ldots, n$ does.

Translations between (the sorts and operation symbols of) many-sorted signatures are specified using *many-sorted signature morphisms*.

Definition 2.3.18. *Let Σ and Σ' denote many-sorted signatures with sort sets S and respectively S'. A* **(many-sorted) signature morphism** *$\phi : \Sigma \to \Sigma'$ consists of a function $\phi : S \to S'$ together with an $S^* \times S$-sorted function $(\phi_{w,s})_{w \in S^*, s \in S}$, with $\phi_{w,s} : \Sigma_{w,s} \to \Sigma'_{\phi^*(w), \phi(s)}$ for $w \in S^*$ and $s \in S$ (with ϕ^* denoting the pointwise extension of ϕ to a function from S^* to S'^*).*

The category of many-sorted signatures and signature morphisms is denoted Sign.

At the syntactic level, many-sorted signature morphisms $\phi : \Sigma \to \Sigma'$ induce (pointwise) translations of Σ-terms to Σ'-terms, and of Σ-equations to Σ'-equations. This yields a functor Eqn : Sign $\to$ SET, taking a signature Σ to the set of all Σ-equations, and a signature morphism $\phi : \Sigma \to \Sigma'$ to the function translating Σ-equations to Σ'-equations along ϕ.

Also, at the semantic level, many-sorted signature morphisms $\phi : \Sigma \to \Sigma'$ induce reduct functors $\mathsf{U}_\phi : \mathsf{Alg}(\Sigma') \to \mathsf{Alg}(\Sigma)$, taking Σ'-algebras to the Σ-algebras obtained by ignoring the interpretations of the Σ'-sorts and operation symbols that are not in the image of ϕ:

1. for $A \in |\mathsf{Alg}(\Sigma')|$, $\mathsf{U}_\phi A$ is given by:
   (a) $(\mathsf{U}_\phi A)_s = A_{\phi(s)}$ for $s \in S$
   (b) $\sigma_{\mathsf{U}_\phi A}(a_1, \ldots, a_n) = \phi(\sigma)_A(a_1, \ldots, a_n)$ for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $a_i \in (\mathsf{U}_\phi A)_{s_i}$, $i = 1, \ldots, n$
2. for $(f : A \to B) \in \| \mathsf{Alg}(\Sigma') \|$, $\mathsf{U}_\phi f$ is given by $(\mathsf{U}_\phi f)_s = f_{\phi(s)}$ for $s \in S$.

This yields a functor $\mathsf{Alg} : \mathsf{Sign} \to \mathsf{CAT}^{\mathsf{op}}$, taking a signature Σ to the category $\mathsf{Alg}(\Sigma)$, and a signature morphism $\phi : \Sigma \to \Sigma'$ to the functor $\mathsf{U}_\phi : \mathsf{Alg}(\Sigma') \to \mathsf{Alg}(\Sigma)$.

Furthermore, the satisfaction condition holds for the tuple $(\mathsf{Sign}, \mathsf{Alg}, \mathsf{Eqn}, \models)$.

Theorem 2.3.19 ([GB92]). $(\mathsf{Sign}, \mathsf{Alg}, \mathsf{Eqn}, \models)$ *is an institution.*

This institution is known as *many-sorted equational logic*, and has the property that its category of signatures is finitely cocomplete.

Theorem 2.3.20 ([GB84]). Sign *is finitely cocomplete.*

It then follows by Corollary 2.2.6 that the category Spec of many-sorted specifications and specification morphisms it itself finitely cocomplete.

Corollary 2.3.21 ([GB84]). Spec *is finitely cocomplete.*

The semantic constructions used to provide denotations for many-sorted specifications and specification morphisms are outlined in the following.

Definition 2.3.22. *Let Σ denote a many-sorted signature with sort set S, and let A denote a Σ-algebra. A Σ-***congruence** *on A is an S-sorted equivalence relation $\equiv$ on the carrier of A, additionally satisfying: $\sigma_A(a_1, \ldots, a_n) \equiv_s \sigma_A(a'_1, \ldots, a'_n)$ for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $a_i, a'_i \in A_{s_i}$ with $a_i \equiv_{s_i} a'_i$, $i = 1, \ldots, n$. The* **quotient** *of a Σ-algebra A by a Σ-congruence $\equiv$ on A is the Σ-algebra, denoted $A/_\equiv$, whose carrier is given by: $(A/_\equiv)_s = A_s/_{\equiv_s}$ for $s \in S$, and whose operations are given by: $\sigma_{A/_\equiv}([a_1], \ldots, [a_n]) = [\sigma_A(a_1, \ldots, a_n)]$ for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $a_i \in A_{s_i}$, $i = 1, \ldots, n$.*

If Σ denotes a many-sorted signature and A denotes a Σ-algebra, then any set E of Σ-equations induces a Σ-congruence $\equiv_{A,E}$ on A. Specifically, $\equiv_{A,E}$ is the least Σ-congruence containing $=_{A,E}$, with $=_{A,E} \subseteq A \times A$ being given by:

$$\theta^\#(l) =_{A,E} \theta^\#(r) \text{ whenever } (\forall\, \mathcal{V})\ l = r \text{ is in } E \text{ and } \theta : \mathcal{V} \to \mathsf{U}A$$

Theorem 2.3.14 then extends from signatures to specifications.

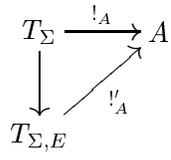
Theorem 2.3.23 ([GTW78]). *Let* $\mathsf{U} : \mathsf{Alg}(\Sigma, E) \to \mathsf{Set}^S$ *denote the functor taking (Σ, E)-algebras to their carrier. Then, for any $A \in |\mathsf{Set}^S|$, there exists $A^\# \in |\mathsf{Alg}(\Sigma, E)|$ free over A w.r.t.* U*.*

Proof (sketch). For $A \in |\mathsf{Set}^S|$, $A^\#$ is the quotient of the free Σ-algebra A' over A by the Σ-congruence $\equiv_{A',E}$ induced by E on A'. □

Theorem 2.3.23 provides a canonical way of extending an S-sorted set to a (Σ, E)-algebra. By Proposition 2.1.54, the free (Σ, E)-algebra over the empty S-sorted set is an initial (Σ, E)-algebra, denoted $T_{\Sigma,E}$. The Σ-algebra $T_{\Sigma,E}$ has the property that the ground Σ-equations it satisfies are precisely those semantically entailed by E.

Proposition 2.3.24. *Let Σ denote a many-sorted signature, let E denote a set of Σ-equations, and let e denote a ground Σ-equation. Then, $E \models_\Sigma e$ if and only if $T_{\Sigma,E} \models_\Sigma e$.*

Proof. For a (Σ, E)-algebra A, let $!'_A : T_{\Sigma,E} \to A$ denote the unique Σ-homomorphism from the initial (Σ, E)-algebra to A. Then, by initiality of T_Σ, the following diagram commutes:



That is, $t_A = !'_A(t_{T_{\Sigma,E}})$ for any ground Σ-term t. Hence, $l_{T_{\Sigma,E}} = r_{T_{\Sigma,E}}$ implies $l_A = r_A$ for any ground Σ-equation $(\forall \emptyset)\ l = r$. This proves the if direction.

The only if direction follows immediately from $T_{\Sigma,E}$ being a (Σ, E)-algebra. □

The preceding result justifies the use of initial algebras as denotations for many-sorted specifications.

An alternative way of interpreting a many-sorted specification is as the category of algebras satisfying it. This category enjoys a closure property which will be described in the following.

Definition 2.3.25. *Let Σ denote a many-sorted signature with sort set S, and let A denote a Σ-algebra. An S-sorted subset B of $|A|$ defines a Σ-***subalgebra** *of A if and only if B carries a Σ-algebra structure which makes the inclusion of B into $|A|$ a Σ-homomorphism. Also, a Σ-algebra C is a* **homomorphic image** *of A if and only if there exists a Σ-homomorphism $f : A \to C$ with f_s surjective for each $s \in S$.*

Proposition 2.3.26. *For a many-sorted signature Σ, the category $\mathsf{Alg}(\Sigma)$ has products.*

Definition 2.3.27. *Let Σ denote a many-sorted signature. A full subcategory of $\mathsf{Alg}(\Sigma)$ is a Σ-***variety** *if and only if it is closed under subalgebras, homomorphic images and arbitrary products.*

The previously-mentioned closure property of the category of algebras of a many-sorted specification amounts to this category being a variety. Furthermore, according to *Birkhoff's variety theorem*, any algebraic variety is characterisable by equations.

Theorem 2.3.28 ([MT92]). *Let Σ denote a many-sorted signature, and let $\mathcal{K}$ denote a full subcategory of $\mathsf{Alg}(\Sigma)$. Then, $\mathcal{K}$ is a variety if and only if $\mathcal{K}$ is equational (i.e. there exists a set E of Σ-equations such that $\mathcal{K} = \mathsf{Alg}(\Sigma, E)$).*

We now describe the semantic constructions used as denotations for signature morphisms and specification morphisms.

Theorem 2.3.29 ([TWW82]). *Let* $\mathsf{U}_\phi : \mathsf{Alg}(\Sigma') \to \mathsf{Alg}(\Sigma)$ *denote the reduct functor induced by a many-sorted signature morphism* $\phi : \Sigma \to \Sigma'$*. Then, for any* $A \in |\mathsf{Alg}(\Sigma)|$*, there exists* $A^\# \in |\mathsf{Alg}(\Sigma')|$ *free over* A *w.r.t.* U_ϕ*.*

Proof (sketch). For $A \in |\mathsf{Alg}(\Sigma)|$, $A^\#$ is the quotient of the free Σ'-algebra $T_{\Sigma'}(A')$ over the S'-sorted set $A' = (\coprod_{\phi(s)=s'} A_s)_{s' \in S'}$ by the least Σ'-congruence relating $\iota_s(\sigma_A(a_1, \ldots, a_n))$ with $\phi(\sigma)_{T_{\Sigma'}(A')}(\iota_{s_1}(a_1), \ldots, \iota_{s_n}(a_n))$ whenever $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $a_i \in A_{s_i}$, $i = 1, \ldots, n$. $\square$

Remark 2.3.30. The free Σ'-algebra $A^\#$ over the Σ-algebra A w.r.t. U_ϕ is the *least* Σ'-extension of A along ϕ (see Definition 2.2.3), in that $A^\#$ has a unique Σ'-homomorphism into any other such extension.

Theorem 2.3.29 generalises from signatures to specifications.

Theorem 2.3.31 ([TWW82]). *Let* $\mathsf{U}_\phi : \mathsf{Alg}(\Sigma', E') \to \mathsf{Alg}(\Sigma, E)$ *denote the reduct functor induced by a many-sorted specification morphism* $\phi : (\Sigma, E) \to (\Sigma', E')$*. Then, for any* $A \in |\mathsf{Alg}(\Sigma, E)|$*, there exists* $A^\# \in |\mathsf{Alg}(\Sigma', E')|$ *free over* A *w.r.t.* U_ϕ*.*

Proof (sketch). For $A \in |\mathsf{Alg}(\Sigma, E)|$, $A^\#$ is the quotient of the free Σ'-algebra over A w.r.t. U_ϕ by the Σ'-congruence induced on it by the equations in E'. $\square$

The following deduction calculus, known as *many-sorted equational deduction*, can be used to reason about the satisfaction of equations by algebras of many-sorted specifications:

$$[\,\textbf{base}\,] \quad \frac{}{E \vdash e} \quad e \in E$$

$$[\,\textbf{reflexivity}\,] \quad \frac{}{E \vdash (\forall\, \mathcal{V})\ t = t}$$

$$[\,\textbf{symmetry}\,] \quad \frac{E \vdash (\forall\, \mathcal{V})\ t = t'}{E \vdash (\forall\, \mathcal{V})\ t' = t}$$

$$[\,\textbf{transitivity}\,] \quad \frac{E \vdash (\forall\, \mathcal{V})\ t = t' \qquad E \vdash (\forall\, \mathcal{V})\ t' = t''}{E \vdash (\forall\, \mathcal{V})\ t = t''}$$

$$[\,\textbf{substitution}\,] \quad \frac{E \vdash (\forall\, X_1) \ldots (\forall\, X_n)\ t = t'}{E \vdash (\forall\, \mathcal{V})\ t(t_1/X_1, \ldots, t_n/X_n) = t'(t_1/X_1, \ldots, t_n/X_n)}$$

$$[\,\textbf{congruence}\,] \quad \frac{E \vdash (\forall\, \mathcal{V})\ t_1 = t_1' \ \ldots\ E \vdash (\forall\, \mathcal{V})\ t_n = t_n'}{E \vdash (\forall\, \mathcal{V})\ \sigma(t_1, \ldots, t_n) = \sigma(t_1', \ldots, t_n')} \quad \sigma \in \Sigma_{s_1 \ldots s_n, s}$$

Letting $\vdash_\Sigma \subseteq \mathcal{P}(\mathsf{Eqn}(\Sigma)) \times \mathsf{Eqn}(\Sigma)$ be given by: $E \vdash_\Sigma e$ if and only if $E \vdash e$ can be inferred using a finite number of applications of the above rules, for $E \subseteq \mathsf{Eqn}(\Sigma)$, $e \in \mathsf{Eqn}(\Sigma)$ and $\Sigma \in |\mathsf{Sign}|$

yields an entailment system (see Definition 2.2.7) $(\mathsf{Sign}, \mathsf{Eqn}, \vdash)$. Moreover, the induced notion of entailment is both sound and complete for the satisfaction of equations by algebras.

Theorem 2.3.32 ([GM81]). *Let Σ denote a many-sorted signature, let E denote a set of Σ-equations, and let e denote a Σ-equation. Then, $E \models_\Sigma e$ if and only if $E \vdash_\Sigma e$.*

The *Lawvere category* of a many-sorted signature is a category whose arrows correspond to algebraic terms over that signature, and has the property that algebras of the signature correspond to product-preserving functors from the Lawvere category to Set, while algebra homomorphisms correspond to natural transformations between such functors.

Definition 2.3.33. *Let Σ denote a many-sorted signature with sort set S. The* **Lawvere category** *of Σ is the least category L^Σ satisfying:*

1. *L^Σ has finite products*
2. *$S \subseteq |\mathsf{L}^\Sigma|$*
3. *$\Sigma_{s_1 \ldots s_n, s} \subseteq \mathsf{L}^\Sigma(s_1 \times \ldots \times s_n, s)$ for $n \in \mathbb{N}$ and $s_1, \ldots, s_n, s \in S$.*

Proposition 2.3.34. *Let Σ denote a many-sorted signature. There exists a one-to-one correspondence between s-sorted Σ-terms $t \in T_\Sigma(\{X_1, \ldots, X_n\})_s$ with $X_i : s_i$ for $i = 1, \ldots, n$ and L^Σ-arrows $[\![t]\!] : s_1 \times \ldots \times s_n \to s$. Furthermore, if $t_i \in T_\Sigma(\mathcal{V})_{s_i}$ for $i = 1, \ldots, n$, then $[\![t(t_1/X_1, \ldots, t_n/X_n)]\!] = [\![t]\!] \circ \langle [\![t_1]\!], \ldots, [\![t_n]\!] \rangle$.*

Proof (sketch). The L^Σ-arrow $[\![t]\!] : s_1 \times \ldots \times s_n \to s$ associated to a Σ-term $t \in T_\Sigma(\{X_1, \ldots, X_n\})_s$ with $X_i : s_i$ for $i = 1, \ldots, n$ is defined inductively by:

1. $[\![X_i]\!] = \pi_i : s_1 \times \ldots \times s_n \to s_i$ for $i \in \{1, \ldots, n\}$
2. $[\![\sigma]\!] = \sigma : 1 \to s$ for $\sigma \in \Sigma_{\epsilon, s}$ and $s \in S$ (with 1 denoting the empty product in L^Σ)
3. $[\![\sigma(t_1, \ldots, t_k)]\!] = [\![\sigma]\!] \circ \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle : s_1 \times \ldots \times s_n \to s$ for $\sigma \in \Sigma_{s'_1 \ldots s'_k, s}$, $s'_1, \ldots, s'_k, s \in S$ and $t_j \in T_\Sigma(\{X_1, \ldots, X_n\})_{s'_j}$ for $j = 1, \ldots, k$.

□

Proposition 2.3.35. *Let Σ denote a many-sorted signature. Then, Σ-algebras A are in one-to-one correspondence with product-preserving functors $[\![A]\!] : \mathsf{L}^\Sigma \to \mathsf{Set}$, while Σ-homomorphisms $f : A \to B$ are in one-to-one correspondence with natural transformations $[\![f]\!] : [\![A]\!] \Rightarrow [\![B]\!]$.*

Proof (sketch). For a Σ-algebra A, $[\![A]\!]$ is given by:

1. $[\![A]\!](s) = A_s$ for $s \in S$
2. $[\![A]\!](\sigma) = \sigma_A : [\![A]\!](s_1 \times \ldots \times s_n) \to [\![A]\!](s)$ for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$

□

Corollary 2.3.36. *Let Σ denote a many-sorted signature, and let A denote a Σ-algebra. Then, $A \models_\Sigma (\forall \mathcal{V})\ l = r$ if and only if $[\![A]\!]([\![l]\!]) = [\![A]\!]([\![r]\!])$.*

Proof (sketch). The fact that $[\![A]\!]([\![t]\!]) = t_A$ for any $t \in T_\Sigma(\mathcal{V})$ is used. □

2.3.2 Algebras of Endofunctors

The notion of algebra of a many-sorted signature is an instance of that of an *algebra of an endofunctor*.

Definition 2.3.37. *Let* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ *denote an arbitrary endofunctor. An* F-**algebra** *is a pair $\langle A, \alpha \rangle$ with A a* C*-object and $\alpha : \mathsf{F}A \to A$ a* C*-arrow. A is called the* **carrier** *of the algebra. An* F-**algebra homomorphism** *between* F*-algebras $\langle A, \alpha \rangle$ and $\langle B, \beta \rangle$ is a* C*-arrow $f : A \to B$ such that $\beta \circ \mathsf{F}f = f \circ \alpha$.*

$$\begin{array}{ccc} \mathsf{F}A & \xrightarrow{\alpha} & A \\ {\scriptstyle \mathsf{F}f}\downarrow & & \downarrow{\scriptstyle f} \\ \mathsf{F}B & \xrightarrow[\beta]{} & B \end{array}$$

For an endofunctor F, the category of F-algebras and F-algebra homomorphisms is denoted Alg(F).

Remark 2.3.38. If Σ denotes a many-sorted signature with sort set S, then $\mathsf{Alg}(\Sigma)$ is isomorphic to $\mathsf{Alg}(\mathsf{F}_\Sigma)$, with $\mathsf{F}_\Sigma : \mathsf{Set}^S \to \mathsf{Set}^S$ being given by:

$$(\mathsf{F}_\Sigma X)_s = \coprod_{\sigma \in \Sigma_{s_1 \ldots s_n, s}} (X_{s_1} \times \ldots \times X_{s_n})\ , \quad s \in S$$

Example 2.3.39. Natural numbers (see Example 2.3.2) are specified by taking C to be Set and $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ to be the functor $X \mapsto 1 + X$ (with 1 denoting a final object in Set). Then, F-algebras are given by functions $\alpha : 1 + A \to A$, with $A \in |\mathsf{Set}|$. (The function $\alpha \circ \iota_1 : 1 \to A$ provides the interpretation of the constant symbol 0, while the function $\alpha \circ \iota_2 : A \to A$ provides the interpretation of the unary operation symbol succ.)

If $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ denotes an arbitrary endofunctor, and if $\mathsf{U} : \mathsf{Alg}(\mathsf{F}) \to \mathsf{C}$ denotes the functor taking F-algebras to their carrier, then a desirable property of U is the *creation of coequalisers of congruences*.

Definition 2.3.40 ([Poi92]). *A functor* $\mathsf{U} : \mathsf{D} \to \mathsf{C}$ **creates coequalisers of congruences** *if, whenever $f, g : A \to B$ in* D *and $h : \mathsf{U}B \to C$ in* C *are such that $\mathsf{U}f, \mathsf{U}g : \mathsf{U}A \to \mathsf{U}B$ define a kernel pair in* C *and h defines a coequaliser for $\mathsf{U}f, \mathsf{U}g$ in* C*, there exists a unique* D*-arrow $k : B \to D$ in* D *such that $\mathsf{U}k = h$, and moreover, k is a coequaliser for f, g in* D*. If, in addition,* U *creates limits and has a left adjoint, then* U *is called* **algebraic**.

Example 2.3.41. If Σ denotes a many-sorted signature, the functor taking Σ-algebras to their carrier is algebraic.

The next two definitions give generalisations of the notions of subalgebra, homomorphic image and variety (Definitions 2.3.25 and 2.3.27) to arbitrary endofunctors $\mathsf{F} : \mathsf{C} \to \mathsf{C}$.

Definition 2.3.42. *Let $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ denote an arbitrary endofunctor, and let $\langle A, \alpha \rangle$ denote an* F*-algebra. An* F-**subalgebra** *of $\langle A, \alpha \rangle$ is an object $\langle \langle B, \beta \rangle, b \rangle$ of the slice category $\mathsf{Alg}(\mathsf{F})/\langle A, \alpha \rangle$, such that the* C*-arrow underlying b is a monomorphism. Also, a* **homomorphic image** *of $\langle A, \alpha \rangle$ is an object $\langle \langle B, \beta \rangle, c \rangle$ of the category $\mathsf{Alg}(\mathsf{F})\backslash\langle A, \alpha \rangle$, such that the* C*-arrow underlying c is an epimorphism.*

Definition 2.3.43. *Let $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ denote an arbitrary endofunctor. A full subcategory of $\mathsf{Alg}(\mathsf{F})$ is an* F-**variety** *if and only if it is closed under subalgebras, homomorphic images and arbitrary products.*

The following result, obtained by strengthening the hypotheses of a result in [SP82], gives sufficient conditions for the existence of initial algebras of endofunctors.

Theorem 2.3.44. *Let* C *denote a category with initial object and colimits of ω-chains, and let $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ denote an ω-cocontinuous endofunctor. Then, $\mathsf{Alg}(\mathsf{F})$ has an initial object.*

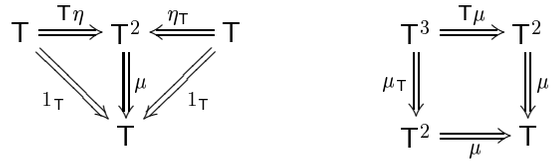
Proof (sketch). If 0 denotes an initial C-object and $! : 0 \to \mathsf{F}0$ denotes its unique C-arrow into $\mathsf{F}0$, then an initial object in $\mathsf{Alg}(\mathsf{F})$ is obtained by suitably endowing the colimit object of the following ω-chain:

$$0 \xrightarrow{!} \mathsf{F}0 \xrightarrow{\mathsf{F}!} \mathsf{F}^2 0 \xrightarrow{\mathsf{F}^2!} \cdots$$

with an F-algebra structure. □

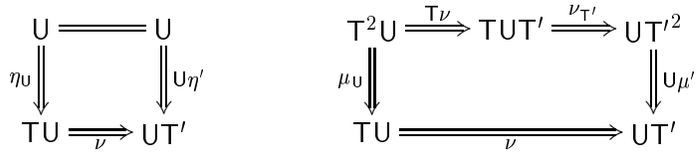
2.3.3 Algebras of Monads

Definition 2.3.45. *Let* C *denote an arbitrary category. A* **monad** *(on* C*) is a tuple $\mathsf{T} = (\mathsf{T}, \eta, \mu)$, with $\mathsf{T} : \mathsf{C} \to \mathsf{C}$, $\eta : \mathsf{Id}_{\mathsf{C}} \Rightarrow \mathsf{T}$ and $\mu : \mathsf{T}^2 \Rightarrow \mathsf{T}$, such that the diagrams:*



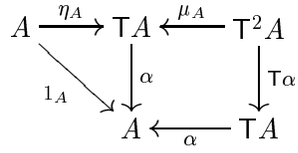
commute. (η is called the **unit***, while μ is called the* **multiplication** *of the monad.)*

Let (T, η, μ) and $(\mathsf{T}', \eta', \mu')$ denote monads on C *and respectively* C′*. A* **monad morphism** *from (T, η, μ) to $(\mathsf{T}', \eta', \mu')$ is a pair (U, ν), with $\mathsf{U} : \mathsf{C}' \to \mathsf{C}$ and $\nu : \mathsf{TU} \Rightarrow \mathsf{UT}'$, such that the diagrams:*

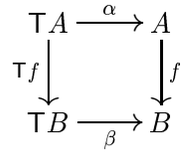


commute.

Definition 2.3.46. *Let* T *denote a monad on* C. *A* T-**algebra** *is a pair* $\langle A, \alpha \rangle$ *with* $A \in |\mathsf{C}|$ *and* $(\alpha : \mathsf{T}A \to A) \in \|\mathsf{C}\|$, *such that the diagram:*



commutes. A T-**algebra homomorphism** *between* T*-algebras* $\langle A, \alpha \rangle$ *and* $\langle B, \beta \rangle$ *is a* C*-arrow* $f : A \to B$ *such that* $\beta \circ \mathsf{T}f = f \circ \alpha$.



For a monad T, the category of T-algebras and T-algebra homomorphisms is denoted Alg(T). Then, monad morphisms $(\mathsf{U}, \nu) : (\mathsf{T}, \eta, \mu) \to (\mathsf{T}', \eta', \mu')$ induce functors $\mathsf{U}_{\nu'} : \mathsf{Alg}(\mathsf{T}') \to \mathsf{Alg}(\mathsf{T})$, with (U, ν) taking a T′-algebra $\langle A, \alpha \rangle$ to the T-algebra $\langle \mathsf{U}A, \mathsf{U}\alpha \circ \nu_A \rangle$. (The conditions in the definition of monad morphisms guarantee that $\langle \mathsf{U}A, \mathsf{U}\alpha \circ \nu_A \rangle$ defines a T-algebra whenever $\langle A, \alpha \rangle$ defines a T′-algebra.)

Remark 2.3.47. For a monad (T, η, μ) on C, the functor $\mathsf{U} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{C}$ taking T-algebras to their carrier has a left adjoint F. (F maps $C \in |\mathsf{C}|$ to $\langle \mathsf{T}C, \mu_C \rangle \in |\mathsf{Alg}(\mathsf{T})|$.) In particular, this results in the existence of an initial T-algebra, given by $\langle \mathsf{T}0, \mu_0 \rangle$, whenever C has an initial object 0.

The following result is typically used to infer the existence of limits and colimits in categories of algebras of monads.

Proposition 2.3.48 ([Bor94b]). *Let* (T, η, μ) *denote a monad on* C. *Then,* $\mathsf{U} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{C}$ *creates limits. Also, if* T *preserves a certain kind of colimits, then* U *creates those colimits.*

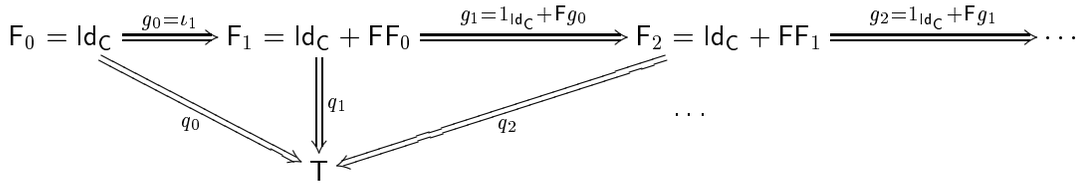
Proof (sketch). Limits (respectively colimits) in Alg(T) are computed as limits (colimits) in C endowed with T-algebra structure. Naturality of η and μ ensure that the resulting structures satisfy the constraints in the definition of algebras of monads. □

Corollary 2.3.49. *Let* (T, η, μ) *denote a monad on* C. *If* C *has a certain kind of limits, then so does* Alg(T), *and* $\mathsf{U} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{C}$ *preserves them. Also, if* C *has, and* T *preserves a certain kind of colimits, then those colimits exist in* Alg(T), *and are preserved by* U.

Endofunctors $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ with C and F subject to additional constraints induce monads (T, η, μ) on C, in such a way that the category of algebras of the induced monad is isomorphic to the category of algebras of the original endofunctor.

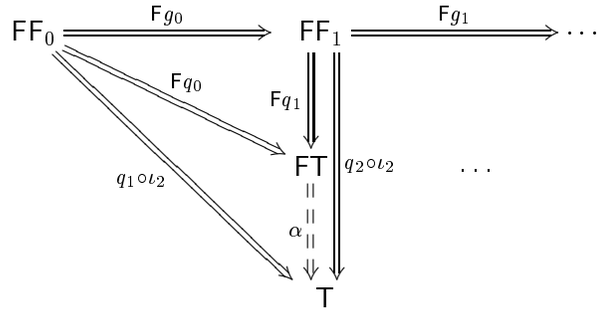
Proposition 2.3.50. *Let* C *denote a category with finite coproducts and colimits of* ω*-chains, and let* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ *denote an endofunctor which preserves colimits of* ω*-chains. Then,* Alg(F) *is isomorphic to* Alg(T) *for some monad* (T, η, μ) *on* C.

Proof. Let $\mathsf{T} : \mathsf{C} \to \mathsf{C}$ denote the endofunctor obtained as a colimit object (in the quasi-category $[\mathsf{C},\mathsf{C}]$) of the ω-chain defined by $g_i : \mathsf{F}_i \Rightarrow \mathsf{F}_{i+1}$ with $i = 0, 1, \ldots$, where $\mathsf{F}_0, \mathsf{F}_1, \ldots : \mathsf{C} \to \mathsf{C}$ are given by: $\mathsf{F}_0 = \mathsf{Id}_\mathsf{C}$, $\mathsf{F}_{i+1} = \mathsf{Id}_\mathsf{C} + \mathsf{F}\mathsf{F}_i$ for $i = 0, 1, \ldots$, while $g_0, g_1, \ldots$ are given by $g_0 = \iota_1$, $g_{i+1} = 1_{\mathsf{Id}_\mathsf{C}} + \mathsf{F}g_i$ for $i = 0, 1, \ldots$. (The existence in $[\mathsf{C},\mathsf{C}]$ of colimits of ω-chains follows by Proposition 2.1.41[3].)

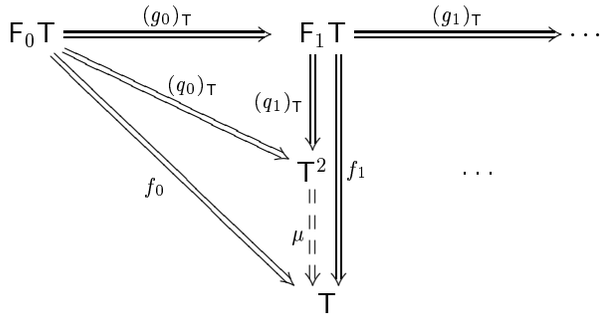


Also, let $q_i : \mathsf{F}_i \Rightarrow \mathsf{T}$ with $i = 0, 1, \ldots$ denote the colimit arrows, and let $\eta : \mathsf{Id}_\mathsf{C} \Rightarrow \mathsf{T}$ be given by q_0.

Note that ω-cocompleteness of F yields a unique natural transformation $\alpha : \mathsf{F}\mathsf{T} \Rightarrow \mathsf{T}$ satisfying $q_{i+1} \circ \iota_2 = \alpha \circ \mathsf{F}q_i$ for $i = 0, 1, \ldots$.



Now let $\mu : \mathsf{T}^2 \Rightarrow \mathsf{T}$ denote the unique natural transformation satisfying: $\mu \circ (q_i)_\mathsf{T} = f_i$ for $i = 0, 1, \ldots$, where the natural transformations $f_i : \mathsf{F}_i\mathsf{T} \Rightarrow \mathsf{T}$ with $i = 0, 1, \ldots$ are given by $f_0 = 1_\mathsf{T}$, $f_{i+1} = [1_\mathsf{T}, \alpha \circ \mathsf{F}f_i]$ for $i = 0, 1, \ldots$.



(The fact that $f_0, f_1, \ldots$ define a cocone on $(g_0)_\mathsf{T}, (g_1)_\mathsf{T}, \ldots$ follows by induction.)

Then, (T, η, μ) defines a monad on C. The fact that $\mu \circ \eta_\mathsf{T} = 1_\mathsf{T}$ follows immediately from the definition of μ. Also, because of the colimiting property of T, proving that $\mu \circ \mathsf{T}\eta = 1_\mathsf{T}$ amounts to

[3] The generalised version of this result (see Remark 2.1.42) is needed if C is not small.

proving that $\mu \circ \mathsf{T}\eta \circ q_i = q_i$ for $i = 0, 1, \ldots$, which follows from:

$$\begin{aligned}
\mu \circ \mathsf{T}\eta \circ q_i = & \quad \text{(definition of } \eta\text{)} \\
\mu \circ \mathsf{T}q_0 \circ q_i = & \quad \text{(naturality of } q_i\text{)} \\
\mu \circ (q_i)_{\mathsf{T}} \circ \mathsf{F}_i q_0 = & \quad \text{(definition of } \mu\text{)} \\
f_i \circ \mathsf{F}_i q_0 = & \quad \text{(*)} \\
q_i &
\end{aligned}$$

(The last of the above equalities follows by induction of i. On the one hand, $f_0 \circ \mathsf{F}_0 q_0 = q_0$ holds. Also, assuming that $f_i \circ \mathsf{F}_i q_0 = q_i$ holds, the fact that $f_{i+1} \circ \mathsf{F}_{i+1} q_0 = q_{i+1}$ also holds follows from:

$$\begin{aligned}
f_{i+1} \circ \mathsf{F}_{i+1} q_0 = & \quad \text{(definitions of } f_{i+1}\text{, } \mathsf{F}_{i+1}\text{)} \\
[1_{\mathsf{T}}, \alpha \circ \mathsf{F}f_i] \circ (q_0 + \mathsf{F}\mathsf{F}_i q_0) = & \quad \text{(property of coproducts)} \\
[q_0, \alpha \circ \mathsf{F}f_i \circ \mathsf{F}\mathsf{F}_i q_0] = & \quad \text{(induction hypothesis)} \\
[q_0, \alpha \circ \mathsf{F}q_i] = & \quad \text{(definition of } \alpha\text{)} \\
[q_0, q_{i+1} \circ \iota_2] = & \quad \text{(definition of T)} \\
[q_{i+1} \circ \iota_1, q_{i+1} \circ \iota_2] = & \quad \text{(property of coproducts)} \\
q_{i+1} &
\end{aligned}$$

for $i = 0, 1, \ldots$.) Finally, because of the colimiting property of T, proving that $\mu \circ \mu_{\mathsf{T}} = \mu \circ \mathsf{T}\mu$ amounts to proving that $\mu \circ \mu_{\mathsf{T}} \circ (q_i)_{\mathsf{T}^2} = \mu \circ \mathsf{T}\mu \circ (q_i)_{\mathsf{T}^2}$ for $i = 0, 1, \ldots$, which follows from:

$$\begin{aligned}
\mu \circ \mu_{\mathsf{T}} \circ (q_i)_{\mathsf{T}^2} = & \quad \text{(definition of } \mu_{\mathsf{T}}\text{)} \\
\mu \circ (f_i)_{\mathsf{T}} = & \quad \text{(*)} \\
f_i \circ \mathsf{F}_i \mu = & \quad \text{(definition of } \mu\text{)} \\
\mu \circ (q_i)_{\mathsf{T}} \circ \mathsf{F}_i \mu = & \quad \text{(naturality of } q_i\text{)} \\
\mu \circ \mathsf{T}\mu \circ (q_i)_{\mathsf{T}^2} &
\end{aligned}$$

(The second of the above equalities uses the fact that $\mu \circ \alpha_{\mathsf{T}} = \alpha \circ \mathsf{F}\mu$. This follows from:

$$\begin{aligned}
\mu \circ \alpha_{\mathsf{T}} \circ \mathsf{F}(q_i)_{\mathsf{T}} = & \quad \text{(definition of } \alpha\text{)} \\
\mu \circ (q_{i+1})_{\mathsf{T}} \circ (\iota_2)_{\mathsf{T}} = & \quad \text{(definition of } \mu\text{)} \\
f_{i+1} \circ (\iota_2)_{\mathsf{T}} = & \quad \text{(definition of } f_{i+1}\text{)} \\
\alpha \circ \mathsf{F}f_i = & \quad \text{(definition of } \mu\text{)} \\
\alpha \circ \mathsf{F}\mu \circ \mathsf{F}(q_i)_{\mathsf{T}} &
\end{aligned}$$

using the colimiting property of T together with the ω-cocontinuity of F. Then, the fact that $\mu \circ (f_i)_{\mathsf{T}} = f_i \circ \mathsf{F}_i \mu$ follows by induction on i. On the one hand, $\mu \circ (f_0)_{\mathsf{T}} = f_0 \circ \mathsf{F}_0 \mu$ holds. Also, assuming that $\mu \circ (f_i)_{\mathsf{T}} = \mu = f_i \circ \mathsf{F}_i \mu$ holds, the fact that $\mu \circ (f_{i+1})_{\mathsf{T}} = f_{i+1} \circ \mathsf{F}_{i+1} \mu$ also holds

follows from:

$$\begin{aligned}
\mu \circ (f_{i+1})_{\mathsf{T}} = &\quad \text{(definition of } f_{i+1}\text{)} \\
\mu \circ [1_{\mathsf{T}^2}, \alpha_{\mathsf{T}} \circ \mathsf{F}(f_i)_{\mathsf{T}}] = &\quad \text{(property of coproducts)} \\
[\mu, \mu \circ \alpha_{\mathsf{T}} \circ \mathsf{F}(f_i)_{\mathsf{T}}] = &\quad (\mu \circ \alpha_{\mathsf{T}} = \alpha \circ \mathsf{F}\mu) \\
[\mu, \alpha \circ \mathsf{F}\mu \circ \mathsf{F}(f_i)_{\mathsf{T}}] = &\quad \text{(induction hypothesis)} \\
[\mu, \alpha \circ \mathsf{F}f_i \circ \mathsf{F}\mathsf{F}_i\mu] = &\quad \text{(property of coproducts)} \\
[1_{\mathsf{T}}, \alpha \circ \mathsf{F}f_i] \circ (\mu + \mathsf{F}\mathsf{F}_i\mu) = &\quad \text{(definition of } f_{i+1}, \mathsf{F}_{i+1}\text{)} \\
f_{i+1} \circ \mathsf{F}_{i+1}\mu &
\end{aligned}$$

for $i = 0, 1, \ldots$)

The monad T thus defined has the property that T-algebras are in one-to-one correspondence with F-algebras. For, given an F-algebra $\langle A, \gamma \rangle$, the unique C-arrow $\delta : \mathsf{T}A \to A$ induced by the cocone $(A, \gamma_0, \gamma_1, \ldots)$ with $\gamma_0 = 1_A$ and $\gamma_{i+1} = [1_A, \gamma \circ \mathsf{F}\gamma_i]$ for $i = 0, 1, \ldots$ on $(g_0)_A, (g_1)_A, \ldots$ defines a T-algebra structure on A. (The fact that $\gamma_0, \gamma_1, \ldots$ define a cocone on $(g_0)_A, (g_1)_A, \ldots$ follows by induction.) Also, given a T-algebra $\langle A, \delta \rangle$, the C-arrow $\delta \circ (q_1)_A \circ \iota_2 : \mathsf{F}A \to A$ defines an F-algebra structure on A. Moreover, the two mappings are functorial, and are inverse to each other. □

Remark 2.3.51. By Proposition 2.1.41, T preserves any small colimits which exist in C and are preserved by F. Also, if F preserves epimorphisms, then so does T.

If (T, η, μ) and $(\mathsf{T}', \eta', \mu')$ denote the monads induced by the endofunctors $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ and $\mathsf{F}' : \mathsf{C}' \to \mathsf{C}'$, then pairs (U, ξ) with $\mathsf{U} : \mathsf{C}' \to \mathsf{C}$ and $\xi : \mathsf{F}\mathsf{U} \Rightarrow \mathsf{U}\mathsf{F}'$ subject to additional constraints can be used to define monad morphisms $(\mathsf{U}, \nu) : (\mathsf{T}, \eta, \mu) \to (\mathsf{T}', \eta', \mu')$.

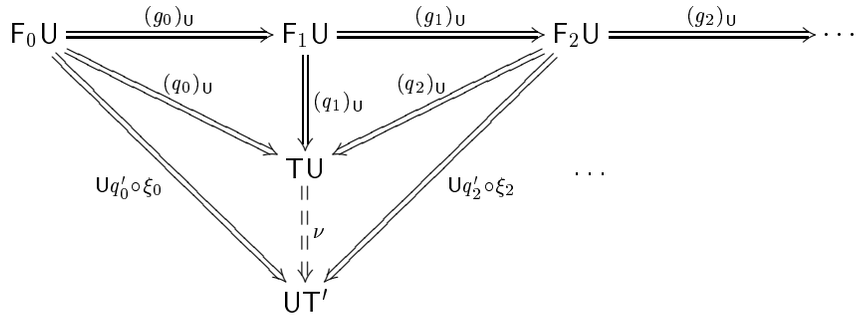
Proposition 2.3.52. *Let* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ *and* $\mathsf{F}' : \mathsf{C}' \to \mathsf{C}'$ *denote endofunctors satisfying the hypotheses of Proposition 2.3.50. Also, let* (T, η, μ) *and* $(\mathsf{T}', \eta', \mu')$ *denote the monads induced by* F *and respectively* F′*, and let* $\mathsf{V} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{Alg}(\mathsf{F})$ *and* $\mathsf{V}' : \mathsf{Alg}(\mathsf{T}') \to \mathsf{Alg}(\mathsf{F}')$ *denote the functors taking* $\langle A, \alpha \rangle \in |\mathsf{Alg}(\mathsf{T})|$ *to* $\langle A, \alpha \circ (q_1)_A \circ \iota_2 \rangle \in |\mathsf{Alg}(\mathsf{F})|$*, and respectively* $\langle A', \alpha' \rangle \in |\mathsf{Alg}(\mathsf{T}')|$ *to* $\langle A', \alpha' \circ (q_1')_{A'} \circ \iota_2' \rangle \in |\mathsf{Alg}(\mathsf{F}')|$*. Finally, let* $\mathsf{U} : \mathsf{C}' \to \mathsf{C}$ *denote an arbitrary functor, let* $\xi : \mathsf{F}\mathsf{U} \Rightarrow \mathsf{U}\mathsf{F}'$ *denote a natural transformation, and let* $\mathsf{U}_\xi : \mathsf{Alg}(\mathsf{F}') \to \mathsf{Alg}(\mathsf{F})$ *denote the functor taking* $\langle A', \alpha' \rangle \in |\mathsf{Alg}(\mathsf{F}')|$ *to* $\langle \mathsf{U}A', \mathsf{U}\alpha' \circ \xi_{A'} \rangle \in |\mathsf{Alg}(\mathsf{F})|$*. Then, the following diagram commutes:*

$$\begin{array}{ccc}
\mathsf{Alg}(\mathsf{T}') & \xrightarrow{\mathsf{U}_\nu} & \mathsf{Alg}(\mathsf{T}) \\
{\scriptstyle \mathsf{V}'}\downarrow & & \downarrow{\scriptstyle \mathsf{V}} \\
\mathsf{Alg}(\mathsf{F}') & \xrightarrow[\mathsf{U}_\xi]{} & \mathsf{Alg}(\mathsf{F})
\end{array}$$

for some monad morphism $(\mathsf{U}, \nu) : (\mathsf{T}, \eta, \mu) \to (\mathsf{T}', \eta', \mu')$*.*

Proof. Let $\nu : \mathsf{T}\mathsf{U} \Rightarrow \mathsf{U}\mathsf{T}'$ denote the natural transformation arising from the observation that $\mathsf{T}\mathsf{U}$ is a colimit object for the ω-chain defined by $(g_i)_{\mathsf{U}} : \mathsf{F}_i\mathsf{U} \Rightarrow \mathsf{F}_{i+1}\mathsf{U}$ for $i = 0, 1, \ldots$ (with colimiting

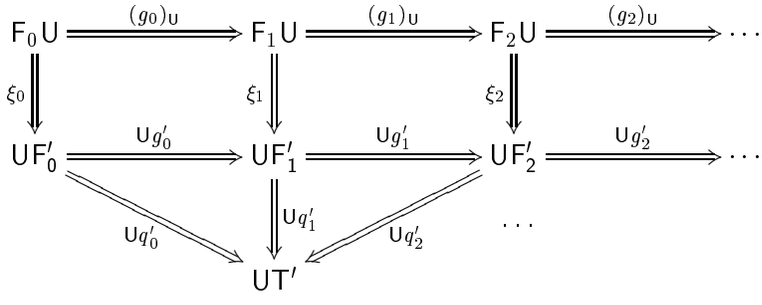
arrows $(q_i)_{\mathsf{U}} : \mathsf{F}_i\mathsf{U} \Rightarrow \mathsf{T}\mathsf{U}$ for $i = 0, 1, \ldots$), whereas $\mathsf{U}\mathsf{T}'$ together with $\mathsf{U}q'_i \circ \xi_i$ for $i = 0, 1, \ldots$ define a cocone on this ω-chain:



with $\xi_i : \mathsf{F}_i\mathsf{U} \Rightarrow \mathsf{U}\mathsf{F}'_i$ being given by: $\xi_0 = 1_{\mathsf{U}}$, $\xi_{i+1} = [\mathsf{U}\iota'_1, \mathsf{U}\iota'_2 \circ \xi_{\mathsf{F}'_i} \circ \mathsf{F}\xi_i]$ for $i = 0, 1, \ldots$:

$$\begin{array}{ccc} \mathsf{F}_{i+1}\mathsf{U} = \mathsf{U} + \mathsf{F}\mathsf{F}_i\mathsf{U} & \xRightarrow{1_{\mathsf{U}} + \mathsf{F}\xi_i} & \mathsf{U} + \mathsf{F}\mathsf{U}\mathsf{F}'_i \\ \| & & \Big\Downarrow{\scriptstyle 1_{\mathsf{U}} + \xi_{\mathsf{F}'_i}} \\ \mathsf{U}\mathsf{F}'_{i+1} = \mathsf{U}(\mathsf{Id}_{\mathsf{C}'} + \mathsf{F}'\mathsf{F}'_i) & \xLeftarrow[[\mathsf{U}\iota'_1, \mathsf{U}\iota'_2]]{} & \mathsf{U} + \mathsf{U}\mathsf{F}'\mathsf{F}'_i \end{array}$$

For, induction on i can be used to show that the upper-half of the following diagram commutes[4]:



whereas the definition of T' ensures that its lower-half commutes, thus making $\mathsf{U}\mathsf{T}'$ the object of a cocone on the ω-chain defining $\mathsf{T}\mathsf{U}$.

The resulting natural transformation ν therefore satisfies $\nu \circ (q_i)_{\mathsf{U}} = \mathsf{U}q'_i \circ \xi_i$ for $i = 0, 1, \ldots$. In particular, $\nu \circ (q_1)_{\mathsf{U}} = \mathsf{U}q'_1 \circ \xi_1$, which, in turn, yields $\mathsf{U}_\xi\mathsf{V}' = \mathsf{V}\mathsf{U}_\nu$. For, the following holds:

$$\begin{array}{rl} \mathsf{U}_\xi\mathsf{V}'\langle A', \alpha'\rangle = & \text{(definition of } \mathsf{V}') \\ \mathsf{U}_\xi\langle A', \alpha' \circ (q'_1)_{A'} \circ \iota'_2\rangle = & \text{(definition of } \mathsf{U}_\xi) \\ \langle \mathsf{U}A', \mathsf{U}\alpha' \circ \mathsf{U}(q'_1)_{A'} \circ \mathsf{U}\iota'_2 \circ \xi_{A'}\rangle = & \text{(definition of } \xi_1) \\ \langle \mathsf{U}A', \mathsf{U}\alpha' \circ \mathsf{U}(q'_1)_{A'} \circ (\xi_1)_{A'} \circ \iota_2\rangle = & (\mathsf{U}q'_1 \circ \xi_1 = \nu \circ (q_1)_{\mathsf{U}}) \\ \langle \mathsf{U}A', \mathsf{U}\alpha' \circ \nu_{A'} \circ (q_1)_{\mathsf{U}A'} \circ \iota_2\rangle = & \text{(definition of } \mathsf{V}) \\ \mathsf{V}\langle \mathsf{U}A', \mathsf{U}\alpha' \circ \nu_{A'}\rangle = & \text{(definition of } \mathsf{U}_\nu) \\ \mathsf{V}\mathsf{U}_\nu\langle A', \alpha'\rangle & \end{array}$$

[4] This observation will also be used in Section 3.3.6.

for any T'-algebra $\langle A', \alpha' \rangle$.

Furthermore, the natural transformation ν satisfies the conditions in the definition of monad morphisms. The fact that $\nu \circ \eta_{\mathsf{U}} = \mathsf{U}\eta'$ follows immediately from the definition of ν (which gives $\nu \circ (q_0)_{\mathsf{U}} = \mathsf{U}q'_0 \circ \xi_0$), while the fact that $\nu \circ \mu_{\mathsf{U}} = \mathsf{U}\mu' \circ \nu_{\mathsf{T}'} \circ \mathsf{T}\nu$ follows from:

$$\begin{aligned}
\nu \circ \mu_{\mathsf{U}} \circ (q_i)_{\mathsf{TU}} &= \quad \text{(definition of } \mu\text{)} \\
\nu \circ (f_i)_{\mathsf{U}} &= \quad \text{(*)} \\
\mathsf{U}f'_i \circ (\xi_i)_{\mathsf{T}'} \circ \mathsf{F}_i\nu &= \quad \text{(definition of } \mu'\text{)} \\
\mathsf{U}\mu' \circ \mathsf{U}(q'_i)_{\mathsf{T}'} \circ (\xi_i)_{\mathsf{T}'} \circ \mathsf{F}_i\nu &= \quad \text{(definition of } \nu'\text{)} \\
\mathsf{U}\mu' \circ \nu_{\mathsf{T}'} \circ (q_i)_{\mathsf{UT}'} \circ \mathsf{F}_i\nu &= \quad \text{(naturality of } q_i\text{)} \\
\mathsf{U}\mu' \circ \nu_{\mathsf{T}'} \circ \mathsf{T}\nu \circ (q_i)_{\mathsf{TU}}
\end{aligned}$$

for $i = 0, 1, \ldots$, using the colimiting property of T. (The second of the above equalities uses the fact that $\nu \circ \alpha_{\mathsf{U}} = \mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\nu$. This follows from:

$$\begin{aligned}
\nu \circ \alpha_{\mathsf{U}} \circ \mathsf{F}(q_i)_{\mathsf{U}} &= \quad \text{(definition of } \alpha_{\mathsf{U}}\text{)} \\
\nu \circ (q_{i+1})_{\mathsf{U}} \circ \iota_2 &= \quad \text{(definition of } \nu\text{)} \\
\mathsf{U}q'_{i+1} \circ \xi_{i+1} \circ \iota_2 &= \quad \text{(definition of } \xi_{i+1}\text{)} \\
\mathsf{U}q'_{i+1} \circ \mathsf{U}\iota'_2 \circ \xi_{\mathsf{F}'_i} \circ \mathsf{F}\xi_i &= \quad \text{(definition of } \alpha'\text{)} \\
\mathsf{U}\alpha' \circ \mathsf{U}\mathsf{F}'q'_i \circ \xi_{\mathsf{F}'_i} \circ \mathsf{F}\xi_i &= \quad \text{(naturality of } \xi\text{)} \\
\mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\mathsf{U}q'_i \circ \mathsf{F}\xi_i &= \quad \text{(definition of } \nu\text{)} \\
\mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\nu \circ \mathsf{F}(q_i)_{\mathsf{U}}
\end{aligned}$$

for $i = 0, 1, \ldots$, using the colimiting property of FTU[5]. Then, the fact that $\nu \circ (f_i)_{\mathsf{U}} = \mathsf{U}f'_i \circ (\xi_i)_{\mathsf{T}'} \circ \mathsf{F}_i\nu$ follows by induction on i. On the one hand, $\nu \circ (f_0)_{\mathsf{U}} = \mathsf{U}f'_0 \circ (\xi_0)_{\mathsf{T}'} \circ \mathsf{F}_0\nu$ holds. Also, assuming that $\nu \circ (f_i)_{\mathsf{U}} = \mathsf{U}f'_i \circ (\xi_i)_{\mathsf{T}'} \circ \mathsf{F}_i\nu$ holds, the fact that $\nu \circ (f_{i+1})_{\mathsf{U}} = \mathsf{U}f'_{i+1} \circ (\xi_{i+1})_{\mathsf{T}'} \circ \mathsf{F}_{i+1}\nu$ also holds follows from:

$$\begin{aligned}
\nu \circ (f_{i+1})_{\mathsf{U}} &= \quad \text{(definition of } f_{i+1}\text{)} \\
\nu \circ [1_{\mathsf{TU}}, \alpha_{\mathsf{U}} \circ \mathsf{F}(f_i)_{\mathsf{U}}] &= \quad (\nu \circ \alpha_{\mathsf{U}} = \mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\nu) \\
[\nu, \mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\nu \circ \mathsf{F}(f_i)_{\mathsf{U}}] &= \quad \text{(induction hypothesis)} \\
[\nu, \mathsf{U}\alpha' \circ \xi_{\mathsf{T}'} \circ \mathsf{F}\mathsf{U}f'_i \circ \mathsf{F}(\xi_i)_{\mathsf{T}'} \circ \mathsf{F}\mathsf{F}_i\nu] &= \quad \text{(naturality of } \xi\text{)} \\
[\nu, \mathsf{U}\alpha' \circ \mathsf{U}\mathsf{F}'f'_i \circ \xi_{\mathsf{F}'_i\mathsf{T}'} \circ \mathsf{F}(\xi_i)_{\mathsf{T}'} \circ \mathsf{F}\mathsf{F}_i\nu] &= \quad \text{(definitions of } f'_{i+1},\ \xi_{i+1},\ \mathsf{F}_{i+1}\text{)} \\
\mathsf{U}f'_{i+1} \circ (\xi_{i+1})_{\mathsf{T}'} \circ \mathsf{F}_{i+1}\nu
\end{aligned}$$

for $i = 0, 1, \ldots$.) $\square$

Remark 2.3.53. If the endofunctors F and F' preserve epimorphisms, and if the functor $\mathsf{U} : \mathsf{C}' \to \mathsf{C}$ preserves finite coproducts and colimits of ω-chains, then whenever ξ is a natural epimorphism (respectively isomorphism), so is ν. Moreover, in either case, the C-arrow underlying the unique T-algebra homomorphism from the initial T-algebra to the T-reduct of the initial T'-algebra is an

[5] The ω-cocontinuity of F is used here.

epimorphism. For, if ξ is a natural epimorphism (isomorphism), then so is each of the ξ_is. (The preservation of epimorphisms by F and the preservation of binary coproducts by U are used here.) Then, if ξ (and hence each of the ξ_is) is a natural epimorphism, the fact that ν is a natural epimorphism follows from the observation that $\nu \circ (q_i)_{\mathsf{U}} = \mathsf{U}q'_i \circ \xi_i$ for $i = 0, 1, \ldots$, together with the colimiting property of UT' (following from the colimiting property of T' together with the preservation of colimits of ω-chains by U):

$$\begin{aligned} f \circ \nu_X = g \circ \nu_X \;&\Rightarrow \\ f \circ \nu_X \circ (q_i)_{\mathsf{U}X} = g \circ \nu_X \circ (q_i)_{\mathsf{U}X},\ i = 0, 1, \ldots \;&\Leftrightarrow \\ f \circ \mathsf{U}(q'_i)_X \circ (\xi_i)_X = g \circ \mathsf{U}(q'_i)_X \circ (\xi_i)_X,\ i = 0, 1, \ldots \;&\Leftrightarrow \\ f \circ \mathsf{U}(q'_i)_X = g \circ \mathsf{U}(q'_i)_X,\ i = 0, 1, \ldots \;&\Leftrightarrow \\ f = g \end{aligned}$$

Also, if ξ (and hence each of the ξ_is) is a natural isomorphism, the fact that ν is a natural isomorphism follows from the observation that, in this case, UT' is itself a colimit object for the ω-chain defining TU. In either case, the C-arrow underlying the unique T-algebra homomorphism from the initial T-algebra $\langle \mathsf{T}0, \mu_0 \rangle$ to the T-reduct $\langle \mathsf{UT}'0', \nu_{\mathsf{T}'0'} \circ \mathsf{U}\mu'_{0'} \rangle$ of the initial T'-algebra $\langle \mathsf{T}'0', \mu'_{0'} \rangle$ is given by $\nu_{0'} \circ \mathsf{T}i$, with 0 and $0'$ denoting initial C- and respectively C′-objects, and with $i : 0 \to \mathsf{U}0'$ denoting the unique C-arrow resulting from the initiality of 0. Moreover, $\nu_{0'}$ is an epimorphism (see above), while i is an isomorphism (as U preserves initial objects). Hence, $\nu_{0'} \circ \mathsf{T}i$ is an epimorphism.

2.4 Coalgebraic Specification

The categorical duals of F-algebras (with $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ an arbitrary endofunctor) are called F-*coalgebras* and are typically used to specify structures that involve observation (as opposed to construction).

Definition 2.4.1. *Let* $\mathsf{G} : \mathsf{C} \to \mathsf{C}$ *denote an arbitrary endofunctor. A* G-**coalgebra** *is a pair* $\langle C, \gamma \rangle$ *with* C *a* C*-object and* $\gamma : C \to \mathsf{G}C$ *a* C*-arrow.* C *is called the* **carrier** *of the coalgebra. A* G-**coalgebra homomorphism** *between* G*-coalgebras* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$ *is a* C*-arrow* $g : C \to D$ *additionally satisfying:* $\delta \circ g = \mathsf{G}g \circ \gamma$*:*

$$\begin{array}{ccc} C & \xrightarrow{\gamma} & \mathsf{G}C \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle \mathsf{G}g} \\ D & \xrightarrow[\delta]{} & \mathsf{G}D \end{array}$$

The category of G-coalgebras and G-coalgebra homomorphisms is denoted Coalg(G).

Example 2.4.2. A coalgebraic specification of infinite lists of natural numbers is obtained by taking C to be Set and $\mathsf{G} : \mathsf{Set} \to \mathsf{Set}$ to be the functor $X \mapsto \mathbb{N} \times X$ (with $\mathbb{N}$ denoting the set of natural numbers). Then, G-coalgebras are given by functions $\gamma : C \to \mathbb{N} \times C$, with $C \in |\mathsf{Set}|$. The functions $\mathtt{head}_\gamma = \pi_1 \circ \gamma : C \to \mathbb{N}$ and $\mathtt{tail}_\gamma = \pi_2 \circ \gamma : C \to C$ give the head and respectively the tail of an infinite list. Moreover, the G-coalgebra $\langle C, \gamma \rangle$ is fully determined by these two functions.

Example 2.4.3. A coalgebraic specification of finite and infinite lists of natural numbers is obtained by taking C to be Set and G : Set $\to$ Set to be the functor $X \mapsto 1 + (\mathbb{N} \times X)$. Then, G-coalgebras are given by functions $\gamma : C \to 1 + (\mathbb{N} \times C)$, associating, to each $c \in C$, either the unique element $*$ of 1, if c denotes an empty list, or a pair $\langle n, c' \rangle \in \mathbb{N} \times C$ (with n and c' giving the head and respectively the tail of c), if c denotes a non-empty list.

Dualising the definitions of subalgebra, homomorphic image and variety (Definitions 2.3.42 and 2.3.43) yields coalgebraic notions of *homomorphic image*, *subcoalgebra* and *covariety*.

Of particular interest amongst the coalgebras of an endofunctor is the final one. The following result gives sufficient conditions for the existence of final coalgebras.

Theorem 2.4.4. *Let* C *denote a category with final object and limits of* ω^{op}*-chains, and let* G : C $\to$ C *denote an* ω^{op}*-continuous endofunctor. Then,* Coalg(G) *has a final object.*

Proof (sketch). The carrier F of the final G-coalgebra is constructed as a limit object of the following ω^{op}-chain:

$$1 \xleftarrow{!} \mathsf{G}1 \xleftarrow{\mathsf{G}!} \mathsf{G}^2 1 \xleftarrow{\mathsf{G}^2!} \cdots$$

with ! : G1 $\to$ 1 denoting the unique arrow from G1 to the final C-object 1. □

Example 2.4.5. Since Set is complete, and since polynomial endofunctors are ω^{op}-continuous (see Example 2.1.37), the above theorem yields final coalgebras for the specifications of infinite lists in Example 2.4.2 and respectively of finite and infinite lists in Example 2.4.3. The carriers of the final coalgebras are given by: $F = \mathbb{N} \times \mathbb{N} \times \ldots \simeq \mathbb{N}^{\mathbb{N}}$ and respectively $F' = 1 + (\mathbb{N} \times (1 + \mathbb{N} \times (\ldots))) \simeq (\mathbb{N})^* \cup \mathbb{N}^{\mathbb{N}}$ (with $(\mathbb{N})^*$ denoting the set of finite sequences of natural numbers), while their coalgebraic structures are given by: $\zeta : F \to \mathbb{N} \times F$, $\zeta((e_n)_{n\in\mathbb{N}}) = \langle e_0, (e_{n+1})_{n\in\mathbb{N}} \rangle$ and respectively $\zeta' : F' \to 1 + (\mathbb{N} \times F')$, $\zeta'(\epsilon) = \iota_1(*)$ (with ϵ denoting the empty sequence of natural numbers), $\zeta'(e : l) = \iota_2(e, l)$.

Final coalgebras satisfy a *coinductive definition principle*, which states the existence of a coalgebra homomorphism from any other coalgebra, and which can be used to define C-arrows into the carriers of final coalgebras.

Example 2.4.6. If $\langle F, \zeta \rangle$ denotes a final coalgebra of the specification of infinite lists in Example 2.4.2, then a merging operation `merge` : $F \times F \to F$ on infinite lists can be defined coinductively as the underlying map of the (unique) coalgebra homomorphism from a suitably-chosen coalgebraic structure φ on $F \times F$ to the final coalgebra. Specifically, $\varphi : F \times F \to \mathbb{N} \times (F \times F)$ is taken to be given by:

$$\varphi(\langle l_1, l_2 \rangle) = \langle \pi_1(\zeta(l_1)), \langle l_2, \pi_2(\zeta(l_1)) \rangle \rangle$$

for $\langle l_1, l_2 \rangle \in F \times F$. Similarly, if $\langle F', \zeta' \rangle$ denotes a final coalgebra of the specification of finite and infinite lists in Example 2.4.3, then an empty list `empty` $\in F'$ can be defined coinductively as the image of the unique element of the one-element set 1 under the (unique) coalgebra homomorphism from $\langle 1, \iota_1 \rangle$ (with $\iota_1 : 1 \to 1 + (\mathbb{N} \times 1)$) to $\langle F', \zeta' \rangle$. Finally, a merging operation `merge` : $F' \times F' \to F'$ on finite and infinite lists can also be defined coinductively, namely as the underlying map of the

(unique) coalgebra homomorphism from the coalgebra $\langle F' \times F', \varphi' \rangle$, with $\varphi' : (F' \times F') \to 1 + (\mathbb{N} \times (F' \times F'))$ being given by:

$$\varphi'(\langle l_1, l_2 \rangle) = \begin{cases} \iota_1(*) & \text{if } \zeta'(l_1) \in \iota_1(1) \text{ and } \zeta'(l_2) \in \iota_1(1) \\ \varphi'(\langle l_2, l_1 \rangle) & \text{if } \zeta'(l_1) \in \iota_1(1) \text{ and } \zeta'(l_2) \in \iota_2(\mathbb{N} \times F') \\ \iota_2(\langle n_1, \langle l_2, t_1 \rangle \rangle) & \text{if } \zeta'(l_1) = \iota_2(\langle n_1, t_1 \rangle) \end{cases}$$

for $\langle l_1, l_2 \rangle \in F' \times F'$, to the final coalgebra.

The following result can be used to infer the existence of limits in Coalg(G).

Proposition 2.4.7. *Let* G : C → C *denote an arbitrary endofunctor. If* G *preserves a certain kind of limits, then the functor* U : Coalg(G) → C *taking* G*-coalgebras to their carrier creates those limits.*

Corollary 2.4.8. *Let* G : C → C *denote an arbitrary endofunctor. If* C *has, and* G *preserves a certain kind of limits, then those limits exist in* Coalg(G)*, and are preserved by* U*.*

Also, the existence of colimits in Coalg(G) follows from the existence of colimits in C.

Proposition 2.4.9 ([Bar93]). *Let* G : C → C *denote an arbitrary endofunctor. Then,* U : Coalg(G) → C *creates colimits.*

Corollary 2.4.10. *Let* G : C → C *denote an arbitrary endofunctor. If* C *has a certain kind of colimits, then so does* Coalg(G)*, and* U *preserves those colimits.*

Limits and colimits in Coalg(G) are computed as limits and colimits in C endowed with G-coalgebra structure.

The notion of *bisimulation* also plays an important rôle in the study of coalgebras.

Definition 2.4.11. *Let* C *denote an arbitrary category, and let* G : C → C*. Also, let* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$ *denote* G*-coalgebras. A relation* $\langle R, r_1, r_2 \rangle$ *on* C, D *defines a* G-**bisimulation** *between* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$ *if and only if there exists a* G*-coalgebra structure* $\langle R, \rho \rangle$ *on* R *such that* r_1, r_2 *define* G*-coalgebra homomorphisms from* $\langle R, \rho \rangle$ *to* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$ *respectively. The largest*[6] G*-bisimulation between* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$*, if it exists, is called* G-**bisimilarity**.

Remark 2.4.12. If the endofunctor G preserves pullbacks, and if a final G-coalgebra exists, then largest G-bisimulations on G-coalgebras exist and are given by (the C-arrows underlying) the kernel pairs of the unique G-coalgebra homomorphisms into the final G-coalgebra. In this case, final coalgebras satisfy a *coinductive proof principle*, which states that any bisimulation on a final coalgebra is contained (via $\leq$) in the equality relation on the carrier of that coalgebra. Consequently, proving equality in a final coalgebra can be reduced to proving equality up to some bisimulation on that coalgebra. This proof principle will be used in the forthcoming chapters to derive techniques for proving properties up to bisimulation.

[6] w.r.t. $\leq$, see Remark 2.1.12.

Example 2.4.13. The notion of bisimilarity associated to the specification of infinite lists in Example 2.4.2 (respectively to the specification of finite and infinite lists in Example 2.4.3) relates two elements c and d of the coalgebras $\langle C, \gamma\rangle$ and respectively $\langle D, \delta\rangle$ if and only if they denote lists with (the same number of elements and) the same elements.

Example 2.4.14. If $\langle F', \zeta'\rangle$, $\mathtt{empty} \in F'$ and $\mathtt{merge} : F' \times F' \to F'$ are as in Example 2.4.6, then proving that $\mathtt{merge(empty,empty)} = \mathtt{empty}$ by coinduction amounts to exhibiting a bisimulation $\langle R, r_1, r_2\rangle$ on $\langle F', \zeta'\rangle$ such that $\mathtt{merge(empty,empty)}\ R\ \mathtt{empty}$. For instance, $\langle R, r_1, r_2\rangle$ can be taken to be the least bisimulation satisfying the above condition, i.e. the relation given by $\{(\mathtt{merge(empty,empty)}, \mathtt{empty})\}$. The fact that this defines a bisimulation on $\langle F', \zeta'\rangle$ follows from the definitions of $\mathtt{empty}$ and $\mathtt{merge}$ as coalgebra homomorphisms. For, these definitions yield:

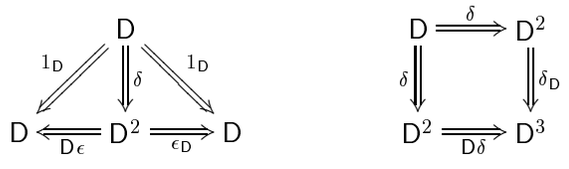
$$\begin{aligned} \zeta'(\mathtt{merge}(\mathtt{empty}, \mathtt{empty})) = &\quad \text{(definition of } \mathtt{merge}\text{)} \\ (1_1 + (1_{\mathbb{N}} \times \mathtt{merge}))(\varphi'(\langle \mathtt{empty}, \mathtt{empty}\rangle)) = &\quad \text{(definitions of } \mathtt{empty} \text{ and } \varphi'\text{)} \\ (1_1 + (1_{\mathbb{N}} \times \mathtt{merge}))(\iota_1(*)) = & \\ \iota_1(*) & \end{aligned}$$

and:

$$\begin{aligned} \zeta'(\mathtt{empty}) = &\quad \text{(definition of } \mathtt{empty}\text{)} \\ \iota_1(*) & \end{aligned}$$

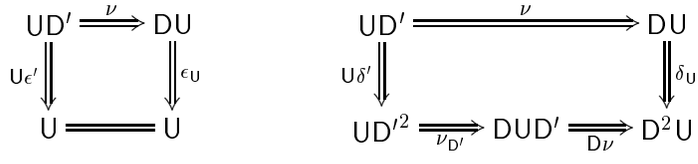
We conclude this section by noting that notions of *comonad* and *coalgebra of a comonad* are defined similarly to those of monad and algebra of a monad.

Definition 2.4.15. *Let* C *denote an arbitrary category. A* **comonad** *(on* C*) is a tuple* $\mathsf{D} = (\mathsf{D}, \epsilon, \delta)$*, with* $\mathsf{D} : \mathsf{C} \to \mathsf{C}$*,* $\epsilon : \mathsf{D} \Rightarrow \mathsf{Id}_\mathsf{C}$ *and* $\delta : \mathsf{D} \Rightarrow \mathsf{D}^2$*, such that the diagrams:*



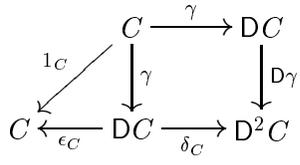
commute. (ϵ *is called the* **counit***, while* δ *is called the* **comultiplication** *of the comonad.)*

Let $(\mathsf{D}, \epsilon, \delta)$ *and* $(\mathsf{D}', \epsilon', \delta')$ *denote comonads on* C *and respectively* C′*. A* **comonad morphism** *from* $(\mathsf{D}, \epsilon, \delta)$ *to* $(\mathsf{D}', \epsilon', \delta')$ *is a pair* (U, ν)*, with* $\mathsf{U} : \mathsf{C}' \to \mathsf{C}$ *and* $\nu : \mathsf{UD}' \Rightarrow \mathsf{DU}$*, such that the diagrams:*

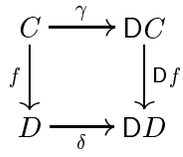


commute.

Definition 2.4.16. *Let* D *denote a comonad on* C*. A* D-**coalgebra** *is a pair* $\langle C, \gamma \rangle$ *with* $C \in |\mathsf{C}|$ *and* $(\gamma : C \to \mathsf{D}C) \in \|\mathsf{C}\|$*, such that the diagram:*



commutes. A D-**coalgebra homomorphism** *between* D*-coalgebras* $\langle C, \gamma \rangle$ *and* $\langle D, \delta \rangle$ *is a* C*-arrow* $f : C \to D$ *such that* $\mathsf{D}f \circ \gamma = \delta \circ f$*.*



3 Equational Specification in an Abstract Setting

This chapter presents an abstract equational framework for the specification of structures having both an observational and a computational component. The framework is obtained by clearly distinguishing between observational and computational features and using coalgebra and respectively algebra for their formalisation, and by taking a layered approach to their integration. In addition, the framework uses abstract equational sentences to formalise properties which concern either the observational or the computational features, with both categories of features being used in defining the associated notions of satisfaction.

First, a coalgebraic framework for the specification of observational structures is introduced. This framework unifies some of the existing equational approaches to the specification of state-based systems, including [Rei95, HR95, Jac96c, Jac96a, GM97, Cor98], by capturing at an abstract level the concepts typically employed by such approaches. The resulting framework involves notions of *abstract cosignature*, used to specify particular kinds of observational structures, *coalgebra of an abstract cosignature*, used to provide a particular interpretation for the structure specified by the cosignature, *observer over a cosignature*, used to extract information from the coalgebras of the cosignature according to their particular interpretation for the specified structure, and *coequation over a cosignature*, used to constrain the coalgebras of the cosignature by requiring different observers to yield similar (either equal or just observationally equal) results on the same coalgebra. In addition, a notion of (*horizontal*) *cosignature morphism* is used to specify a change in the type of information being observed, and this is shown to yield an institution w.r.t. the satisfaction (up to observability) of coequations by coalgebras. Canonical ways of combining related specifications and their implementations are shown to exist, while final and cofree coalgebras are shown to provide suitable denotations for coalgebraic specifications and their morphisms.

An algebraic framework for the specification of structures involving computation is then derived essentially by dualising the previously-obtained coalgebraic framework. The dualisation results in notions of *abstract signature*, *algebra of an abstract signature*, *constructor* and *equation*, with the many-sorted algebraic notions of signature, algebra, term and equation being instances of the corresponding abstract notions.

Finally, the two frameworks are integrated in order to account for the relationship between computations and observations in structures having both a computational and an observational component. Such an integration builds on the work in [Tur96] (see also [TP97]) on well-behaved operational semantics, and amounts to lifting the coalgebraic structure of semantic domains to computations over these semantic domains, in order to interpret computations on the semantic domains induced by the observational component. The resulting interpretations are well-behaved, in that the notion of bisimulation induced by the observational component is preserved by computations, whereas the

notion of reachability induced by the computational component is preserved by observations. A dual approach, which involves interpreting observations on the syntactic domains induced by the computational component is also obtained. In each case, equations and coequations are used to formalise correctness properties of the resulting interpretations, with such properties only being required to hold up to bisimulation and respectively up to reachability. This gives rise to institutions. Also in each case, both initial and final models exist, with the quotients of the unique homomorphisms from the initial to the final models providing suitable denotations for the resulting specifications. Two compositionality results, allowing specifications and respectively implementations to be combined in a canonical way are also formulated for the combined framework.

The chapter is structured as follows. Section 3.1 introduces the equational coalgebraic framework for the specification of structures involving observation, Section 3.2 derives a similar framework for the specification of structures involving computation, while Section 3.3 integrates the two frameworks.

3.1 Specification of Observational Structures

[Rut96] presents a general coalgebraic framework for the specification of state-based systems, with arbitrary endofunctors on Set being used to specify system behaviour, and with coalgebras of such endofunctors providing (abstractions of) particular implementations of the specified behaviours. The approach in [Rut96] is here specialised in order to give a categorical account of *equational* coalgebraic approaches to specification[1]. A framework which unifies some of the existing equational approaches to system specification (including [HR95, Jac96c, GM97, Cor98]) is introduced, the existence of suitable denotations for the specification techniques supported by this framework is investigated, and the expressiveness of the resulting approach is briefly discussed.

3.1.1 Cosignatures, Coalgebras, Finality and Bisimulation

We begin by introducing an abstract syntax for specifying observational structures.

Definition 3.1.1. *An* **(abstract) cosignature** *is a pair* (C, F)*, with* C *a category which is complete, cocomplete*[2] *and regular, and with* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ *an endofunctor which preserves pullbacks and limits of* ω^{op}*-chains.*

Remark 3.1.2. Powerset functors (of form $\mathcal{P}(L \times \_) : \mathrm{Set} \to \mathrm{Set}$, taking a set X to the set of subsets of $L \times X$, or of form $\mathcal{P}(\_)^L : \mathrm{Set} \to \mathrm{Set}$, taking a set X to the set of mappings from L to the set of subsets of X) are not ω^{op}-continuous, and therefore do not give rise to abstract cosignatures. A generalisation of the coalgebraic framework presented here which also accounts for powerset functors *of bounded cardinality* (with only the subsets of cardinality smaller than some fixed cardinal being considered in this case) constitutes the subject of future work. The approach to such a generalisation is briefly outlined in Section 6.2.

[1] Our setting is, however, more abstract than the one in [Rut96], as endofunctors on arbitrary categories are considered here.

[2] It would be sufficient to require that C has finite limits, limits of ω^{op}-chains, quotients and coproducts.

Remark 3.1.3. A framework for the specification of structures involving computation will be derived in Section 3.2 essentially by dualising the coalgebraic specification framework described in this section. However, since both in the coalgebraic case and in the algebraic case the semantic constructions of interest involve quotients, the condition requiring the regularity of the underlying categories of abstract cosignatures will be carried over unchanged to the algebraic framework, while the condition requiring the preservation of pullbacks by the endofunctors defining abstract cosignatures will be replaced by a condition which guarantees the creation of quotients by the functors taking algebras of abstract signatures to their carrier[3]. All the forthcoming definitions and results which do not depend on/make use of the preservation of pullbacks by the endofunctors defining abstract cosignatures will be marked *, in order to indicate that a dual version of the definition or result in question exists in the algebraic framework.

Abstract cosignatures specify the type of information that can be observed about particular systems. The coalgebras of the endofunctors in question then provide (abstractions of) specific system implementations.

Definition* 3.1.4. *Let* (C, F) *denote an abstract cosignature. A* (C, F)-**coalgebra** *(respectively* (C, F)-**coalgebra homomorphism***) is an* F*-coalgebra (*F*-coalgebra homomorphism).*

For an abstract cosignature (C, F), the category of (C, F)-coalgebras and (C, F)-coalgebra homomorphisms is denoted $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$, while the functor taking (C, F)-coalgebras to their carrier is denoted $\mathsf{U}_\mathsf{C} : \mathsf{Coalg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$.

Remark 3.1.5. An abstract cosignature (C, F) induces a comonad $(\mathsf{D}, \epsilon, \delta)$ on C having the property that $\mathsf{Coalg}(\mathsf{D}) \simeq \mathsf{Coalg}(\mathsf{C}, \mathsf{F})$. This is a consequence of the existence in C of finite products and of limits of ω^{op}-chains, together with the preservation by F of limits of ω^{op}-chains. The construction of this comonad is dual to the construction of the monad induced by an endofunctor satisfying dual restrictions (see Proposition 2.3.50). In particular, the endofunctor $\mathsf{D} : \mathsf{C} \to \mathsf{C}$ is obtained as a limit in (the quasi-category) $[\mathsf{C}, \mathsf{C}]$ of the following ω^{op}-chain:

$$\mathsf{Id}_\mathsf{C} \xleftarrow{\pi_1} \mathsf{Id}_\mathsf{C} \times \mathsf{F} \xleftarrow{1_{\mathsf{Id}_\mathsf{C}} \times \mathsf{F}\pi_1} \mathsf{Id}_\mathsf{C} \times \mathsf{F}(\mathsf{Id}_\mathsf{C} \times \mathsf{F}) \xleftarrow{1_{\mathsf{Id}_\mathsf{C}} \times \mathsf{F}(1_{\mathsf{Id}_\mathsf{C}} \times \mathsf{F}\pi_1)} \cdots$$

The next three examples are the first in a series of examples aimed to illustrate that the abstract coalgebraic specification framework introduced here unifies some of the existing equational formalisms for the specification of state-based, dynamical systems.

Example 3.1.6 (**Destructor Hidden Signatures***).* Let V denote a set of *visible sorts*, let Ψ denote a V-sorted signature (the *data signature*), and let D denote a Ψ-algebra all of whose elements are named by Ψ-terms[4] (the *data algebra*). A *hidden signature (over* Ψ*)* [GM97] is a pair (H, Σ), with H a set of *hidden sorts* (disjoint from V) and Σ an $S = V \cup H$-sorted signature, additionally satisfying:

- $\Sigma_{w,v} = \Psi_{w,v}$ for $w \in V^*$ and $v \in V$

[3] Note that the preservation of pullbacks by the endofunctors defining abstract cosignatures also ensures the creation of quotients by the functors taking coalgebras of abstract cosignatures to their carrier (see Proposition 2.4.7 and 2.4.9).

[4] That is, for any $v \in V$ and $d \in D_v$, there exists $t \in T_{\Psi,v}$ satisfying $t_D = d$.

– if $\sigma \in \Sigma_{w,s}$, then w contains at most one hidden sort.

That is, apart from Ψ-operation symbols, hidden signatures may only contain *generalised hidden constants* (i.e. operation symbols $\sigma : w \to h$ with $w \in V^*$ and $h \in H$), *methods* (i.e. operation symbols $\sigma : w \to h$ with w containing precisely one hidden sort and $h \in H$) and *attributes* (i.e. operation symbols $\sigma : w \to v$ with w containing precisely one hidden sort and $v \in V$). Given a hidden signature Σ, a *hidden Σ-algebra* (respectively *hidden Σ-homomorphism*) is a many-sorted Σ-algebra A (many-sorted Σ-homomorphism f) such that $A{\restriction_\Psi} = D$ ($f{\restriction_V} = 1_D$). The category of hidden Σ-algebras and hidden Σ-homomorphisms is denoted $\mathsf{Alg}(\Sigma)$. Finally, a *destructor hidden signature* [Cîr98] is a hidden signature Σ additionally satisfying: $\Sigma_{w,h} = \emptyset$ for any $w \in V^*$ and $h \in H$. That is, destructor hidden signatures contain no generalised hidden constants. As a result, destructor hidden signatures (H, Σ) induce abstract cosignatures $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$, with $\mathsf{F}_\Sigma : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ being given by:

$$(\mathsf{F}_\Sigma X)_s = \begin{cases} \prod_{\sigma \in \Sigma_{sw,s'}} X_{s'}^{D_w} & \text{if } s \in H \\ D_v & \text{if } s \in V \end{cases}$$

for $X \in |\mathsf{Set}_D^S|$ and $s \in S$. For, the category Set_D^S is complete, cocomplete and regular (see Examples 2.1.31 and 2.1.47), while F_Σ preserves pullbacks and limits of ω^{op}-chains (see Example 2.1.37). Moreover, $\mathsf{Alg}(\Sigma) \simeq \mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$ (see [Cîr98]).

To illustrate the relationship between the hidden algebra formalism and the abstract coalgebraic framework introduced here, we consider a hidden signature specifying cells holding natural numbers. (The example is taken from [GM00].) This signature consists of a visible sort `Nat`, a hidden sort `State`, a hidden constant `init` : $\to$ `State`, an attribute `getx` : `State` $\to$ `Nat` and a method `putx` : `Nat State` $\to$ `State`. The destructor hidden subsignature of this hidden signature, obtained by leaving out the hidden constant `init`, induces an abstract cosignature $(\mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}, \mathsf{F})$, with $\mathsf{F} : \mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}} \to \mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}$ taking $(\mathbb{N}, X) \in |\mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}|$ to $(\mathbb{N}, \mathbb{N} \times X^{\mathbb{N}}) \in |\mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}|$.

Example 3.1.7. Coalgebras of endofunctors of form $\mathsf{G} : \mathsf{Set} \to \mathsf{Set}$, $\mathsf{G}X = \prod_{i=1,\ldots,n} (B_i + C_i \times X)^{A_i}$ are used in [Jac96c, Jac96a, Jac97] to specify object interfaces. Such endofunctors preserve pullbacks and limits of ω^{op}-chains (see Example 2.1.37), and therefore induce abstract cosignatures.

Example 3.1.8 (**Co-signatures***).* A *co-signature* [Cor98] is a triple $\Pi = \langle S, OP, [\![\_]\!] \rangle$, where S (the *sorts*), OP (the *operators*) and $[\![\_]\!]$ (the *interpretation of visible sorts*) are as follows:

– S is a triple $\langle X, \{I_1, \ldots, I_k\}, \{O_1, \ldots, O_h\} \rangle$, where X is the *hidden sort*, I_j is an *input sort* for $j = 1, \ldots, k$, and O_j is an *output sort* for $j = 1, \ldots, h$. The sets of input and output sorts need not be disjoint. Their elements are called *visible sorts*.

– OP is a pair $OP = \langle \{m_1, \ldots, m_l\}, \{a_1, \ldots, a_n\} \rangle$, where $m_j : X \times I_{k_j} \to X$ is a *method* for $j = 1, \ldots, l$, and $a_j : X \times I_{k_j} \to O_{h_j}$ is an *attribute* for $j = 1, \ldots, n$.

– $[\![\_]\!]$ is a function mapping each visible sort to a non-empty set. Moreover, if V is a visible sort, then each $v \in [\![V]\!]$ is denoted by a *constant* $\underline{v} : V$.

Given a co-signature Π, a Π-*coalgebra* [Cor98] consists of a set X_A (the *carrier*), a function $m_{j_A} : X_A \times [\![I_{k_j}]\!] \to X_A$ for each $j \in \{1, \ldots, l\}$, and a function $a_{j_A} : X_A \times [\![I_{k_j}]\!] \to [\![O_{h_j}]\!]$ for each $j \in \{1, \ldots, n\}$. Given two Π-coalgebras A and B, a Π-*coalgebra homomorphism* is a function $f : X_A \to X_B$ additionally satisfying:

1. $m_B(f(x), v) = f(m_A(x, v))$ for any $x \in X_A$ and $v \in [\![I]\!]$
2. $a_B(f(x), v) = a_A(x, v)$ for any $x \in X_A$ and $v \in [\![I]\!]$

The category of Π-coalgebras and Π-coalgebra homomorphisms is denoted $\mathsf{Coalg}(\Pi)$. Co-signatures $\Pi = \langle S, OP, [\![\_]\!] \rangle$ induce abstract cosignatures $(\mathsf{Set}, \mathsf{F}_\Pi)$, with $\mathsf{F}_\Pi : \mathsf{Set} \to \mathsf{Set}$ being given by:

$$\mathsf{F}_\Pi X = \prod_{j \in \{1,\ldots,l\}} X^{[\![I_{k_j}]\!]} \times \prod_{j \in 1,\ldots,n\}} [\![O_{h_j}]\!]^{[\![I_{k_j}]\!]}$$

for $X \in |\mathsf{Set}|$. For, Set is complete, cocomplete and regular (see Examples 2.1.31 and 2.1.47), while F_Π preserves pullbacks and limits of ω^{op}-chains (see Example 2.1.37). Moreover, $\mathsf{Coalg}(\Pi) \simeq \mathsf{Coalg}(\mathsf{Set}, \mathsf{F}_\Pi)$ (see [Cor98]).

An immediate consequence of Definition 3.1.1 is the existence of final coalgebras of abstract cosignatures.

Proposition* 3.1.9. *Let* (C, F) *denote an abstract cosignature. Then,* $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ *has a final object.*

Proof. The conclusion follows by Theorem 2.4.4. □

Also, for an abstract cosignature (C, F), the existence of pullbacks in C together with the preservation of pullbacks by F result in the existence of pullbacks in $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$.

Proposition 3.1.10. *Let* (C, F) *denote an abstract cosignature. Then,* U_C *creates pullbacks.*

Proof. The conclusion follows by Proposition 2.4.7. □

Corollary 3.1.11. *For an abstract cosignature* (C, F), $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ *has pullbacks and* U_C *preserves them.*

Proof. The conclusion follows by Corollary 2.4.8. □

Corollary 3.1.12. *Let* (C, F) *denote an abstract cosignature. Then,* U_C *preserves as well as reflects monomorphisms.*

Proof. Let $f : \langle C, \gamma \rangle \to \langle D, \delta \rangle$ denote a (C, F)-coalgebra homomorphism. Then, by 3 of Proposition 2.1.43, f (respectively $\mathsf{U}_\mathsf{C} f$) is a monomorphism if and only if $\langle 1_{\langle C,\gamma \rangle}, 1_{\langle C,\gamma \rangle} \rangle$ ($\langle 1_C, 1_C \rangle$) defines a kernel pair for f ($\mathsf{U}_\mathsf{C} f$). The conclusion then follows from the fact that U_C preserves as well as creates kernel pairs. □

Proposition 2.1.30 together with Proposition 3.1.9 and Corollary 3.1.11 yield the following.

Corollary 3.1.13. *For an abstract cosignature* (C, F)*,* Coalg(C, F) *has finite limits.*

Remark 3.1.14. Since U_C preserves kernel pairs, it follows by 5 of Proposition 2.1.43 that kernel pairs in Coalg(C, F) define (C, F)-bisimulations. And since F preserves pullbacks and Coalg(C, F) has a final object, it follows that largest bisimulations on (C, F)-coalgebras exist and are given by (the C-arrows underlying) the kernel pairs of the unique (C, F)-coalgebra homomorphisms into the final (C, F)-coalgebra (see Remark 2.4.12). Furthermore, the resulting (C, F)-coalgebra structures on the largest bisimulations are the *only* ones which make the two C-arrows defining the largest bisimulations into (C, F)-coalgebra homomorphisms.

A notion of *observability* of coalgebras can also be defined.

Definition* 3.1.15. *Let* (C, F) *denote an abstract cosignature. A* (C, F)*-coalgebra* $\langle C, \gamma \rangle$ *is* **observable** *if and only if the* C*-arrow underlying the unique homomorphism from* $\langle C, \gamma \rangle$ *to the final* (C, F)*-coalgebra is a monomorphism.*

It then follows by Remark 3.1.14 together with 3 of Proposition 2.1.43 that a (C, F)-coalgebra is observable if and only if the largest bisimulation on it is given by the equality relation on its carrier.

Example 3.1.16 (**Final Hidden Algebras, Behavioural Equivalence***).* Let (H, Σ) denote a hidden signature. A Σ*-context of sort* $h \in H$ [GM97] is a visible-sorted Σ-term containing a single occurrence of a variable z of sort h. Then, *behavioural equivalence* [GM97] on a hidden Σ-algebra A is the S-sorted relation $\sim$ on A given by:

- $a_1 \sim_v a_2$ if and only if $a_1 = a_2$, for $v \in V$
- $a_1 \sim_h a_2$ if and only $c_A(a_1) = c_A(a_2)$ for any Σ-context c of sort h, for $h \in H$ (with c_A denoting the function interpreting the Σ-context c as a function on A_h).

It is shown in [GM97] that final algebras exist for destructor hidden signatures – their elements of type h are given by functions assigning, to each context c of sort h and result type v, a value $d \in D_v$. As a result, if $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$ denotes the abstract cosignature induced by a destructor hidden signature Σ (see Example 3.1.6), then behavioural equivalence on a Σ-algebra A is the same as $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$-bisimulation on the corresponding $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$-coalgebra.

Given the hidden signature of cells in Example 3.1.6, the following are contexts of sort `State`: `getx(z)`, `getx(putx(0,z))`, `getx(putx(1,putx(0,z)))`. One algebra of this hidden signature has hidden carrier $\mathbb{N}$, and interprets `init`, `getx` and `putx` as $0 \in \mathbb{N}$, $1_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ and respectively $\pi_1 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Behavioural equivalence on this algebra is given by the equality relation. Another algebra of the hidden signature of cells has hidden carrier $(\mathbb{N})^+$ (finite, non-empty sequences of natural numbers, providing a history of cells), and interprets `init`, `getx` and `putx` as $(0) \in (\mathbb{N})^+$, first : $(\mathbb{N})^+ \to \mathbb{N}$ (taking a list to its first element) and respectively append : $\mathbb{N} \times (\mathbb{N})^+ \to (\mathbb{N})^+$ (appending the natural number given by its first argument to the front of the list given by its

second argument). Behavioural equivalence on this algebra relates any two lists whose first elements coincide. The final algebra of the destructor hidden subsignature of the hidden signature of cells is given by the reduct of the first algebra considered here to this subsignature.

Example 3.1.17. [Jac96c] gives a characterisation of final coalgebras of endofunctors $\mathsf{G} : \mathsf{Set} \to \mathsf{Set}$ of form $\mathsf{G}X = \prod_{i=1,\ldots,n} (B_i + C_i \times X)^{A_i}$: their elements are given by suitably-restricted functions from $(A_1 + \ldots + A_n)^+$ to $(B_1 + \ldots + B_n) + (C_1 + \ldots + C_n)$ (assigning, to each sequence of inputs, a value defining the visible outcome of the experiment defined by these inputs on the given element).

The existence of colimits in C results in the existence of colimits in $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ and in their preservation by U_C.

Proposition* 3.1.18. *Let* (C, F) *denote an abstract cosignature. Then,* $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ *has colimits and* U_C *preserves them.*

Proof. The conclusion follows by Corollary 2.4.10. □

Also, the creation of cokernel pairs by U_C yields a characterisation of epimorphisms in $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$.

Corollary* 3.1.19. *Let* (C, F) *denote an abstract cosignature. Then,* U_C *preserves as well as reflects epimorphisms.*

Proof. Similar to the proof of Corollary 3.1.12. □

Finally, the existence of quotients in C together with the preservation of kernel pairs by F result in the existence of quotients in $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ and in their preservation by U_C.

Proposition 3.1.20. *Let* (C, F) *denote an abstract cosignature. Then,* U_C *creates quotients.*

Proof. U_C creates kernel pairs (see Proposition 3.1.10) and coequalisers (see Proposition 2.4.9). □

Corollary 3.1.21. *For an abstract cosignature* (C, F), $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ *has quotients and* U_C *preserves them.*

The quotient of a (C, F)-coalgebra homomorphism defines a homomorphic image of the domain of this homomorphism (as U_C preserves epis, see Corollary 3.1.19). And because C is regular, this also yields a subcoalgebra of the codomain of the given homomorphism.

Corollary 3.1.22. *Let* (C, F) *denote an abstract cosignature, let* f *denote a* (C, F)*-coalgebra homomorphism with quotient* e*, and let* $f = \iota \circ e$ *denote the factorisation of* f *resulting from the universality of* e*. Then,* e *defines a homomorphic image of the domain of* f*, while* ι *defines a* (C, F)*-subcoalgebra of the codomain of* f*.*

Proof. The preservation of quotients by U_C results in $\mathsf{U}_\mathsf{C} e$ being an epimorphism. Also, the regularity of C results in $\mathsf{U}_\mathsf{C} \iota$ being a monomorphism (see Proposition 2.1.46). □

3.1.2 Observers, Coequations and Satisfaction (up to Bisimulation)

Abstract cosignatures specify basic ways of observing particular systems: an abstract cosignature (C,F) induces a natural transformation $\lambda : \mathsf{U_C} \Rightarrow \mathsf{FU_C}$, with $\lambda_{\langle C,\gamma\rangle} : C \to \mathsf{F}C$ being given by $\lambda_{\langle C,\gamma\rangle} = \gamma$ for $\langle C,\gamma\rangle \in |\mathsf{Coalg}(\mathsf{C},\mathsf{F})|$. ($\lambda$ uses the coalgebraic structure given by γ to extract information of type F from the carrier C of γ.) More complex observations on the carriers of (C,F)-coalgebras can also be defined, e.g. by considering natural transformations of form $\mathsf{F}^{n-1}\lambda\circ\ldots\circ\mathsf{F}\lambda\circ\lambda : \mathsf{U_C} \Rightarrow \mathsf{F}^n\mathsf{U_C}$, with $n \in \mathbb{N}^*$. The next definition formally captures such complex observations, as well as more general ones, by exploiting the fact that the result of an observation depends solely on the coalgebraic structure.

Definition* 3.1.23. *Let* (C,F) *denote an abstract cosignature. A* (C,F)-**observer** *is a pair* (K,c)*, with* $\mathsf{K} : \mathsf{C} \to \mathsf{C}$ *an endofunctor which preserves monomorphisms, and with* $c : \mathsf{U_C} \Rightarrow \mathsf{KU_C}$ *a natural transformation. (*K *is called the* **type** *of the observer.)*

(C,F)-observers are parameterised by (C,F)-coalgebras: a (C,F)-observer (K,c) specifies, for each (C,F)-coalgebra $\langle C,\gamma\rangle$, a C-arrow $c_\gamma : C \to \mathsf{K}C$, extracting information of type K from C using the coalgebraic structure γ. In addition, (C,F)-observers are well-behaved w.r.t. coalgebra homomorphisms, in that if $f : \langle C,\gamma\rangle \to \langle D,\delta\rangle$ is a (C,F)-coalgebra homomorphism, then $c_\delta \circ \mathsf{U_C}f = \mathsf{KU_C}f \circ c_\gamma$.

$$\begin{array}{ccc} C & \xrightarrow{\mathsf{U_C}f} & D \\ {\scriptstyle c_\gamma}\downarrow & & \downarrow{\scriptstyle c_\delta} \\ \mathsf{K}C & \xrightarrow[\mathsf{KU_C}f]{} & \mathsf{K}D \end{array}$$

That is, the extraction of K-information from coalgebras commutes with coalgebra homomorphisms.

Remark 3.1.24. (C,F)-observers can be composed. Specifically, if (K,c) and (K',c') are (C,F)-observers, then so is $(\mathsf{KK}',\mathsf{K}c'\circ c)$. (The preservation of monomorphisms by K and K′ results in their preservation by KK′.)

In particular, if $\lambda : \mathsf{U_C} \Rightarrow \mathsf{FU_C}$ is as before, then $\mathsf{F}^{n-1}\lambda\circ\ldots\circ\mathsf{F}\lambda\circ\lambda : \mathsf{U_C} \Rightarrow \mathsf{F}^n\mathsf{U_C}$ is a (C,F)-observer, for any $n \in \mathbb{N}^*$.

Pairs of observers are used to constrain system implementations, by requiring that the specified observers yield similar results.

Definition* 3.1.25. *Let* (C,F) *denote an abstract cosignature. A* (C,F)-**coequation** *is a tuple* (K,l,r)*, with* (K,l) *and* (K,r) *denoting* (C,F)*-observers. A* (C,F)*-coalgebra* $\langle C,\gamma\rangle$ **satisfies** *a* (C,F)*-coequation* (K,l,r) *(written* $\langle C,\gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$*) if and only if* $l_\gamma = r_\gamma$*.*

For an abstract cosignature (C,F) and a class E of (C,F)-coequations, the full subcategory of $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$ whose objects satisfy the coequations in E is denoted $\mathsf{Coalg}(\mathsf{C},\mathsf{F},E)$.

Example 3.1.26 (**Standard Equational Satisfaction in Hidden Algebra***).* If $(\mathsf{Set}^S_D,\mathsf{F}_\Sigma)$ denotes the abstract cosignature induced by a destructor hidden signature Σ (see Example 3.1.6), then any

conditional Σ-equation e in one hidden-sorted variable induces a $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$-coequation c such that $A \models_\Sigma e$ if and only if $\langle C, \gamma\rangle \models_{(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)} c$, with A a hidden Σ-algebra and $\langle C, \gamma\rangle$ its associated $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$-coalgebra. Specifically, if e is of form $(\forall\, Z)\ l = r$ if $l_1 = r_1, \ldots, l_n = r_n$, then c is of form (K, l', r'), with $\mathsf{K} : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ being given by:

$$(\mathsf{K}X)_h = \begin{cases} 1 + X_s & \text{if } \ Z : h \\ 1 & \text{otherwise} \end{cases}, \quad h \in H$$
$$(\mathsf{K}X)_v = D_v, \quad v \in V$$

where s denotes the type of l and r, and with $l', r' : \mathsf{U} \Rightarrow \mathsf{KU}$ (where $\mathsf{U} : \mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{F}_\Sigma) \to \mathsf{Set}_D^S$ denotes the functor taking $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$-coalgebras to their carrier) being given by:

$$(l'_{\langle C,\gamma\rangle})_h = \begin{cases} l_A^{(l_1,r_1),\ldots,(l_n,r_n)} & \text{if } \ Z : h \\ ! : C_h \to 1 & \text{otherwise} \end{cases}, \quad h \in H$$
$$(l'_{\langle C,\gamma\rangle})_v = 1_{D_v} \quad v \in V$$

for $\langle C, \gamma\rangle \in |\mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)|$ with A its associated hidden Σ-algebra (and similarly for r'), where $l_A^{(l_1,r_1),\ldots,(l_n,r_n)}, r_A^{(l_1,r_1),\ldots,(l_n,r_n)} : A_h \to 1 + A_s$ are given by:

$$l_A^{(l_1,r_1),\ldots,(l_n,r_n)}(a) = \begin{cases} \iota_2(l_A(a)) & \text{if } \ (l_i)_A(a) = (r_i)_A(a) \text{ for } i = 1, \ldots, n \\ \iota_1(*) & \text{otherwise} \end{cases}$$

for $a \in A_h$ (and similarly for $r_A^{(l_1,r_1),\ldots,(l_n,r_n)}$). The fact that K preserves monomorphisms follows by Example 2.1.37.

Similarly, any conditional Σ-equation e in one hidden-sorted variable and a number of visible-sorted variables induces a $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$-coequation c, such that $A \models_\Sigma e$ if and only if $\langle C, \gamma\rangle \models_{(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)} c$. Specifically, if e is of form $(\forall\, Z)(\forall\, X_1)\ldots(\forall\, X_m)\ l = r$ if $l_1 = r_1, \ldots, l_n = r_n$ with $X_i : v_i$ for $i = 1, \ldots, m$, then c is of form (K, l', r'), with $\mathsf{K} : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ being given by:

$$(\mathsf{K}X)_h = \begin{cases} \prod\limits_{D_{v_1}} \cdots \prod\limits_{D_{v_m}} (1 + X_s) & \text{if } \ Z : h \\ 1 & \text{otherwise} \end{cases}, \quad h \in H$$
$$(\mathsf{K}X)_v = D_v, \quad v \in V$$

and with $l', r' : \mathsf{U} \Rightarrow \mathsf{KU}$ being given by:

$$(l'_{\langle C,\gamma\rangle})_h = \begin{cases} l_A^{(l_1,r_1),\ldots,(l_n,r_n)} & \text{if } \ Z : h \\ ! : C_h \to 1 & \text{otherwise} \end{cases}, \quad h \in H$$
$$(l'_{\langle C,\gamma\rangle})_v = 1_{D_v}, \quad v \in V$$

(and similarly for r') where $l_A^{(l_1,r_1),\ldots,(l_n,r_n)}, r_A^{(l_1,r_1),\ldots,(l_n,r_n)} : A_h \to \prod\limits_{D_{v_1}} \cdots \prod\limits_{D_{v_m}} (1 + A_s)$ are given by:

$$(l_A^{(l_1,r_1),\ldots,(l_n,r_n)}(a))_{\bar{d}} = \begin{cases} \iota_2(l_A(a, \bar{d})) & \text{if } \ (l_i)_A(a, \bar{d}) = (r_i)_A(a, \bar{d}) \text{ for } i = 1, \ldots, n \\ \iota_1(*) & \text{otherwise} \end{cases}$$

for $a \in A_h$ and $\bar{d} \in D_{v_1} \times \ldots \times D_{v_m}$ (and similarly for $r_A^{(l_1,r_1),\ldots,(l_n,r_n)}$).

In the case of unconditional equations, the induced coequations have a simpler form, with $(\mathsf{K}X)_h$ being given by X_s and respectively $\prod_{D_{v_1}} \ldots \prod_{D_{v_m}} X_s$ (rather than by $1 + X_s$ and respectively $\prod_{D_{v_1}} \ldots \prod_{D_{v_m}} (1 + X_s)$), and with the definitions of l', r' being adjusted accordingly.

The following equations are used in [GM00] to constrain the behaviour of cells (see Example 3.1.6):

$$(\forall\, \mathtt{N})(\forall\, \mathtt{S})\ \mathtt{getx}(\mathtt{putx}(\mathtt{N},\mathtt{S})) = \mathtt{N}$$
$$(\forall\, \mathtt{S})(\forall\, \mathtt{N})(\forall\, \mathtt{M})\ \mathtt{putx}(\mathtt{N},\mathtt{putx}(\mathtt{M},\mathtt{S})) = \mathtt{putx}(\mathtt{N},\mathtt{S})$$

The $(\mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}, \mathsf{F})$-coequations (K, l, r) and (K', l', r') induced by the above equations are given by:

$$\begin{aligned}(\mathsf{K}X)_{\mathtt{State}} &= \prod_{n\in\mathbb{N}} \mathbb{N}\\ (l_{\langle C,\gamma\rangle})_{\mathtt{State}} &= \langle \mathtt{getx}_A \circ \mathtt{putx}_A(n,\_)\rangle_{n\in\mathbb{N}}\\ (r_{\langle C,\gamma\rangle})_{\mathtt{State}} &= \langle n\rangle_{n\in\mathbb{N}}\end{aligned}$$

(with the constant function taking $a \in A_{\mathtt{State}}$ to $n \in \mathbb{N}$ also being denoted by n) and respectively:

$$\begin{aligned}(\mathsf{K}'X)_{\mathtt{State}} &= \prod_{m\in\mathbb{N}} \prod_{n\in\mathbb{N}} X_{\mathtt{State}}\\ (l'_{\langle C,\gamma\rangle})_{\mathtt{State}} &= \langle \mathtt{putx}_A(n,\_) \circ \mathtt{putx}_A(m,\_)\rangle_{m\in\mathbb{N},n\in\mathbb{N}}\\ (r'_{\langle C,\gamma\rangle})_{\mathtt{State}} &= \langle \mathtt{putx}_A(n,\_)\rangle_{m\in\mathbb{N},n\in\mathbb{N}}\end{aligned}$$

for $\langle C, \gamma\rangle \in |\mathsf{Coalg}(\mathsf{Set}_{\mathbb{N}}^{\{\mathtt{Nat},\mathtt{State}\}}, \mathsf{F})|$ with A its associated hidden algebra.

Example 3.1.27 (**Co-signatures Contd.***).* Let $\Pi = \langle S, OP, [\![\_]\!]\rangle$ denote a co-signature (see Example 3.1.8). A Π-*context of sort* Y [Cor98] is a well-sorted term of sort Y built from the operators of Π and from constants of input sorts, containing precisely one occurrence of a variable, denoted x, which must be of hidden sort. Contexts of hidden sort are called *transition sequences*, while contexts of output sorts are called *observations*. Given a Π-context c together with a Π-coalgebra $A = \langle X_A, (m_{j_A})_{j\in\{1,\ldots,l\}}, (a_{j_A})_{j\in\{1,\ldots,n\}}\rangle$, the *interpretation of* c *in* A is a function $[c]_A$ with domain X_A defined as follows:

- $[x]_A : X_A \to X_A$ is given by 1_{X_A}.
- If $c : X$ is a transition sequence of form $m(c', \underline{v})$ for some transition sequence $c' : X$, some method m and some constant $\underline{v}$ of input sort, then $[c]_A : X_A \to X_A$ is given by $[c]_A(y) = m_A([c']_A(y), v)$ for $y \in X_A$.
- If $c : O$ is an observation of form $a(c', \underline{v})$ for some transition sequence $c' : X$, some attribute a and some constant $\underline{v}$ of input sort, then $[c]_A : X_A \to [\![O]\!]$ is given by $[c]_A(y) = a_A([c']_A(y), v)$ for $y \in X_A$.

A Π-*term* [Cor98] is either an observation or a constant of output sort. The *interpretation* of a Π-term $t : O$ in a Π-coalgebra A is a function $[t]_A : X_A \to [\![O]\!]$ defined by:

$$[t]_A = \begin{cases} [c]_A & \text{if } t \text{ is given by an observation } c \\ v & \text{if } t \text{ is given by a constant } \underline{v} \end{cases}$$

A Π-*equation* [Cor98] is a pair $\langle t_1 : O, t_2 : O \rangle$ of Π-terms of the same sort, and is alternatively denoted $t_1 =_O t_2$. A Π-coalgebra A is said to *satisfy* a Π-equation $t_1 =_O t_2$ if and only if $[t_1]_A = [t_2]_A$ (as functions from X_A to $[\![O]\!]$).

If $(\mathsf{Set}, \mathsf{F}_\Pi)$ denotes the abstract cosignature induced by the co-signature Π (see Example 3.1.8), then Π-contexts c induce $(\mathsf{Set}, \mathsf{F}_\Pi)$-observers (K, o), with $\mathsf{K} : \mathsf{Set} \to \mathsf{Set}$ being given by:

$$\mathsf{K}X = \begin{cases} X & \text{if } c \text{ is a transition sequence} \\ [\![O]\!] & \text{if } c \text{ is an observation of type } O \end{cases}$$

for $X \in |\mathsf{Set}|$, and with $o : \mathsf{U}_{\mathsf{Set}} \Rightarrow \mathsf{KU}_{\mathsf{Set}}$ being given by:

$$o_{\langle C,\gamma \rangle} = [c]_A$$

for $\langle C, \gamma \rangle \in |\mathsf{Coalg}(\mathsf{Set}, \mathsf{F}_\Pi)|$ with A its associated Π-coalgebra. Similarly, Π-terms $t : O$ induce $(\mathsf{Set}, \mathsf{F}_\Pi)$-observers (K, c), with $\mathsf{K} : \mathsf{Set} \to \mathsf{Set}$ being given by:

$$\mathsf{K}X = [\![O]\!]$$

for $X \in |\mathsf{Set}|$, and with $c : \mathsf{U}_{\mathsf{Set}} \Rightarrow \mathsf{KU}_{\mathsf{Set}}$ being given by:

$$c_{\langle C,\gamma \rangle} = [t]_A$$

for $\langle C, \gamma \rangle \in |\mathsf{Coalg}(\mathsf{Set}, \mathsf{F}_\Pi)|$ with A its associated Π-coalgebra. Finally, Π-equations $t_1 =_O t_2$ induce $(\mathsf{Set}, \mathsf{F}_\Pi)$-coequations (K, c_1, c_2), with K being as before and with $c_1, c_2 : \mathsf{U}_{\mathsf{Set}} \Rightarrow \mathsf{KU}_{\mathsf{Set}}$ being the $(\mathsf{Set}, \mathsf{F}_\Pi)$-observers induced by the Π-terms t_1, t_2, such that $A \models_\Pi t_1 =_O t_2$ if and only if $\langle C, \gamma \rangle \models_{(\mathsf{Set}, \mathsf{F}_\Pi)} (\mathsf{K}, c_1, c_2)$, with A a Π-coalgebra and $\langle C, \gamma \rangle$ its associated $(\mathsf{Set}, \mathsf{F}_\Pi)$-coalgebra.

The notions of satisfaction considered in Examples 3.1.26 and 3.1.27 have the property of being preserved by homomorphisms whose underlying functions are surjective, and reflected by homomorphisms whose underlying functions are injective. The next result generalises this property to the abstract notion of satisfaction of coequations.

Proposition* 3.1.28. *Let* (C, F) *denote an abstract cosignature, let* $f : \langle C, \gamma \rangle \to \langle D, \delta \rangle$ *denote a* (C, F)*-coalgebra homomorphism, and let* (K, l, r) *denote a* (C, F)*-coequation. Then, the following hold:*

1. *If* $\mathsf{U}_\mathsf{C} f$ *is an epimorphism, then* $\langle C, \gamma \rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$ *implies* $\langle D, \delta \rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$.
2. *If* $\mathsf{U}_\mathsf{C} f$ *is a monomorphism, then* $\langle D, \delta \rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$ *implies* $\langle C, \gamma \rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$.

Proof. 1 follows from:

$$\begin{aligned}
\langle C,\gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r) &\Leftrightarrow\\
l_\gamma = r_\gamma &\Rightarrow\\
\mathsf{KU_C}f \circ l_\gamma = \mathsf{KU_C}f \circ r_\gamma &\Leftrightarrow\\
l_\delta \circ \mathsf{U_C}f = r_\delta \circ \mathsf{U_C}f &\Leftrightarrow\\
l_\delta = r_\delta &\Leftrightarrow\\
\langle D,\delta\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)
\end{aligned}$$

while 2 follows from:

$$\begin{aligned}
\langle D,\delta\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r) &\Leftrightarrow\\
l_\delta = r_\delta &\Rightarrow\\
l_\delta \circ \mathsf{U_C}f = r_\delta \circ \mathsf{U_C}f &\Leftrightarrow\\
\mathsf{KU_C}f \circ l_\gamma = \mathsf{KU_C}f \circ r_\gamma &\Leftrightarrow\\
l_\gamma = r_\gamma &\Leftrightarrow\\
\langle C,\gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)
\end{aligned}$$

(The fact that K preserves monomorphisms is used in proving 2.) □

The requirement that $\mathsf{U_C}f$ is an epimorphism (respectively a monomorphism) amounts to f defining a homomorphic image (subcoalgebra).

Another property of the abstract notion of satisfaction of coequations is that it is preserved by colimits in Coalg(C, F).

Proposition* 3.1.29. *Let* (C, F) *denote an abstract cosignature. Also, let* d : D → Coalg(C, F) *denote a diagram of shape* D *in* Coalg(C, F)*, and let* $(\langle C,\gamma\rangle, (f_D : \mathsf{d}D \to \langle C,\gamma\rangle)_{D\in|\mathsf{D}|})$ *denote its colimit. Finally, let* (K,l,r) *denote a* (C, F)*-coequation. If* $\mathsf{d}D \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ *for any* $D \in |\mathsf{D}|$*, then* $\langle C,\gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$*.*

Proof (sketch). The conclusion follows from: $l_\gamma \circ \mathsf{U_C}f_D = \mathsf{KU_C}f_D \circ l_{\mathsf{d}D} = \mathsf{KU_C}f_D \circ r_{\mathsf{d}D} = r_\gamma \circ \mathsf{U_C}f_D$ for $D \in |\mathsf{D}|$, together with $\mathsf{U_C}$ preserving D-colimits. □

Corollary* 3.1.30. *Let* (C, F) *denote an abstract cosignature, and let* E *denote a class of* (C, F)*-coequations. Then,* Coalg(C, F, E) *is a covariety.*

Proof. The conclusion follows by Propositions 3.1.28 and 3.1.29. □

Remark 3.1.31. The question whether any (C, F)-covariety is of form Coalg(C, F, E) for a suitably-chosen E will be discussed in Section 3.1.6.

For a (C, F)-coalgebra $\langle C,\gamma\rangle$ and a (C, F)-coequation e (respectively a class E of (C, F)-coequations), the full subcategory of Coalg(C, F)/γ whose objects satisfy the coequation e (the coequations in

E) is denoted $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$ ($(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^E$). (Recall from Example 2.1.6 that the slice category $\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma$ has objects given by pairs $\langle\langle D,\delta\rangle,d\rangle$ with $\langle D,\delta\rangle$ a (C,F)-coalgebra and $d:\langle D,\delta\rangle\to\langle C,\gamma\rangle$ a (C,F)-coalgebra homomorphism, and arrows from $\langle\langle D,\delta\rangle,d\rangle$ to $\langle\langle D',\delta'\rangle,d'\rangle$ given by (C,F)-coalgebra homomorphisms $f:\langle D,\delta\rangle\to\langle D',\delta'\rangle$ satisfying $d=d'\circ f$.)

The observation that abstract cosignatures induce comonads (see Remark 3.1.5) results in the existence of final objects in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$ and $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^E$.

Proposition 3.1.32. *Let* (C,F) *denote an abstract cosignature, let* $\langle C,\gamma\rangle$ *denote a* (C,F)*-coalgebra, and let* e *denote a* (C,F)*-coequation. Then,* $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$ *has a final object, which at the same time defines a* (C,F)*-subcoalgebra of* $\langle C,\gamma\rangle$.

Proof. Say e is of form (K,l,r). Let $(\mathsf{D},\epsilon,\delta)$ denote the comonad induced by (C,F), let $l^\flat_\gamma, r^\flat_\gamma : \langle C,\gamma\rangle\to\langle\mathsf{DK}C,\gamma'\rangle$ denote the unique (co)extensions of the C-arrows $l_\gamma, r_\gamma : C\to\mathsf{K}C$ to (C,F)-coalgebra homomorphisms, and let $\iota:\langle S,\xi\rangle\to\langle C,\gamma\rangle$ denote an equaliser for $l^\flat_\gamma, r^\flat_\gamma$ (see Corollary 3.1.13). Then, $\mathsf{U_C}\iota$ is a monomorphism (see Corollary 3.1.12), and therefore $\langle\langle S,\xi\rangle,\iota\rangle$ defines a (C,F)-subcoalgebra of $\langle C,\gamma\rangle$. Moreover, $\langle\langle S,\xi\rangle,\iota\rangle$ is final in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$. The fact that $\langle S,\xi\rangle\models_{(\mathsf{C},\mathsf{F})} e$ follows from:

$$\begin{aligned}
\mathsf{KU_C}\iota\circ l_\xi &= \quad (\text{naturality of } l)\\
l_\gamma\circ\mathsf{U_C}\iota &= \quad (\text{definition of } l^\flat_\gamma)\\
\epsilon_{\mathsf{K}C}\circ\mathsf{U_C}l^\flat_\gamma\circ\mathsf{U_C}\iota &= \quad (\text{definition of } \iota)\\
\epsilon_{\mathsf{K}C}\circ\mathsf{U_C}r^\flat_\gamma\circ\mathsf{U_C}\iota &= \quad (\text{definition of } r^\flat_\gamma)\\
r_\gamma\circ\mathsf{U_C}\iota &= \quad (\text{naturality of } r)\\
\mathsf{KU_C}\iota\circ r_\xi &
\end{aligned}$$

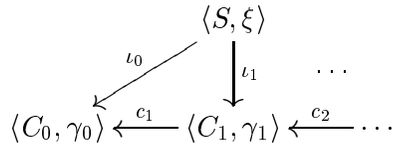
together with the observation that $\mathsf{KU_C}\iota$ is a monomorphism (as $\mathsf{U_C}\iota$ is a monomorphism, while K preserves monomorphisms). Also, finality of $\langle\langle S,\xi\rangle,\iota\rangle$ in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$ follows from the uniqueness of (C,F)-coalgebra homomorphisms with codomain $\langle\mathsf{DK}C,\gamma'\rangle$ (co)extending a given C-arrow with codomain $\mathsf{K}C$, together with the couniversality of ι. For, given $\langle\langle D,\delta\rangle,d\rangle\in|(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e|$, the fact that $\langle D,\delta\rangle\models_{(\mathsf{C},\mathsf{F})} e$ together with the naturality of l,r yield $l_\gamma\circ\mathsf{U_C}d=r_\gamma\circ\mathsf{U_C}d$, that is, $\epsilon_{\mathsf{K}C}\circ\mathsf{U_C}(l^\flat_\gamma\circ d)=\epsilon_{\mathsf{K}C}\circ\mathsf{U_C}(r^\flat_\gamma\circ d)$. Hence, $l^\flat_\gamma\circ d=r^\flat_\gamma\circ d$. Then, the couniversality of ι yields a unique (C,F)-coalgebra homomorphism $e:\langle D,\delta\rangle\to\langle S,\xi\rangle$ satisfying $d=\iota\circ e$. That is, there exists precisely one $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^e$-arrow from $\langle\langle D,\delta\rangle,d\rangle$ to $\langle\langle S,\xi\rangle,\iota\rangle$. This concludes the proof. □

The existence of limits of ω^{op}-chains in $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$ (following from the existence of such limits in C together with their preservation by F, see Corollary 2.4.8) now yields the following.

Proposition 3.1.33. *Let* (C,F) *denote an abstract cosignature, let* $\langle C,\gamma\rangle$ *denote a* (C,F)*-coalgebra, and let* E *denote an enumerable set of* (C,F)*-coequations. Then,* $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^E$ *has a final object, which at the same time defines a* (C,F)*-subcoalgebra of* $\langle C,\gamma\rangle$.

Proof. Say $E=\{\ e_i\mid i\in\mathbb{N}\ \}$. Now let $\langle C_0,\gamma_0\rangle=\langle C,\gamma\rangle$. Also, for $i\in\mathbb{N}$, let $\langle\langle C_{i+1},\gamma_{i+1}\rangle,c_{i+1}\rangle$ denote a final object in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma_i)^{e_i}$. Finally, let $(\langle S,\xi\rangle,(\iota_i)_{i\in\mathbb{N}})$ define a limit for the ω^{op}-chain

defined by $c_1, c_2, \ldots$.



The fact that $\mathsf{U_C}$ creates limits of ω^{op}-chains together with the fact that each of $\mathsf{U_C}c_1, \mathsf{U_C}c_2, \ldots$ are monomorphisms result in each of $\mathsf{U_C}\iota_0, \mathsf{U_C}\iota_1, \ldots$ being monomorphisms. In particular, $\mathsf{U_C}\iota_0$ is a monomorphism, and hence $\langle\langle S, \xi\rangle, \iota_0\rangle$ defines a (C, F)-subcoalgebra of $\langle C, \gamma\rangle$. Moreover, $\langle S, \xi\rangle \models_{(\mathsf{C},\mathsf{F})} E$. This follows by 2 of Proposition 3.1.28 from $\mathsf{U_C}\iota_{i+1}$ being a monomorphism together with $\langle C_{i+1}, \gamma_{i+1}\rangle \models_{(\mathsf{C},\mathsf{F})} e_i$, for $i \in \mathbb{N}$. Also, the fact that $\langle\langle C_{i+1}, \gamma_{i+1}\rangle, c_{i+1}\rangle$ is final in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma_i)^{e_i}$, for $i \in \mathbb{N}$ results in $\langle\langle S, \xi\rangle, \iota_0\rangle$ being final in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^E$. For, given $\langle\langle D, \delta\rangle, d\rangle \in |(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma)^E|$, the finality of $\langle\langle C_{i+1}, \gamma_{i+1}\rangle, c_{i+1}\rangle$ in $(\mathsf{Coalg}(\mathsf{C},\mathsf{F})/\gamma_i)^{e_{i+1}}$, with $i = 0, 1, \ldots$ successively yields (C, F)-coalgebra homomorphisms $d_{i+1} : \langle D, \delta\rangle \to \langle C_{i+1}, \gamma_{i+1}\rangle$ satisfying $c_{i+1} \circ d_{i+1} = d_i$, for $i = 0, 1, \ldots$ (with $d_0 = d$). The couniversality of $(\langle S, \xi\rangle, (\iota_i)_{i\in\mathbb{N}})$ then yields a unique (C, F)-coalgebra homomorphism $f : \langle D, \delta\rangle \to \langle S, \xi\rangle$ satisfying $\iota_i \circ f = d_i$ for $i \in \mathbb{N}$. In particular, $\iota_0 \circ f = d$. This concludes the proof. □

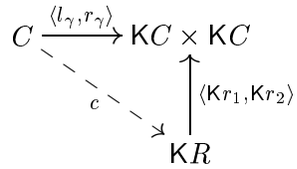
Corollary 3.1.34. *Let* (C, F) *denote an abstract cosignature, and let* E *denote an enumerable set of* (C, F)*-coequations. Then,* $\mathsf{Coalg}(\mathsf{C},\mathsf{F},E)$ *has a final object.*

Proof. Propositions 3.1.9 and 3.1.33 are used. □

A notion of satisfaction of coequations up to bisimulation, which generalises the various notions of satisfaction up to observability employed by existing equational specification formalisms, including [GM97, Jac96a, Jac97] can also be defined.

Definition* 3.1.35. *Let* (C, F) *denote an abstract cosignature. A* (C, F)*-coalgebra* $\langle C, \gamma\rangle$ **satisfies** *a* (C, F)*-coequation* (K, l, r) **up to bisimulation** *(written* $\langle C, \gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$*) if and only if* $\mathsf{KU_C}! \circ l_\gamma = \mathsf{KU_C}! \circ r_\gamma$*, with* $! : \langle C, \gamma\rangle \to \langle F, \zeta\rangle$ *denoting the unique* (C, F)*-coalgebra homomorphism from* $\langle C, \gamma\rangle$ *to the final* (C, F)*-coalgebra.*

Remark 3.1.36. If K preserves kernel pairs, then $\langle C, \gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$ holds precisely when $\langle l_\gamma, r_\gamma\rangle$ factors through $\langle \mathsf{K}r_1, \mathsf{K}r_2\rangle$:



with $\langle R, r_1, r_2\rangle$ denoting (C, F)-bisimilarity on $\langle C, \gamma\rangle$. For, in this case, $\mathsf{K}r_1, \mathsf{K}r_2$ define a kernel pair for $\mathsf{KU_C}!$ (see also Remark 3.1.14).

Example 3.1.37 (**Behavioural Equational Satisfaction in Hidden Algebra***).* Let (H, Σ) denote a hidden signature (see Example 3.1.6). A Σ-context c (see Example 3.1.16) is said to be *appropriate* [GM97] for a Σ-term t if and only if the sort of t matches that of the variable z. In this case, one writes

$c[t]$ for the result of substituting t for z in c. A Σ-algebra A is said to *behaviourally satisfy* [GM97] an unconditional Σ-equation e of form $(\forall\, \mathcal{V})\ l = r$ (written $A \models\!\!\equiv_\Sigma e$) if and only if $\theta^\#(c[l]) = \theta^\#(c[r])$ holds for any Σ-context c appropriate for l, r, and any assignment $\theta : \mathcal{V} \to \mathsf{U}A$. Also, A is said to *behaviourally satisfy* [GM97] a conditional Σ-equation e of form $(\forall\, \mathcal{V})\ l = r$ if $l_1 = r_1, \ldots, l_n = r_n$ (written $A \models\!\!\equiv_\Sigma e$) if and only if $\theta^\#(c[l]) = \theta^\#(c[r])$ holds for any Σ-context c appropriate for l, r whenever $\theta^\#(c_i[l_i]) = \theta^\#(c_i[r_i])$ holds for any Σ-context c_i appropriate for l_i, r_i, for $i = 1, \ldots, n$, for any assignment $\theta : \mathcal{V} \to \mathsf{U}A$. Equivalently, $A \models\!\!\equiv_\Sigma e$ if and only if $\theta^\#(l) \sim \theta^\#(r)$ whenever $\theta^\#(l_i) \sim \theta^\#(r_i)$ for $i = 1, \ldots, n$, for any assignment $\theta : \mathcal{V} \to \mathsf{U}A$ (with $\sim$ denoting behavioural equivalence on A).

Now let $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$ denote the abstract cosignature induced by a destructor hidden signature Σ (see Example 3.1.6). Also, let e denote a conditional Σ-equation of form $(\forall\, Z)(\forall\, X_1) \ldots (\forall\, X_m)\ l = r$ if $l_1 = r_1, \ldots, l_n = r_n$, with $Z : h$ and $X_i : v_i$ for $i = 1, \ldots, m$. Then, e induces a $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$-coequation c, with c being constructed as in Example 3.1.26, except that the conditions $(l_i)_A(a) = (r_i)_A(a)$ are replaced by $(l_i)_A(a) \sim (r_i)_A(a)$. Moreover, $A \models\!\!\equiv_\Sigma e$ if and only if $\langle C, \gamma\rangle \models^{\mathsf{b}}_{(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)} c$. This follows by the previously-mentioned characterisation of behavioural satisfaction in terms of behavioural equivalence.

Consider, for instance, the equations used to specify the behaviour of cells (see Example 3.1.26). The first of these equations, namely $(\forall\, \mathtt{N})(\forall\, \mathtt{S})\ \mathtt{getx}(\mathtt{putx}(\mathtt{N}, \mathtt{S})) = \mathtt{N}$, is of visible sort, and consequently its behavioural satisfaction by an algebra of the cell signature is equivalent to its standard satisfaction by that algebra. The same can not, however, be said about the equation: $(\forall\, \mathtt{S})(\forall\, \mathtt{N})(\forall\, \mathtt{M})\ \mathtt{putx}(\mathtt{N}, \mathtt{putx}(\mathtt{M}, \mathtt{S})) = \mathtt{putx}(\mathtt{N}, \mathtt{S})$. Its behavioural satisfaction by algebras of the cell signature will later be shown to follow from the (behavioural) satisfaction of the previous equation by such algebras.

Standard satisfaction of (K, l, r) by $\langle C, \gamma\rangle$ implies its satisfaction up to bisimulation by $\langle C, \gamma\rangle$. And if, in addition, $\langle C, \gamma\rangle$ is observable, then the converse also holds. For, in this case, $\mathsf{U_C}!$ is a monomorphism, while K preserves monomorphisms.

Satisfaction up to bisimulation can be expressed in terms of standard satisfaction by the codomains of the quotients of the unique homomorphisms into the final coalgebra.

Proposition 3.1.38. *Let* (C, F) *denote an abstract cosignature, let* $\langle C, \gamma\rangle$ *denote a* (C, F)*-coalgebra, and let* (K, l, r) *denote a* (C, F)*-coequation. Also, let* $e : \langle C, \gamma\rangle \to \langle E, \eta\rangle$ *denote the quotient of the unique* (C, F)*-coalgebra homomorphism from* $\langle C, \gamma\rangle$ *to the final* (C, F)*-coalgebra. Then,* $\langle C, \gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$ *if and only if* $\langle E, \eta\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$.

Proof. Let $!_\gamma : \langle C, \gamma\rangle \to \langle F, \zeta\rangle$ and $!_\eta : \langle E, \eta\rangle \to \langle F, \zeta\rangle$ denote the unique (C, F)-coalgebra homomorphisms from $\langle C, \gamma\rangle$ and respectively $\langle E, \eta\rangle$ to the final (C, F)-coalgebra. Also, note that $\mathsf{U_C}e$ is an epimorphism (see Corollary 3.1.22), whereas $\mathsf{KU_C}!_\eta$ is a monomorphism (as $\mathsf{U_C}!_\eta$ is a monomorphism (see Corollary 3.1.22), while K preserves monomorphisms). Then, the conclusion

follows from:

$$\begin{aligned}
\langle C,\gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r) &\Leftrightarrow\\
\mathsf{KU_C}!_\gamma \circ l_\gamma = \mathsf{KU_C}!_\gamma \circ r_\gamma &\Leftrightarrow\\
\mathsf{KU_C}!_\eta \circ \mathsf{KU_C}e \circ l_\gamma = \mathsf{KU_C}!_\eta \circ \mathsf{KU_C}e \circ r_\gamma &\Leftrightarrow\\
\mathsf{KU_C}!_\eta \circ l_\eta \circ \mathsf{U_C}e = \mathsf{KU_C}!_\eta \circ r_\eta \circ \mathsf{U_C}e &\Leftrightarrow\\
\mathsf{KU_C}!_\eta \circ l_\eta = \mathsf{KU_C}!_\eta \circ r_\eta &\Leftrightarrow\\
l_\eta = r_\eta &\Leftrightarrow\\
\langle E,\eta\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)
\end{aligned}$$

□

The notion of satisfaction of coequations up to bisimulation enjoys properties similar to those of standard satisfaction. In particular, Propositions 3.1.28, 3.1.29, 3.1.32 and 3.1.33, as well as Corollaries 3.1.30 and 3.1.34 hold. Moreover, the implication in 1 of Proposition 3.1.28 becomes an equivalence, while 2 of Proposition 3.1.28 requires no restriction on the homomorphism involved.

Proposition* 3.1.39. *Let* (C,F) *denote an abstract cosignature, let* $f : \langle C,\gamma\rangle \to \langle D,\delta\rangle$ *denote a* (C,F)*-coalgebra homomorphism, and let* (K,l,r) *denote a* (C,F)*-coequation. Then, the following hold:*

1. *If* $\mathsf{U_C}f$ *is an epimorphism, then* $\langle C,\gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ *implies* $\langle D,\delta\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$.
2. $\langle D,\delta\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ *implies* $\langle C,\gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$.

Proof. Let $!_\gamma : \langle C,\gamma\rangle \to \langle F,\zeta\rangle$ and $!_\delta : \langle D,\delta\rangle \to \langle F,\zeta\rangle$ denote the unique (C,F)-coalgebra homomorphisms from $\langle C,\gamma\rangle$ and respectively $\langle D,\delta\rangle$ to the final (C,F)-coalgebra. Then, the conclusion follows from:

$$\begin{aligned}
\langle D,\delta\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r) &\Leftrightarrow\\
\mathsf{KU_C}!_\delta \circ l_\delta = \mathsf{KU_C}!_\delta \circ r_\delta &\Rightarrow\\
\mathsf{KU_C}!_\delta \circ l_\delta \circ \mathsf{U_C}f = \mathsf{KU_C}!_\delta \circ r_\delta \circ \mathsf{U_C}f &\Leftrightarrow\\
\mathsf{KU_C}!_\delta \circ \mathsf{KU_C}f \circ l_\gamma = \mathsf{KU_C}!_\delta \circ \mathsf{KU_C}f \circ r_\gamma &\Leftrightarrow\\
\mathsf{KU_C}!_\gamma \circ l_\gamma = \mathsf{KU_C}!_\gamma \circ r_\gamma &\Leftrightarrow\\
\langle C,\gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)
\end{aligned}$$

with the implication in the second line of the proof becoming an equivalence if $\mathsf{U_C}f$ is an epimorphism. □

Proposition* 3.1.40. *Let* (C,F) *denote an abstract cosignature. Also, let* $\mathsf{d} : \mathsf{D} \to \mathsf{Coalg}(\mathsf{C},\mathsf{F})$ *denote a diagram of shape* D *in* $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$*, and let* $(\langle C,\gamma\rangle, (f_D : \mathsf{d}D \to \langle C,\gamma\rangle)_{D\in|\mathsf{D}|})$ *denote its colimit. Finally, let* (K,l,r) *denote a* (C,F)*-coequation. If* $\mathsf{d}D \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ *for any* $D \in |\mathsf{D}|$*, then* $\langle C,\gamma\rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$.

Proof. Similar to the proof of Proposition 3.1.29. □

Corollary* 3.1.41. *Let* (C, F) *denote an abstract cosignature, and let* E *denote a class of* (C, F)*-coequations. Then, the full subcategory of* Coalg(C, F) *whose objects satisfy the coequations in* E *up to bisimulation is a covariety.*

Proposition 3.1.42. *Let* (C, F) *denote an abstract cosignature, let* $\langle C, \gamma \rangle$ *denote a* (C, F)*-coalgebra, and let* E *denote an enumerable set of* (C, F)*-coequations. Then, the full subcategory of* Coalg(C, F)$/\gamma$ *whose objects satisfy the coequations in* E *up to bisimulation has a final object. Furthermore, the final object defines a* (C, F)*-subcoalgebra of* $\langle C, \gamma \rangle$*.*

Proof. Similar to the proof of Proposition 3.1.33. (A version of Proposition 3.1.32 concerning the satisfaction of equations up to bisimulation is also needed.) □

The following also holds.

Proposition 3.1.43. *Let* (C, F) *denote an abstract cosignature, let* E *denote an enumerable set of* (C, F)*-coequations, and let* $\langle F', \zeta' \rangle$ *denote a final* (C, F, E)*-coalgebra*[5]*. Then,* $\langle F', \zeta' \rangle$ *is final amongst the* (C, F)*-coalgebras satisfying the coequations in* E *up to bisimulation. Moreover, a* (C, F)*-coequation holds in* $\langle F', \zeta' \rangle$ *if and only if it holds, up to bisimulation, in all* (C, F)*-coalgebras satisfying* E *up to bisimulation.*

Proof. Say $E = \{ e_i \mid i \in \mathbb{N} \}$. Finality of $\langle F', \zeta' \rangle$ amongst the (C, F)-coalgebras satisfying E up to bisimulation follows from its finality in Coalg(C, F, E) together with the observation that, if $\langle C, \gamma \rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} E$, then $\langle C, \gamma \rangle$ defines the object of a cone on the following ω-chain:

$$\langle F_0, \zeta_0 \rangle = \langle F, \zeta \rangle \xleftarrow{f_1} \langle F_1, \zeta_1 \rangle \xleftarrow{f_2} \cdots$$

with $\langle F, \zeta \rangle$ denoting a final (C, F)-coalgebra, and with $\langle \langle F_{i+1}, \zeta_{i+1} \rangle, f_{i+1} \rangle$ denoting a final object in (Coalg(C, F)$/\zeta_i)^{e_i}$, for $i = 0, 1, \ldots$.

Then, the **if** direction follows from $\langle F', \zeta' \rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} E$ (as $\langle F', \zeta' \rangle \models_{(\mathsf{C},\mathsf{F})} E$), while the **only if** direction follows from the finality of $\langle F', \zeta' \rangle$ amongst the (C, F)-coalgebras satisfying E up to bisimulation, together with the satisfaction of coequations up to bisimulation being reflected by coalgebra homomorphisms. □

Provided that the functors K used to define the types of coequations preserve kernel pairs, proofs of satisfaction of coequations up to bisimulation can benefit from the use of *generic bisimulations*, as defined below.

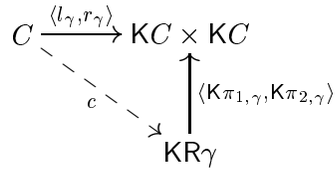
Definition 3.1.44. *Let* (C, F) *denote an abstract cosignature, and let* $\mathcal{C}$ *denote a full subcategory of* Coalg(C, F)[6]*. A* **generic** (C, F)**-bisimulation on** $\mathcal{C}$ *is given by a tuple* $\langle \mathsf{R}, \pi_1, \pi_2 \rangle$ *with* $\mathsf{R} : \mathcal{C} \to \mathsf{C}$ *and* $\pi_1, \pi_2 : \mathsf{R} \Rightarrow \mathsf{U_C} \restriction_{\mathcal{C}}$*, such that* $\langle \mathsf{R}\gamma, \pi_{1,\gamma}, \pi_{2,\gamma} \rangle$ *defines a* (C, F)*-bisimulation on* $\langle C, \gamma \rangle$ *for any* $\langle C, \gamma \rangle \in |\mathcal{C}|$*.*

[5] Hence, $\langle F', \zeta' \rangle \models^{\mathsf{b}}_{(\mathsf{C},\mathsf{F})} E$.

[6] $\mathcal{C}$ could, for instance, consist of all (C, F)-coalgebras satisfying (possibly only up to bisimulation) a given set of (C, F)-coequations.

That is, a generic bisimulation on $\mathcal{C}$ associates to each coalgebra in $\mathcal{C}$ a bisimulation relation on it, with this association being functorial.

Then, proving that a (C, F)-coequation (K, l, r) with K preserving kernel pairs holds, up to bisimulation, in a full subcategory $\mathcal{C}$ of Coalg(C, F) can be reduced to exhibiting a generic (C, F)-bisimulation $\langle \mathsf{R}, \pi_1, \pi_2 \rangle$ on $\mathcal{C}$, such that $\langle l_\gamma, r_\gamma \rangle$ factors through $\langle \mathsf{K}\pi_{1,\gamma}, \mathsf{K}\pi_{2,\gamma} \rangle$ for any $\langle C, \gamma \rangle \in |\mathcal{C}|$:



(see also Remark 3.1.36).

Example 3.1.45 (**Hidden Coinduction***).* Let Σ denote a hidden signature. A *hidden congruence* on a Σ-algebra A is a many-sorted Σ-congruence $\equiv$ on A (see Definition 2.3.22) additionally satisfying: $\equiv_v \; = \; =_v$ for $v \in V$. Then, behavioural equivalence on a Σ-algebra A is the largest hidden congruence on A. This observation is exploited in [GM99, GM00] to derive a proof technique for behavioural satisfaction called *hidden coinduction*. Using this technique for proving the behavioural satisfaction of an equation by an algebra amounts to exhibiting a hidden congruence on that algebra which relates the interpretations of the lhs and rhs of the given equation. The technique can also be used to prove the behavioural satisfaction of an equation by all the algebras satisfying a given equational specification. In this case, the required hidden congruences are defined generically, typically as behavioural equivalence relations w.r.t. a subsignature of the underlying hidden signature. The use of generic bisimulations in proving the satisfaction of coequations up to bisimulation generalises the use of such generic hidden congruences in proving the behavioural satisfaction of hidden equations. An example of a such a proof is given in the following. Consider the specification of cells holding natural numbers, given by the hidden signature in Example 3.1.6 together with the equation $(\forall\, \mathtt{N})(\forall\, \mathtt{S})\ \mathtt{getx}(\mathtt{putx}(\mathtt{N},\mathtt{S})) = \mathtt{N}$. Also, for an algebra A of this specification, consider the relation $\equiv$ on A given by:

- $n \equiv_{\mathtt{Nat}} n'$ if and only if $n = n'$
- $a \equiv_{\mathtt{State}} a'$ if and only if $\mathtt{getx}_A(a) = \mathtt{getx}_A(a')$.

Then, $a \equiv_{\mathtt{State}} a'$ implies $\mathtt{putx}(n, a) \equiv_{\mathtt{State}} \mathtt{putx}(n, a')$ for any $n \in A_{\mathtt{Nat}}$. (The fact that $A \models (\forall\, \mathtt{N})(\forall\, \mathtt{S})\ \mathtt{getx(putx(N,S))} = \mathtt{N}$ is used here.) That is, $\equiv$ is a hidden congruence on A, and consequently $\equiv \subseteq \sim$ (with $\sim$ denoting behavioural equivalence on A). As a result, proving that $A \models (\forall\, \mathtt{S})(\forall\, \mathtt{N})(\forall\, \mathtt{M})\ \mathtt{putx(N,putx(M,S))} = \mathtt{putx(N,S)}$ can be reduced to proving that $A \models (\forall\, \mathtt{S})(\forall\, \mathtt{N})(\forall\, \mathtt{M})\ \mathtt{getx(putx(N,putx(M,S)))} = \mathtt{getx(putx(N,S))}$. But this follows from $A \models (\forall\, \mathtt{N})(\forall\, \mathtt{S})\ \mathtt{getx(putx(N,S))} = \mathtt{N}$ by standard equational reasoning.

Example 3.1.46. The *assertions* used in [Jac96c, Jac96a, Jac97] to specify object behaviour are an instance of the abstract notion of coequation. To illustrate this, we consider the following example of an object specification, taken from [Jac96a]:

class spec: Stack(A)
methods:
push : $X \times A \to X$
pop : $X \to X$
top : $X \to 1 + A$
assertions:
s.push(a).top $= a$
s.push(a).pop $\leftrightarrows s$
s.top $= * \vdash s$.pop $\leftrightarrows s$
creation:
new.top $= *$
end class spec

The models of the above specification are given by coalgebras of the endofunctor G : Set $\to$ Set, $\mathsf{G}X = X^A \times X \times (1 + A)$ (or, equivalently, by tuples $\langle C, \mathsf{push}_C, \mathsf{pop}_C, \mathsf{top}_C \rangle$, with $C \in |\mathsf{Set}|$, $\mathsf{push}_C : C \times A \to C$, $\mathsf{pop}_C : C \to C$ and $\mathsf{top}_C : C \to 1 + A$) which, in addition, satisfy the specified assertions. (The satisfaction of assertions by coalgebras is defined by interpreting the symbol $\leftrightarrows$ as bisimilarity on those coalgebras.)

Since assertions refer to single states (denoted by s in the above example), they induce (Set, G)-coequations. For instance, the (Set, G)-coequation (K, l, r) induced by the last of the above assertions has K : Set $\to$ Set given by $\mathsf{K}X = 1 + X$ for $X \in |\mathsf{Set}|$, and $l, r : \mathsf{U} \Rightarrow \mathsf{KU}$ (with U : Coalg(Set, G) $\to$ Set taking (Set, G)-coalgebras to their carrier) given by:

$$l_{\langle C,\gamma\rangle}(c) = \begin{cases} \iota_2(\mathsf{pop}_C(c)) & \text{if } \mathsf{top}_C(c) \in \iota_1(1) \\ \iota_1(*) & \text{if } \mathsf{top}_C(c) \in \iota_2(A) \end{cases}$$

$$r_{\langle C,\gamma\rangle}(c) = \begin{cases} \iota_2(c) & \text{if } \mathsf{top}_C(c) \in \iota_1(1) \\ \iota_1(*) & \text{if } \mathsf{top}_C(c) \in \iota_2(A) \end{cases}$$

for any (Set, G)-coalgebra $\langle C, \gamma \rangle$, with $\mathsf{push}_C : C \times A \to C$, $\mathsf{pop}_C : C \to C$ and $\mathsf{top}_C : C \to 1 + A$ alternatively defining its structure, and any $c \in C$. Furthermore, a coalgebra satisfies a given assertion precisely when it satisfies the induced coequation up to bisimulation.

3.1.3 Institutions of Observational Structures

Translations between abstract cosignatures, specifying a change in the type of information that can be observed about a system, are captured by *abstract cosignature morphisms*.

Definition* 3.1.47. *An* **(abstract) cosignature morphism** *between abstract cosignatures* (C, F) *and* (D, G) *is a pair* (U, η)*, with* U : D $\to$ C *a functor which preserves limits*[7] *and has a right adjoint* R[8]*, and with* η : UG $\Rightarrow$ FU *a natural transformation. If, in addition,* U *lifts limits and colimits*[9]*, and if* R *is also a right inverse to* U*, then the cosignature morphism* (U, η) *is called* **strong**.

[7] It would be sufficient to require that U preserves pullbacks and limits of ω^{op}-chains.
[8] Consequently, U also preserves colimits (see Proposition 2.1.54).
[9] Hence, by Proposition 2.1.35, U preserves limits and colimits.

If $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ and $(\mathsf{V}, \xi) : (\mathsf{D}, \mathsf{G}) \to (\mathsf{E}, \mathsf{H})$ are (strong) abstract cosignature morphisms, then so is $(\mathsf{UV}, \eta_{\mathsf{V}} \circ \mathsf{U}\xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{E}, \mathsf{H})$. The quasi-category of abstract cosignatures and abstract cosignature morphisms is denoted Cosign, while the quasi-category of abstract cosignatures and strong abstract cosignature morphisms is denoted SCosign.

An abstract cosignature morphism $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ induces a reduct functor $\mathsf{U}_\eta : \mathsf{Coalg}(\mathsf{D}, \mathsf{G}) \to \mathsf{Coalg}(\mathsf{C}, \mathsf{F})$, with U_η taking a (D, G)-coalgebra $\langle C, \gamma \rangle$ to the (C, F)-coalgebra $\langle \mathsf{U}C, \eta_C \circ \mathsf{U}\gamma \rangle$. (Intuitively, the action of U_η can be regarded as extracting a (C, F)-(sub)system from a given (D, G)-system.) This yields a functor $\mathsf{Coalg} : \mathsf{Cosign} \to \mathsf{CAT}^{\mathsf{op}}$, taking an abstract cosignature to its category of coalgebras, and an abstract cosignature morphism to the induced reduct functor.

An abstract cosignature morphism $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ also induces a translation, itself denoted η, of (C, F)-observers to (D, G)-observers. This translation takes a (C, F)-observer (K, c) to the (D, G)-observer $(\mathsf{RKU}, c^\flat_{\mathsf{U}_\eta})$:

$$\begin{array}{ccc} \mathsf{U_C U}_\eta = \mathsf{U U_D} & & \mathsf{U_D} \\ {\scriptstyle c\mathsf{U}_\eta}\Downarrow & & {\scriptstyle c^\flat_{\mathsf{U}_\eta}}\Downarrow \\ \mathsf{KU_C U}_\eta = \mathsf{KUU_D} & & \mathsf{RKUU_D} \end{array}$$

where:

$$\begin{array}{ccc} \mathsf{Coalg}(\mathsf{C}, \mathsf{F}) & \xleftarrow{\mathsf{U}_\eta} & \mathsf{Coalg}(\mathsf{D}, \mathsf{G}) \\ {\scriptstyle \mathsf{U_C}}\downarrow & & \downarrow{\scriptstyle \mathsf{U_D}} \\ \mathsf{C} & \underset{\mathsf{R}}{\overset{\mathsf{U}}{\leftarrow\!\!\dashrightarrow}} & \mathsf{D} \end{array}$$

The preservation of monomorphisms by RKU follows from the preservation of monomorphisms by each of U (as U preserves pullbacks), K and R. And if, in addition, K preserves kernel pairs, then so does RKU (as both U and R preserve kernel pairs).

The translation of (C, F)-observers into (D, G)-observers extends to a translation of (C, F)-coequations into (D, G)-coequations. This yields a functor $\mathsf{Coeqn} : \mathsf{Cosign} \to \mathsf{SET}$, taking abstract cosignatures to their classes of coequations, and abstract cosignature morphisms to the functions translating coequations over their source to coequations over their target.

As one would expect, a (C, F)-coequation holds in the (C, F)-reduct of a (D, G)-coalgebra if and only if its translation along the cosignature morphism $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ holds in the given (D, G)-coalgebra.

Proposition* 3.1.48. *Let $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ denote an abstract cosignature morphism, let $\langle D, \delta \rangle$ denote a (D, G)-coalgebra, and let e denote a (C, F)-coequation. Then, $\mathsf{U}_\eta \langle D, \delta \rangle \models_{(\mathsf{C},\mathsf{F})} e$ if and only if $\langle D, \delta \rangle \models_{(\mathsf{D},\mathsf{G})} \eta(e)$.*

Proof. If e is of form (K, l, r), then $\mathsf{U}_\eta \langle D, \delta \rangle \models_{(\mathsf{C},\mathsf{F})} e$ translates to $l_{\mathsf{U}_\eta \delta} = r_{\mathsf{U}_\eta \delta}$, while $\langle D, \delta \rangle \models_{(\mathsf{D},\mathsf{G})} \eta(e)$ translates to $l^\flat_{\mathsf{U}_\eta \delta} = r^\flat_{\mathsf{U}_\eta \delta}$. The conclusion then follows by standard properties of adjunctions. □

Theorem* 3.1.49. $(\mathsf{Cosign}, \mathsf{Coalg}, \mathsf{Coeqn}, \models)$ *is an institution.*

Definition* 3.1.50. *A specification (respectively specification morphism) of this institution is called an* **abstract coalgebraic specification** *(***specification morphism***).*

Since final coalgebras are used as denotations for coalgebraic specifications, of particular interest amongst coalgebraic specification morphisms will prove those which preserve these denotations.

Definition* 3.1.51. *A coalgebraic specification morphism* $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}, E) \to (\mathsf{D}, \mathsf{G}, E')$ *is* **conservative** *if and only if* $\mathsf{U}_\eta\langle F', \zeta'\rangle \simeq \langle F, \zeta\rangle$*, with* $\langle F, \zeta\rangle$ *and* $\langle F', \zeta'\rangle$ *denoting the final* $(\mathsf{C}, \mathsf{F}, E)$*- and respectively* $(\mathsf{D}, \mathsf{G}, E')$*-coalgebras*[10]*.*

In Chapter 4 (see e.g. Example 4.2.20), conservative specification morphisms will be shown to arise e.g. from *coinductive definitions*. (Such definitions can be regarded as definitions up to bisimulation.)

Example 3.1.52 (**Destructor Hidden Signature Maps***).* Let (H, Σ) and (H', Σ') denote hidden signatures over (V, Ψ). A *hidden signature map* [GM00] $\phi : (H, \Sigma) \to (H', \Sigma')$ is a many-sorted signature morphism $\phi : (V \cup H, \Sigma) \to (V \cup H', \Sigma')$ additionally satisfying: $\phi{\restriction_\Psi} = \iota_\Psi : \Psi \to \Sigma'$ and $\phi(H) \subseteq H'$. Hidden signature maps $\phi : (H, \Sigma) \to (H', \Sigma')$ induce reduct functors $\mathsf{U}_\phi : \mathsf{Alg}(\Sigma') \to \mathsf{Alg}(\Sigma)$, with the action of U_ϕ on a hidden Σ-algebra A being given by the action of the reduct functor induced by the underlying many-sorted signature morphism on the many-sorted Σ-algebra A.

Now let (H, Σ) and (H', Σ') denote destructor hidden signatures, and let $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$ and $(\mathsf{Set}_D^{S'}, \mathsf{F}_{\Sigma'})$ denote the abstract cosignatures induced by Σ and respectively Σ' (see Example 3.1.6). Then, hidden signature maps $\phi : (H, \Sigma) \to (H', \Sigma')$ induce abstract cosignature morphisms (U, η_ϕ), with $\mathsf{U} : \mathsf{Set}_D^{S'} \to \mathsf{Set}_D^S$ taking $A' \in |\mathsf{Set}_D^{S'}|$ to $(A'_{\phi(s)})_{s \in S} \in |\mathsf{Set}_D^S|$, and with $\eta_\phi : \mathsf{UF}_{\Sigma'} \Rightarrow \mathsf{F}_\Sigma\mathsf{U}$ being given by:

$$(\eta_{\phi,X})_h(f) = (f_{\phi(\sigma)})_{\sigma \in \Sigma_{hw,s}} \text{ if } f = (f_{\sigma'})_{\sigma' \in \Sigma'_{\phi(h)w',s'}} \in \prod_{\sigma' \in \Sigma'_{\phi(h)w',s'}} X_{s'}^{D_{w'}}, \quad h \in H$$

$$(\eta_{\phi,X})_v = 1_{D_v}, \quad v \in V$$

for $X \in |\mathsf{Set}_D^{S'}|$. For, the functor U preserves limits (see Example 2.1.38) and has a right adjoint R (see Example 2.1.52). And if, in addition, ϕ is injective on sorts, then U lifts limits and colimits (see Example 2.1.38), while R is also a right inverse to U (see Example 2.1.52). Moreover, U_{η_ϕ} agrees with U_ϕ, while the translation of $(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)$-coequations along η_ϕ agrees with the translation of Σ-equations in one hidden-sorted variable along ϕ.

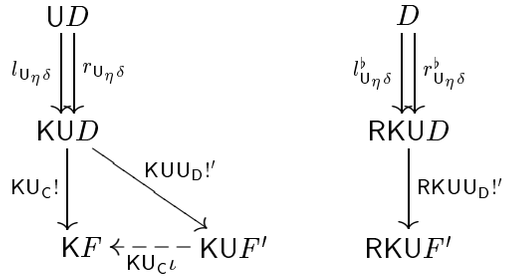
Since the notions of bisimilarity associated to the source and target of cosignature morphisms do not, in general, coincide (that is, the largest bisimulation on the reduct of a coalgebra of the target cosignature is not, in general, isomorphic to the image under U of the largest bisimulation on the original coalgebra), the notion of cosignature morphism does not give rise to an institution w.r.t. the satisfaction of coequations up to bisimulation. However, restricting attention to a certain category of cosignature morphisms does yield an institution, as shown in the following.

[10] E and E' are assumed to be enumerable sets, in order to ensure the existence of final $(\mathsf{C}, \mathsf{F}, E)$- and $(\mathsf{D}, \mathsf{G}, E')$-coalgebras (see Proposition 3.1.34).

Definition* 3.1.53. *An abstract cosignature morphism* $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ *is* **horizontal** *if and only if* η *is a natural monomorphism.*

The quasi-category of abstract cosignatures and horizontal abstract cosignature morphisms is denoted HCosign.

Remark 3.1.54. If $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ denotes a horizontal abstract cosignature morphism, then the C-arrow underlying the unique (C, F)-coalgebra homomorphism from the (C, F)-reduct of a final (D,G)-coalgebra to a final (C, F)-coalgebra is a monomorphism. (This follows by an argument similar to the one given in Remark 2.3.53.) Consequently, if $\langle D, \delta \rangle$ denotes a (D, G)-coalgebra, and if $! : \mathsf{U}_\eta \langle D, \delta \rangle \to \langle F, \zeta \rangle$ and $!' : \langle D, \delta \rangle \to \langle F', \zeta' \rangle$ denote the unique coalgebra homomorphisms from $\mathsf{U}_\eta \langle D, \delta \rangle$ and $\langle D, \delta \rangle$ to the final (C, F)- and respectively (D, G)-coalgebras, then $\mathsf{KU_C}! \circ l_{\mathsf{U}_\eta \delta} = \mathsf{KU_C}! \circ r_{\mathsf{U}_\eta \delta}$ is equivalent to $\mathsf{RKUU_D}!' \circ l^\flat_{\mathsf{U}_\eta \delta} = \mathsf{RKUU_D}!' \circ r^\flat_{\mathsf{U}_\eta \delta}$.



For, if $\iota : \mathsf{U}_\eta \langle F', \zeta' \rangle \to \langle F, \zeta \rangle$ denotes the unique homomorphism resulting from the finality of $\langle F, \zeta \rangle$, then the following holds:

$$\begin{aligned}
\mathsf{KU_C}! \circ l_{\mathsf{U}_\eta\delta} = \mathsf{KU_C}! \circ r_{\mathsf{U}_\eta\delta} &\Leftrightarrow (! = \iota \circ \mathsf{U}_\eta!',\ \mathsf{U_C U}_\eta = \mathsf{UU_D}) \\
\mathsf{KU_C}\iota \circ \mathsf{KUU_D}!' \circ l_{\mathsf{U}_\eta\delta} = \mathsf{KU_C}\iota \circ \mathsf{KUU_D}!' \circ r_{\mathsf{U}_\eta\delta} &\Rightarrow (\mathsf{U_C}\iota \text{ is mono, K preserves monos}) \\
\mathsf{KUU_D}!' \circ l_{\mathsf{U}_\eta\delta} = \mathsf{KUU_D}!' \circ r_{\mathsf{U}_\eta\delta} &\Leftrightarrow (\text{property of adjunction}) \\
\mathsf{RKUU_D}!' \circ l^\flat_{\mathsf{U}_\eta\delta} = \mathsf{RKUU_D}!' \circ r^\flat_{\mathsf{U}_\eta\delta}
\end{aligned}$$

Proposition* 3.1.55. *Let* $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ *denote a horizontal abstract cosignature morphism, let* $\langle D, \delta \rangle$ *denote a* (D, G)*-coalgebra, and let* e *denote a* (C, F)*-coequation. Then,* $\mathsf{U}_\eta \langle D, \delta \rangle \models^\flat_{(\mathsf{C},\mathsf{F})} e$ *if and only if* $\langle D, \delta \rangle \models^\flat_{(\mathsf{D},\mathsf{G})} \eta(e)$.

Proof. Say e is of form (K, l, r). The conclusion then follows from:

$$\begin{aligned}
\mathsf{U}_\eta \langle D, \delta \rangle \models^\flat_{(\mathsf{C},\mathsf{F})} e &\Leftrightarrow (\text{definition of } \models^\flat) \\
\mathsf{KU_C}! \circ l_{\mathsf{U}_\eta\delta} = \mathsf{KU_C}! \circ r_{\mathsf{U}_\eta\delta} &\Leftrightarrow (\text{Remark 3.1.54}) \\
\mathsf{RKUU_D}!' \circ l^\flat_{\mathsf{U}_\eta\delta} = \mathsf{RKUU_D}!' \circ r^\flat_{\mathsf{U}_\eta\delta} &\Leftrightarrow (\text{definition of } \models^\flat) \\
\langle D, \delta \rangle \models^\flat_{(\mathsf{C},\mathsf{F})} e
\end{aligned}$$

□

Theorem* 3.1.56. $(\mathsf{HCosign}, \mathsf{Coalg}\restriction_{\mathsf{HCosign}}, \mathsf{Coeqn}\restriction_{\mathsf{HCosign}}, \models^\flat)$ *is an institution.*

Example 3.1.57 (**Destructor Hidden Signature Morphisms).** A hidden signature map $\phi : (H, \Sigma) \to (H', \Sigma')$ defines a *hidden signature morphism* [GD94] if and only if whenever $\sigma' \in \Sigma'_{w',s'}$ and some sort appearing in w' is of form $\phi(h)$ for some $h \in H$, it follows that $\sigma' = \phi(\sigma)$ for some $\sigma \in \Sigma_{w,s}$, with h appearing in w[11]. Now, if $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$ and $(\mathsf{Set}^{S'}_D, \mathsf{F}_{\Sigma'})$ denote the abstract cosignatures induced by the destructor hidden signatures (H, Σ) and (H', Σ') (see Example 3.1.6), and if $\phi : (H, \Sigma) \to (H', \Sigma')$ denotes a hidden signature morphism, then the abstract cosignature morphism (U, η_ϕ) induced by ϕ (see Example 3.1.52) is horizontal. (In this case, all the components of the natural transformation η_ϕ are monomorphisms.)

3.1.4 Compositionality Results

This section is devoted to the formulation of two compositionality results, allowing coalgebraic specifications and their models to be combined in a canonical way.

We begin by noting that any functor $\mathsf{U} : \mathsf{D} \to \mathsf{C}$ satisfying the conditions in the definition of abstract cosignature morphisms induces a lifting of abstract cosignatures over C to abstract cosignatures over D. Specifically, the lifting of an abstract cosignature (C, F) along U is the abstract cosignature $(\mathsf{D}, \bar{\mathsf{F}})$, with $\bar{\mathsf{F}} = \mathsf{RFU}$. (The preservation of pullbacks and of limits of ω^{op}-chains by $\bar{\mathsf{F}}$ follows from their preservation by each of U, F and R.) Moreover, the counit $\epsilon : \mathsf{UR} \Rightarrow \mathsf{Id}_\mathsf{C}$ of the adjunction $\mathsf{U} \dashv \mathsf{R}$ induces a cosignature morphism $(\mathsf{U}, \epsilon_{\mathsf{FU}}) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \bar{\mathsf{F}})$. Then, an arbitrary cosignature morphism $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ determines an abstract cosignature morphism $(\mathsf{Id}_\mathsf{D}, \bar{\eta}) : (\mathsf{D}, \bar{\mathsf{F}}) \to (\mathsf{D}, \mathsf{G})$, with $\bar{\eta} : \mathsf{G} \Rightarrow \bar{\mathsf{F}}$ denoting the natural transformation whose components are given by $\bar{\eta}_D = \eta^\flat_D$ for $D \in |\mathsf{D}|$, such that the following holds: $(\mathsf{Id}_\mathsf{D}, \bar{\eta}) \circ (\mathsf{U}, \epsilon_{\mathsf{FU}}) = (\mathsf{U}, \eta)$. This observation will allow any finite diagram in SCosign to be lifted to a diagram all of whose cosignatures have the same underlying category.

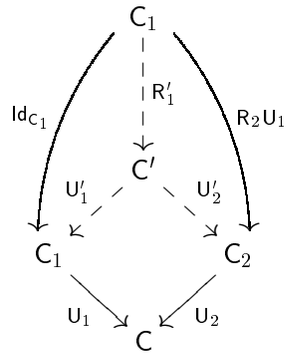
Theorem 3.1.58. SCosign *is finitely cocomplete.*

Proof. An initial object in SCosign is given by the abstract cosignature $(1, \mathsf{Id}_1)$, with 1 denoting a category with one object and one arrow. Given an arbitrary cosignature (C, F), the cosignature morphism $(\mathsf{U}, \eta) : (1, \mathsf{Id}_1) \to (\mathsf{C}, \mathsf{F})$, with $\mathsf{U} : \mathsf{C} \to 1$ denoting the only functor from C to 1, and with $\eta : \mathsf{UF} \Rightarrow \mathsf{U}$ denoting the natural transformation all of whose components are identities is the only cosignature morphism from $(1, \mathsf{Id}_1)$ to (C, F).

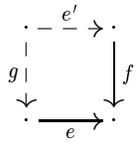
The pushout in SCosign of $(\mathsf{U}_1, \eta_1) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}_1, \mathsf{F}_1)$ and $(\mathsf{U}_2, \eta_2) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}_2, \mathsf{F}_2)$ is computed as follows. First, let $\mathsf{U}'_1 : \mathsf{C}' \to \mathsf{C}_1$, $\mathsf{U}'_2 : \mathsf{C}' \to \mathsf{C}_2$ define the pullback of U_1, U_2 in CAT (see Example 2.1.32 and Remark 2.1.33), and let $\mathsf{R}'_1 : \mathsf{C}_1 \to \mathsf{C}'$, $\mathsf{R}'_2 : \mathsf{C}_2 \to \mathsf{C}'$ denote the only functors satisfying $\mathsf{U}'_1\mathsf{R}'_1 = \mathsf{Id}_{\mathsf{C}_1}$, $\mathsf{U}'_2\mathsf{R}'_1 = \mathsf{R}_2\mathsf{U}_1$ (resulting from the fact that $\mathsf{U}_1\mathsf{Id}_{\mathsf{C}_1} = \mathsf{U}_1 = \mathsf{U}_2\mathsf{R}_2\mathsf{U}_1$), and

[11] The condition used here is slightly stronger than the one given in [GD94]. This is actually needed in order for the notion of hidden signature morphism to yield an institution.

similarly for R'_2.



Then, R'_1 and R'_2 define right adjoints, right inverses to U'_1 and respectively U'_2. (This follows from the construction of pullbacks in CAT. For, any C_1-arrow $f : \mathsf{U}'_1 C' \to C_1$ determines a C'-arrow $f' : C' \to \mathsf{R}'_1 C_1$, with f' being the only C'-arrow satisfying $\mathsf{U}'_1 f' = f$ and $\mathsf{U}'_2 f' = (\mathsf{U}_1 f)^\flat : \mathsf{U}'_2 C' \to \mathsf{R}_2\mathsf{U}_1 C_1$, resulting from the fact that $\mathsf{U}_1 f = \mathsf{U}_2(\mathsf{U}_1 f)^\flat$.) Moreover, $\mathsf{R}'_1\mathsf{R}_1 = \mathsf{R}'_2\mathsf{R}_2$. This follows from $\mathsf{U}'_i\mathsf{R}'_1\mathsf{R}_1 = \mathsf{U}'_i\mathsf{R}'_2\mathsf{R}_2$ for $i = 1, 2$, together with the couniversality of $(\mathsf{U}'_1, \mathsf{U}'_2)$. Also, the existence of limits and colimits in C, C_1 and C_2 together with the lifting of limits and colimits by U_1 and U_2 result in the existence of pullbacks and of limits of ω^{op}-chains in C' and in their lifting by U'_1 and U'_2 (see Proposition 2.1.39 and Remark 2.1.40). Finally, the regularity of each of C, C_1 and C_2 together with the preservation of pullbacks and coequalisers by U'_1, U'_2 and $\mathsf{U}_1\mathsf{U}'_1 = \mathsf{U}_2\mathsf{U}'_2$ result in C' also being regular. For, given the following pullback diagram in C':

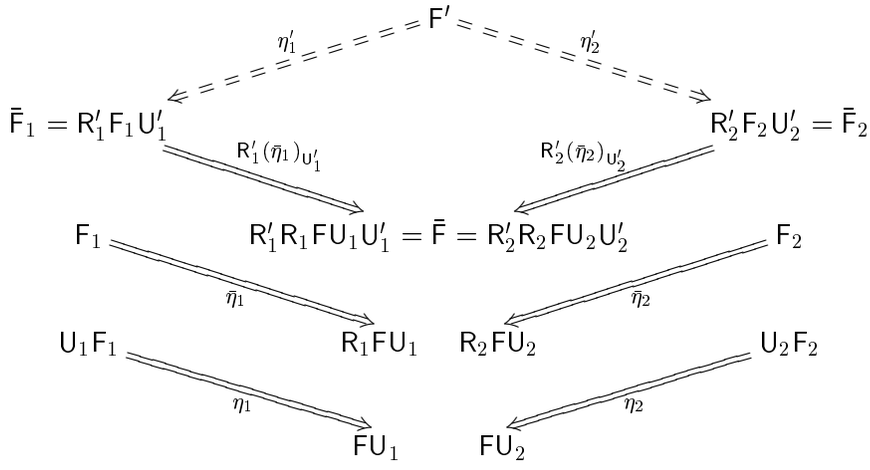


with e a regular epimorphism, the fact that e is a coequaliser of its kernel pair (see 7 of Proposition 2.1.43) together with the preservation of coequalisers by U'_1, U'_2 and $\mathsf{U}_1\mathsf{U}'_1$ result in $\mathsf{U}'_1 e$, $\mathsf{U}'_2 e$ and $\mathsf{U}_1\mathsf{U}'_1 e$ being coequalisers, and hence regular epimorphisms. The regularity of C_1, C_2 and C together with the preservation of pullbacks by U'_1, U'_2 and $\mathsf{U}_1\mathsf{U}'_1$ then result in $\mathsf{U}'_1 e'$, $\mathsf{U}'_2 e'$ and $\mathsf{U}_1\mathsf{U}'_1 e'$ being themselves regular epimorphisms, and hence (as C_1, C_2 and C have kernel pairs) coequalisers of their kernel pairs. The preservation of kernel pairs by U'_1, U'_2 and $\mathsf{U}_1\mathsf{U}'_1$ together with the construction of coequalisers in C' from suitably-chosen coequalisers in C_1, C_2 and C then results in e' defining a coequaliser of its kernel pair, and hence a regular epimorphism. This concludes the proof of the regularity of C'.

Now let $\bar{\mathsf{F}}, \bar{\mathsf{F}}_1, \bar{\mathsf{F}}_2 : \mathsf{C}' \to \mathsf{C}'$ denote the liftings of F, F_1 and F_2 along $\mathsf{U} = \mathsf{U}_1\mathsf{U}'_1 = \mathsf{U}_2\mathsf{U}'_2$, U'_1 and U'_2 respectively. Finally, let $\eta'_1 : \mathsf{F}' \Rightarrow \bar{\mathsf{F}}_1$, $\eta'_2 : \mathsf{F}' \Rightarrow \bar{\mathsf{F}}_2$ denote the pullback of $\mathsf{R}'_1(\bar{\eta}_1)_{\mathsf{U}'_1} : \bar{\mathsf{F}}_1 \Rightarrow \bar{\mathsf{F}}$ and $\mathsf{R}'_2(\bar{\eta}_2)_{\mathsf{U}'_2} : \bar{\mathsf{F}}_2 \Rightarrow \bar{\mathsf{F}}$ in (the quasi-category) $[\mathsf{C}', \mathsf{C}']$ (see Proposition 2.1.41 and Remark 2.1.42). Note that $\mathsf{U}'_i\eta'_i$ has domain $\mathsf{U}'_i\mathsf{F}'$ and codomain $\mathsf{U}'_i\mathsf{R}'_i\mathsf{F}_i\mathsf{U}'_i = \mathsf{F}_i\mathsf{U}'_i$, for $i = 1, 2$.

The preservation of pullbacks and of limits of ω^{op}-chains by each of $\bar{\mathsf{F}}$, $\bar{\mathsf{F}}_1$ and $\bar{\mathsf{F}}_2$ results in their

preservation by F' (see Proposition 2.1.41).



The pushout of (U_1, η_1) and (U_2, η_2) is then given by $(\mathsf{U}'_1, \mathsf{U}'_1\eta'_1) : (\mathsf{C}_1, \mathsf{F}_1) \to (\mathsf{C}', \mathsf{F}')$ and $(\mathsf{U}'_2, \mathsf{U}'_2\eta'_2) : (\mathsf{C}_2, \mathsf{F}_2) \to (\mathsf{C}', \mathsf{F}')$. The universality of $((\mathsf{U}'_1, \mathsf{U}'_1\eta'_1), (\mathsf{U}'_2, \mathsf{U}'_2\eta'_2))$ is a consequence of the couniversality of $(\mathsf{U}'_1, \mathsf{U}'_2)$ together with the couniversality of each of $\bar{\eta}_1$, $\bar{\eta}_2$ and the couniversality of (η'_1, η'_2).

The conclusion now follows by the dual of Proposition 2.1.30. □

Remark 3.1.59. Theorem 3.1.58 and Corollary 2.2.6 result in the subcategories of the categories of specifications associated to the institutions $(\mathsf{Cosign}, \mathsf{Coalg}, \mathsf{Coeqn}, \models)$ and $(\mathsf{HCosign}, \mathsf{Coalg}\restriction_{\mathsf{HCosign}}, \mathsf{Coeqn}\restriction_{\mathsf{HCosign}}, \models^{\mathsf{b}})$ (see Theorems 3.1.49 and 3.1.56) whose specification morphisms are underlain by strong abstract cosignature morphisms also being finitely cocomplete.

The construction of pushouts in SCosign also yields the following result.

Theorem 3.1.60. *The functor* $\mathsf{Coalg}\restriction_{\mathsf{SCosign}} : \mathsf{SCosign} \to \mathsf{CAT}^{\mathsf{op}}$ *preserves finite colimits.*

Proof. It suffices to prove that $\mathsf{Coalg}\restriction_{\mathsf{SCosign}}$ preserves the initial object and pushouts. First, the category of coalgebras of the cosignature $(1, \mathsf{Id}_1)$ (see the proof of Theorem 3.1.58) has one object and one arrow, and therefore is a final object in CAT. Also, given the pushout diagram in SCosign obtained in the proof of Theorem 3.1.58, the fact that its image under $\mathsf{Coalg}\restriction_{\mathsf{SCosign}}$ defines a pullback in CAT follows from any span $\langle \mathsf{G}_1, \mathsf{G}_2 \rangle$ with source D on $\mathsf{Coalg}(\mathsf{C}_1, \mathsf{F}_1), \mathsf{Coalg}(\mathsf{C}_2, \mathsf{F}_2)$ in CAT, additionally satisfying $\mathsf{U}_{\eta_1}\mathsf{G}_1 = \mathsf{U}_{\eta_2}\mathsf{G}_2$, inducing a unique functor $\mathsf{G} : \mathsf{D} \to \mathsf{Coalg}(\mathsf{C}', \mathsf{F}')$ satisfying $\mathsf{U}_{\mathsf{U}'_i\eta'_i}\mathsf{G} = \mathsf{G}_i$ for $i = 1, 2$. Specifically, for $D \in |\mathsf{D}|$, $\mathsf{G}D$ is the unique $(\mathsf{C}', \mathsf{F}')$-coalgebra induced by the images of D under G_1, G_2 and $\mathsf{U}_{\eta_1}\mathsf{G}_1 = \mathsf{U}_{\eta_2}\mathsf{G}_2$ respectively. Say $\mathsf{G}_i D = \langle C_i, \gamma_i \rangle$ for $i = 1, 2$. Then, $\mathsf{U}_{\eta_1}\mathsf{G}_1 = \mathsf{U}_{\eta_2}\mathsf{G}_2$ immediately gives $\mathsf{U}_1 C_1 = \mathsf{U}_2 C_2 = C$ and $\mathsf{U}_1((\bar{\eta}_1)_{C_1} \circ \gamma_1) = \mathsf{U}_2((\bar{\eta}_2)_{C_2} \circ \gamma_2)$ (where $(\bar{\eta}_i)_{C_i} \circ \gamma_i : C_i \to \mathsf{R}_i\mathsf{F}C$ for $i = 1, 2$). The construction of pullbacks in CAT (see the proof of Proposition 2.1.39 and Remark 2.1.40) now yields a unique C'-arrow $c' : C' \to \mathsf{R}'_1\mathsf{R}_1\mathsf{F}C = \mathsf{R}'_2\mathsf{R}_2\mathsf{F}C$ satisfying $\mathsf{U}'_i c' = (\bar{\eta}_i)_{C_i} \circ \gamma_i$ for $i = 1, 2$. Furthermore, $c' = \mathsf{R}'_i(\bar{\eta}_i)_{C_i} \circ \gamma_i^\flat$ for $i = 1, 2$. This follows from R'_i being right adjoint, right inverse to U'_i together with $\mathsf{U}'_i(\mathsf{R}'_i(\bar{\eta}_i)_{C_i} \circ \gamma_i^\flat) = (\bar{\eta}_i)_{C_i} \circ \gamma_i = \mathsf{U}'_i c'$, for $i = 1, 2$. The definition of η'_1, η'_2 together with $\mathsf{R}'_1(\bar{\eta}_1)_{C_1} \circ \gamma_1^\flat = \mathsf{R}'_2(\bar{\eta}_2)_{C_2} \circ \gamma_2^\flat$ now yields a unique

C′-arrow $\gamma' : C' \to \mathsf{F}'C'$ satisfying $(\eta'_i)_{C'} \circ \gamma' = \gamma_i^\flat$, and hence $\mathsf{U}'_i(\eta'_i)_{C'} \circ \mathsf{U}'_i\gamma' = \gamma_i$ for $i = 1, 2$ (with $\eta'_1 : \mathsf{F}' \Rightarrow \bar{\mathsf{F}}_1$ and $\eta'_2 : \mathsf{F}' \Rightarrow \bar{\mathsf{F}}_2$ being as in the proof of Theorem 3.1.58). That is, $\mathsf{U}_{\mathsf{U}'_i\eta'_i}\langle C', \gamma'\rangle = \langle C_i, \gamma_i\rangle$ for $i = 1, 2$. (The uniqueness of γ' follows from the couniversality of $(\mathsf{U}'_1, \mathsf{U}'_2)$ together with the couniversality of (η'_1, η'_2).) Moreover, the couniversality of (η'_1, η'_2) can be used to show that the correspondence $D \mapsto \langle C', \gamma'\rangle$ is functorial. For, given $d : D \to E$ in D, with $D \mapsto \langle C', \gamma'\rangle$ and $E \mapsto \langle F', \varphi'\rangle$, the fact that $\mathsf{U}_{\eta_1}\mathsf{G}_1 d = \mathsf{U}_{\eta_2}\mathsf{G}_2 d$ (and hence $\mathsf{U}_1\mathsf{U}_{\mathsf{C}_1}\mathsf{G}_1 d = \mathsf{U}_2\mathsf{U}_{\mathsf{C}_2}\mathsf{G}_2 d$) yields a unique C′-arrow $f : C' \to F'$ satisfying $\mathsf{U}'_i f = \mathsf{U}_{\mathsf{C}_i}\mathsf{G}_i d$ for $i = 1, 2$. Moreover, f defines a $(\mathsf{C}', \bar{\mathsf{F}}_1)$- as well as a $(\mathsf{C}', \bar{\mathsf{F}}_2)$-coalgebra homomorphism (as $\mathsf{U}'_1 f$ and $\mathsf{U}'_2 f$ define $(\mathsf{C}_1, \mathsf{F}_1)$- and respectively $(\mathsf{C}_2, \mathsf{F}_2)$-coalgebra homomorphisms). Then, $\varphi' \circ f = \mathsf{F}'f \circ \gamma'$. This follows from the couniversality of (η'_1, η'_2), together with:

$$\begin{aligned}
(\eta'_i)_{F'} \circ \mathsf{F}'f \circ \gamma' = &\quad \text{(naturality of } \eta'_i\text{)} \\
\bar{\mathsf{F}}_i f \circ (\eta'_i)_{C'} \circ \gamma' = &\quad \text{(f defines a } (\mathsf{C}', \bar{\mathsf{F}}_i)\text{-coalgebra homomorphism)} \\
(\eta'_i)_{F'} \circ \varphi' \circ f &
\end{aligned}$$

for $i = 1, 2$. That is, f defines a $(\mathsf{C}', \mathsf{F}')$-coalgebra homomorphism from $\langle C', \gamma'\rangle$ to $\langle F', \varphi'\rangle$. This concludes the proof. □

3.1.5 Cofree Coalgebras

This section investigates the existence of cofree constructions along abstract cosignature morphisms (i.e. the existence of right adjoints to the reduct functors induced by such cosignature morphisms). In particular, a generalisation of a result in [Rut96] regarding the existence of cofree coalgebras is formulated, and this generalisation is used to obtain a cofreeness result for equational coalgebraic specification. (The generalisation involves endofunctors on arbitrary categories, and accounts for a possible change in their underlying category when moving from one category of coalgebras to another.)

Remark 3.1.5 immediately results in the existence of cofree coalgebras w.r.t. the functor $\mathsf{U}_\mathsf{C} : \mathsf{Coalg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$[12].

Proposition* 3.1.61. *Let* (C, F) *denote an abstract cosignature. Then,* U_C *has a right adjoint.*

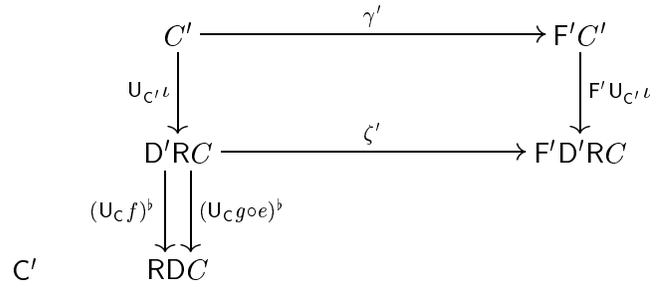
The constraints in the definitions of abstract cosignatures (in particular, the preservation of pullbacks by the endofunctors involved) and respectively abstract cosignature morphisms (in particular, the existence of right adjoints to the functors between the underlying categories) will now be used to prove the existence of cofree coalgebras w.r.t. the reduct functors induced by abstract cosignature morphisms.

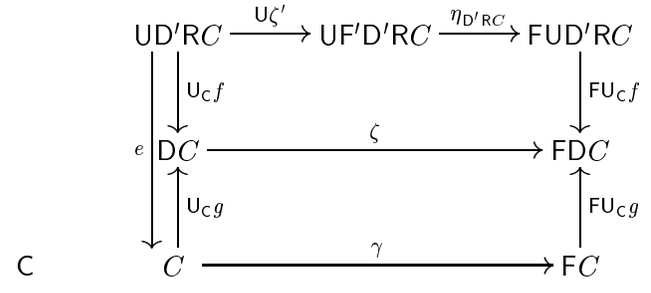
Theorem 3.1.62. *Let* $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}', \mathsf{F}')$ *denote an abstract cosignature morphism. Then,* $\mathsf{U}_\eta : \mathsf{Coalg}(\mathsf{C}', \mathsf{F}') \to \mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ *has a right adjoint.*

Proof. Let R denote a right adjoint to U, let $\eta_0 : \mathsf{Id}_{\mathsf{C}'} \Rightarrow \mathsf{RU}$ and $\epsilon_0 : \mathsf{UR} \Rightarrow \mathsf{Id}_\mathsf{C}$ denote the unit and respectively the counit of the adjunction $\mathsf{U} \dashv \mathsf{R}$, and let $(\mathsf{D}, \epsilon, \delta)$ and $(\mathsf{D}', \epsilon', \delta')$ denote the comonads induced by the abstract cosignatures (C, F) and respectively $(\mathsf{C}', \mathsf{F}')$.

[12] See also Remark 2.3.47 and Proposition 2.3.50.

Now let $\langle C, \gamma\rangle$ denote a (C, F)-coalgebra. The construction of a (C′, F′)-coalgebra $\langle C', \gamma'\rangle$ cofree over $\langle C, \gamma\rangle$ w.r.t. U_η is as follows. Consider the cofree (C, F)-coalgebra $\langle \mathsf{D}C, \zeta\rangle$ over C, as well as the cofree (C′, F′)-coalgebra $\langle \mathsf{D}'\mathsf{R}C, \zeta'\rangle$ over $\mathsf{R}C$. Then, the C-arrows $e = \epsilon_{0,C} \circ \mathsf{U}\epsilon'_{\mathsf{R}C} : \mathsf{U}\mathsf{D}'\mathsf{R}C \to C$ and $1_C : C \to C$ extend to (C, F)-coalgebra homomorphisms $f : \mathsf{U}_\eta\langle \mathsf{D}'\mathsf{R}C, \zeta'\rangle \to \langle \mathsf{D}C, \zeta\rangle$ and respectively $g : \langle C, \gamma\rangle \to \langle \mathsf{D}C, \zeta\rangle$, satisfying $\epsilon_C \circ \mathsf{U}_\mathsf{C} f = e$ and respectively $\epsilon_C \circ \mathsf{U}_\mathsf{C} g = 1_C$. Moreover, the second equality results in $\mathsf{U}_\mathsf{C} g$ being a monomorphism.





Now let $\iota : \langle C', \gamma'\rangle \to \langle \mathsf{D}'\mathsf{R}C, \zeta'\rangle$ denote the equaliser of $((\mathsf{U}_\mathsf{C} f)^\flat)^\flat$ and $((\mathsf{U}_\mathsf{C} g \circ e)^\flat)^\flat$ in Coalg(C′, F′)[13] (see Corollary 3.1.13). Then, $e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota$ defines a (C, F)-coalgebra homomorphism $c : \mathsf{U}_\eta\langle C', \gamma'\rangle \to \langle C, \gamma\rangle$. This follows from:

$$
\begin{aligned}
\mathsf{F}\mathsf{U}_\mathsf{C} g \circ \gamma \circ e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota = \quad & (g \text{ is a (C, F)-coalgebra homomorphism}) \\
\zeta \circ \mathsf{U}_\mathsf{C} g \circ e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota = \quad & (\text{definition of } \iota) \\
\zeta \circ \mathsf{U}_\mathsf{C} f \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota = \quad & (f \text{ is a (C, F)-coalgebra homomorphism}) \\
\mathsf{F}\mathsf{U}_\mathsf{C} f \circ \eta_{\mathsf{D}'\mathsf{R}C} \circ \mathsf{U}\zeta' \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota = \quad & (\iota \text{ is a (C, F)-coalgebra homomorphism}) \\
\mathsf{F}\mathsf{U}_\mathsf{C} f \circ \eta_{\mathsf{D}'\mathsf{R}C} \circ \mathsf{U}\mathsf{F}'\mathsf{U}_{\mathsf{C}'}\iota \circ \mathsf{U}\gamma' = \quad & (\text{naturality of } \eta) \\
\mathsf{F}\mathsf{U}_\mathsf{C} f \circ \mathsf{F}\mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota \circ \eta_{C'} \circ \mathsf{U}\gamma' = \quad & (\text{definition of } \iota) \\
\mathsf{F}\mathsf{U}_\mathsf{C} g \circ \mathsf{F}e \circ \mathsf{F}\mathsf{U}\mathsf{U}_{\mathsf{C}'}\iota \circ \eta_{C'} \circ \mathsf{U}\gamma' &
\end{aligned}
$$

together with $\mathsf{F}\mathsf{U}_\mathsf{C} g$ being a monomorphism (as $\mathsf{U}_\mathsf{C} g$ is a monomorphism, while F preserves monomorphisms).

Moreover, $\langle C', \gamma'\rangle$ is cofree over $\langle C, \gamma\rangle$ w.r.t. U_η. For, given an arbitrary (C′, F′)-coalgebra $\langle D, \delta\rangle$ together with a (C, F)-coalgebra homomorphism $k : \mathsf{U}_\eta\langle D, \delta\rangle \to \langle C, \gamma\rangle$, the fact that R is a right adjoint to U yields a C′-arrow $(\mathsf{U}_\mathsf{C} k)^\flat : D \to \mathsf{R}C$ satisfying $\epsilon_{0,C} \circ \mathsf{U}(\mathsf{U}_\mathsf{C} k)^\flat = \mathsf{U}_\mathsf{C} k$. This, in turn,

[13] The inner $^\flat$ refers to the adjunction between C and C′, whereas the outer $^\flat$ refers to the adjunction between C′ and Coalg(C′, F′).

yields a $(\mathsf{C}', \mathsf{F}')$-coalgebra homomorphism $l : \langle D, \delta\rangle \to \langle \mathsf{D}'\mathsf{R}C, \zeta'\rangle$ satisfying $\epsilon'_{\mathsf{R}C} \circ \mathsf{U}_{\mathsf{C}'} l = (\mathsf{U}_{\mathsf{C}} k)^\flat$. Moreover, $(\mathsf{U}_{\mathsf{C}} f)^\flat \circ \mathsf{U}_{\mathsf{C}'} l = (\mathsf{U}_{\mathsf{C}} g \circ e)^\flat \circ \mathsf{U}_{\mathsf{C}'} l$. This follows from:

$$\begin{aligned}
\epsilon_{0,C} \circ \mathsf{U}((\mathsf{U}_{\mathsf{C}} f)^\flat \circ \mathsf{U}_{\mathsf{C}'} l) = &\quad \text{(definition of } (\mathsf{U}_{\mathsf{C}} f)^\flat\text{)} \\
\mathsf{U}_{\mathsf{C}} f \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad (\mathsf{U}\mathsf{U}_{\mathsf{C}'} = \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta) \\
\mathsf{U}_{\mathsf{C}} f \circ \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta l = &\quad (f \circ \mathsf{U}_\eta l = g \circ k) \\
\mathsf{U}_{\mathsf{C}} g \circ \mathsf{U}_{\mathsf{C}} k = &\quad \text{(definition of } (\mathsf{U}_{\mathsf{C}} k)^\flat\text{)} \\
\mathsf{U}_{\mathsf{C}} g \circ \epsilon_{0,C} \circ \mathsf{U}(\mathsf{U}_{\mathsf{C}} k)^\flat = &\quad \text{(definition of } l\text{)} \\
\mathsf{U}_{\mathsf{C}} g \circ \epsilon_{0,C} \circ \mathsf{U}\epsilon'_{\mathsf{R}C} \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } e\text{)} \\
\mathsf{U}_{\mathsf{C}} g \circ e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } (\mathsf{U}_{\mathsf{C}} g \circ e)^\flat\text{)} \\
\epsilon_{0,C} \circ \mathsf{U}((\mathsf{U}_{\mathsf{C}} g \circ e)^\flat \circ \mathsf{U}_{\mathsf{C}'} l) &
\end{aligned}$$

together with R being a right adjoint to U, where the fact that $f \circ \mathsf{U}_\eta l = g \circ k$ follows from the uniqueness of (C, F)-coalgebra homomorphisms $n : \mathsf{U}_\eta\langle D, \delta\rangle \to \langle \mathsf{D}C, \zeta\rangle$ satisfying $\epsilon_C \circ \mathsf{U}_{\mathsf{C}} n = m$ with $m : \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta\langle D, \delta\rangle \to C$ a C-arrow, together with:

$$\begin{aligned}
\epsilon_C \circ \mathsf{U}_{\mathsf{C}} f \circ \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta l = &\quad \text{(definition of } f\text{)} \\
e \circ \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta l = &\quad (\mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta = \mathsf{U}\mathsf{U}_{\mathsf{C}'}) \\
e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } e\text{)} \\
\epsilon_{0,C} \circ \mathsf{U}\epsilon'_{\mathsf{R}C} \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } l\text{)} \\
\epsilon_{0,C} \circ \mathsf{U}(\mathsf{U}_{\mathsf{C}} k)^\flat = &\quad \text{(definition of } (\mathsf{U}_{\mathsf{C}} k)^\flat\text{)} \\
\mathsf{U}_{\mathsf{C}} k = &\quad \text{(definition of } g\text{)} \\
\epsilon_C \circ \mathsf{U}_{\mathsf{C}} g \circ \mathsf{U}_{\mathsf{C}} k &
\end{aligned}$$

Then, couniversality of ι yields a $(\mathsf{C}', \mathsf{F}')$-coalgebra homomorphism $k^\flat : \langle D, \delta\rangle \to \langle C', \gamma'\rangle$ satisfying $\iota \circ k^\flat = l$. Moreover, $c \circ \mathsf{U}_\eta k^\flat = k$. This follows from each of c, $\mathsf{U}_\eta k^\flat$ and k being (C, F)-coalgebra homomorphisms, together with:

$$\begin{aligned}
\mathsf{U}_{\mathsf{C}} c \circ \mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta k^\flat = &\quad (\mathsf{U}_{\mathsf{C}}\mathsf{U}_\eta = \mathsf{U}\mathsf{U}_{\mathsf{C}'}) \\
\mathsf{U}_{\mathsf{C}} c \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} k^\flat = &\quad \text{(definition of } c\text{)} \\
e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} \iota \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} k^\flat = &\quad \text{(definition of } k^\flat\text{)} \\
e \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } e\text{)} \\
\epsilon_{0,C} \circ \mathsf{U}\epsilon'_{\mathsf{R}C} \circ \mathsf{U}\mathsf{U}_{\mathsf{C}'} l = &\quad \text{(definition of } l\text{)} \\
\epsilon_{0,C} \circ \mathsf{U}(\mathsf{U}_{\mathsf{C}} k)^\flat = &\quad \text{(definition of } (\mathsf{U}_{\mathsf{C}} k)^\flat\text{)} \\
\mathsf{U}_{\mathsf{C}} k &
\end{aligned}$$

Finally, the uniqueness of a $(\mathsf{C}', \mathsf{F}')$-coalgebra homomorphism $k^\flat$ satisfying $c \circ \mathsf{U}_\eta k^\flat = k$ follows from the cofreeness of $\langle \mathsf{D}'\mathsf{R}C, \zeta'\rangle$ together with the couniversality of ι. □

The induced right adjoint to the reduct functor U_η provides a canonical way of constructing (most general) $(\mathsf{C}', \mathsf{F}')$-coalgebras over given (C, F)-coalgebras. This makes the cofree construction a suitable denotation for abstract cosignature morphisms.

Remark 3.1.63. In [Rut96], a result similar to Theorem 3.1.62 is proved for coalgebras of endofunctors on Set. There, endofunctors $\mathsf{T}, \mathsf{S} : \mathsf{Set} \to \mathsf{Set}$ together with reduct functors $\mathsf{U}_\eta : \mathsf{Coalg}(\mathsf{S}) \to \mathsf{Coalg}(\mathsf{T})$ induced by natural transformations $\eta : \mathsf{S} \Rightarrow \mathsf{T}$ are considered, and the existence of cofree coalgebras w.r.t. U_η is proved under the assumption that, for any set C, the endofunctor $\mathsf{S} \times C : \mathsf{Set} \to \mathsf{Set}$ (taking a set X to the set $\mathsf{S}X \times C$) has a final coalgebra. Theorem 3.1.62 replaces this assumption with the stronger one involving the preservation of limits of ω^{op}-chains by the endofunctors defining abstract cosignatures. In this respect, our result is less general than the one in [Rut96]. On the other hand, the result here is formulated for endofunctors on arbitrary, possibly distinct categories, thus being more abstract than the one in [Rut96].

If attention is restricted to coalgebraic specifications involving enumerable sets of coequations, then Theorem 3.1.62 generalises to specification morphisms.

Theorem 3.1.64. *Let* $(\mathsf{U}, \eta) : (\mathsf{C}, \mathsf{F}, E) \to (\mathsf{D}, \mathsf{G}, E')$ *denote an abstract coalgebraic specification morphism, such that* E' *is enumerable. Then,* $\mathsf{U}_\eta{\restriction}_{\mathsf{Coalg}(\mathsf{D},\mathsf{G},E')} : \mathsf{Coalg}(\mathsf{D}, \mathsf{G}, E') \to \mathsf{Coalg}(\mathsf{C}, \mathsf{F}, E)$ *has a right adjoint.*

Proof. Let $\langle C, \gamma\rangle$ denote a $(\mathsf{C}, \mathsf{F}, E)$-coalgebra, and let $\epsilon_\gamma : \mathsf{U}_\eta\langle D, \delta\rangle \to \langle C, \gamma\rangle$ denote a couniversal arrow from U_η to $\langle C, \gamma\rangle$. Also, let $\langle\langle D', \delta'\rangle, d\rangle$ denote a final object in $(\mathsf{Coalg}(\mathsf{D}, \mathsf{G})/\delta)^{E'}$ (see Proposition 3.1.33). Then, $\epsilon_\gamma \circ \mathsf{U}_\eta d : \mathsf{U}_\eta\langle D', \delta'\rangle \to \langle C, \gamma\rangle$ defines a couniversal arrow from $\mathsf{U}_\eta{\restriction}_{\mathsf{Coalg}(\mathsf{D},\mathsf{G},E')}\ \delta'$ to γ. (Its couniversality is a consequence of the couniversality of ϵ_γ and of the finality of $\langle\langle D', \delta'\rangle, d\rangle$.) The correspondence $\langle C, \gamma\rangle \mapsto \langle D', \delta'\rangle$ extends to a right adjoint to $\mathsf{U}_\eta{\restriction}_{\mathsf{Coalg}(\mathsf{D},\mathsf{G},E')}$. □

3.1.6 Expressiveness

This section investigates the expressiveness of the coalgebraic framework previously introduced from the point of view of characterising covarieties by means of coequations.

The starting point in this investigation is a Birkhoff-style characterisability result in [Kur00], stating that covarieties are definable by means of *modal formulae*. The notion of covariety used in [Kur00] is more general than the one used here, being defined relatively to a factorisation system $(\mathcal{E}, \mathcal{M})$ for the category of coalgebras in question. (Closure under homomorphisms belonging to $\mathcal{M}$ and $\mathcal{E}$ are used instead of closure under subcoalgebras and respectively homomorphic images.) An instantiation of the result in [Kur00] which provides a characterisability result for covarieties over an abstract cosignature (C, F) (as defined here) therefore has to consider a factorisation system $(\mathcal{E}, \mathcal{M})$ for $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$ whose $\mathcal{E}$-arrows are precisely the homomorphic images, and whose $\mathcal{M}$-arrows are precisely the subcoalgebras. Such an instantiation is possible due to the following result.

Proposition 3.1.65. *Let* (C, F) *denote an abstract cosignature, with* $(\mathsf{Epi}(\mathsf{C}), \mathsf{Mono}(\mathsf{C}))$ *defining a factorisation system for* C. *Then,* $(\mathsf{U}_\mathsf{C}^{-1}(\mathsf{Epi}(\mathsf{C})), \mathsf{U}_\mathsf{C}^{-1}(\mathsf{Mono}(\mathsf{C})))$ *defines a factorisation system for* $\mathsf{Coalg}(\mathsf{C}, \mathsf{F})$.

Proof. The fact that F preserves monomorphisms (see Definition 3.1.1) results in the functor

U_C : Coalg(C, F) → C *creating factorisations w.r.t.* (Epi(C), Mono(C))[14]. This follows by [Kur00, Proposition 1.3.3]. The conclusion then follows by [Kur00, Theorem 1.3.7]. □

The *modal formulae* [Kur00] associated to an abstract cosignature (C, F) satisfying the hypothesis of Proposition 3.1.65 are given by (C, F)-subcoalgebras (i.e. U_C^{-1}(Mono(C))-homomorphisms) with U_C^{-1}(Mono(C))-injective codomains. Moreover, according to [Kur00, Proposition 2.2.10] (see also the remark following Proposition 2.2.10), the U_C^{-1}(Mono(C))-injective objects in Coalg(C, F) are retracts of cofree coalgebras. As a result, any modal formula over (C, F) is of form $\varphi : \langle S, \xi\rangle \to \langle D, \delta\rangle$, with φ defining a (C, F)-subcoalgebra, and with $\langle D, \delta\rangle$ denoting a (C, F)-subcoalgebra of a cofree (C, F)-coalgebra.

A modal formula $\varphi : \langle S, \xi\rangle \to \langle D, \delta\rangle$ *holds* [Kur00] in a (C, F)-coalgebra $\langle C, \gamma\rangle$ (written $\langle C, \gamma\rangle \models \varphi$) if and only if any (C, F)-coalgebra homomorphism $f : \langle C, \gamma\rangle \to \langle D, \delta\rangle$ factors through φ.

$$\begin{array}{ccc} & & \langle C,\gamma\rangle \\ & \swarrow & \downarrow f \\ \langle S,\xi\rangle & \xrightarrow[\varphi]{} & \langle D,\delta\rangle \end{array}$$

The following is a consequence of Proposition 3.1.65 and of [Kur00, Corollary 2.5.5].

Theorem 3.1.66. *Let* (C, F) *denote an abstract cosignature, such that* (Epi(C), Mono(C)) *defines a factorisation system for* C*, and such that* C *has enough injectives and is wellpowered*[15]*. Then, a full subcategory* $\mathcal{K}$ *of* Coalg(C, F) *is a covariety if and only if* $\mathcal{K}$ *is definable by a class of modal formulae.*

Proof. By Proposition 3.1.65, $(U_C^{-1}(\text{Epi(C)}), U_C^{-1}(\text{Mono(C)}))$ defines a factorisation system for Coalg(C, F). Also, the fact that C has enough injectives and is wellpowered results in Coalg(C, F) also having enough injectives[16] and being U_C^{-1}(Mono(C))-wellpowered. Finally, Definition 3.1.1 and Corollary 2.4.10 result in Coalg(C, F) having coproducts. The conclusion then follows by [Kur00, Corollary 2.5.5]. □

A closer analysis of the proof of [Kur00, Theorem 2.5.4] (used to derive [Kur00, Corollary 2.5.5]) reveals that, under the hypotheses of Theorem 3.1.66, modal formulae of form $\varphi : \langle S, \xi\rangle \to \langle D, \delta\rangle$ with φ a (C, F)-subalgebra and $\langle D, \delta\rangle$ a cofree (C, F)-coalgebra over an injective C-object are sufficient to characterise (C, F)-covarieties. In the following, modal formulae φ of this form will be shown to induce (C, F)-coequations (K, l, r) having the property that $\langle C, \gamma\rangle \models$ (K, l, r) if and only if $\langle C, \gamma\rangle \models \varphi$. This, in turn, will yield a characterisability result for covarieties in terms of coequations.

The construction of the (C, F)-coequation induced by a modal formula φ of the above form is as follows. Say $\langle D, \delta\rangle$ is cofree over Z w.r.t. U_C, with Z injective. Let $h, k : D \to E$ define a cokernel

[14] Given categories C and D together with a factorisation system $(\mathcal{E}, \mathcal{M})$ for D, a functor U : C → D *creates factorisations w.r.t.* $(\mathcal{E}, \mathcal{M})$ if and only if for any C-arrow f and any $(\mathcal{E}, \mathcal{M})$-factorisation of form $Uf = m \circ e$, there exists a unique factorisation of form $f = m' \circ e'$ with $Ue' = e$ and $Um' = m$.

[15] A category C is *$\mathcal{M}$-wellpowered* if and only if for any C-object C, there exists, up to isomorphism, only a set of $\mathcal{M}$-arrows with codomain C. C is *wellpowered* if it is Mono(C)-wellpowered.

[16] [Kur00, Proposition 2.2.10] is used to show this.

pair for $\mathsf{U_C}\varphi$, and let $\mathsf{K} : \mathsf{C} \to \mathsf{C}$ denote the functor given by:

$$\mathsf{K}A = \prod_{f:A\to Z} E$$

for $A \in |\mathsf{C}|$, and:

$$\mathsf{K}a = \langle \pi_{g\circ a}\rangle_{g:B\to Z} : \prod_{f:A\to Z} E \to \prod_{g:B\to Z} E$$

for $(a : A \to B) \in \| \mathsf{C} \|$. Then, K preserves monomorphisms. For, if $a : A \to B$ is a C-monomorphism, injectivity of Z yields, for each C-arrow $f : A \to Z$, a C-arrow $g : B \to Z$ satisfying $f = g \circ a$. Consequently, $\mathsf{K}a \circ x = \mathsf{K}a \circ y$ implies $\pi_f \circ x = \pi_f \circ y$ for any $f : A \to Z$, which, in turn, implies $x = y$. Finally, let $l, r : \mathsf{U_C} \Rightarrow \mathsf{KU_C}$ be given by:

$$l_{\langle C,\gamma\rangle} = \langle h \circ \mathsf{U_C}f^\flat\rangle_{f:C\to Z}$$
$$r_{\langle C,\gamma\rangle} = \langle k \circ \mathsf{U_C}f^\flat\rangle_{f:C\to Z}$$

for $\langle C, \gamma\rangle \in |\mathsf{Coalg(C, F)}|$. (Recall that $\langle D, \delta\rangle$ is cofree over Z w.r.t. $\mathsf{U_C}$.)

We now show that, under the assumption that all C-monomorphisms are regular, $\langle C, \gamma\rangle \models \varphi$ is equivalent to $\langle C, \gamma\rangle \models (\mathsf{K}, l, r)$. This follows from:

$$\begin{aligned}
\langle C,\gamma\rangle \models \varphi \;&\Leftrightarrow\\
\text{For any } f : C \to Z, \text{ there exists } g : \langle C,\gamma\rangle \to \langle S,\xi\rangle \text{ satisfying } f^\flat = \varphi \circ g \;&\Leftrightarrow\\
\text{For any } f : C \to Z, \text{ there exists } g' : C \to S \text{ satisfying } \mathsf{U_C}f^\flat = \mathsf{U_C}\varphi \circ g' \;&\Leftrightarrow\\
\text{For any } f : C \to Z,\ h \circ \mathsf{U_C}f^\flat = k \circ \mathsf{U_C}f^\flat \;&\Leftrightarrow\\
l_{\langle C,\gamma\rangle} = r_{\langle C,\gamma\rangle} \;&\Leftrightarrow\\
\langle C,\gamma\rangle \models (\mathsf{K}, l, r)&
\end{aligned}$$

$$\begin{array}{ccc}
\mathsf{Coalg(C, F)} & & \langle C,\gamma\rangle \\
& & \swarrow^{g} \;\downarrow f^\flat \\
\downarrow \mathsf{U_C} & & \langle S,\xi\rangle \xrightarrow[\varphi]{} \langle D,\delta\rangle \\
\mathsf{C} & & \\
& & C \\
& & \swarrow^{g'} \;\downarrow \mathsf{U_C}f^\flat \\
& & S \xrightarrow[\mathsf{U_C}\varphi]{} D \underset{k}{\overset{h}{\rightrightarrows}} E
\end{array}$$

The first and last of the above equivalences are just the definitions of the notions of satisfaction of modal formulae and respectively of coequations. The second equivalence is based on the observation that any $g' : C \to S$ satisfying $\mathsf{U_C}f^\flat = \mathsf{U_C}\varphi \circ g'$ defines a (C, F)-coalgebra homomorphism. This

follows from:

$$\begin{aligned}
\mathsf{FU_C}\varphi \circ \mathsf{F}g' \circ \gamma &= (\mathsf{U_C}\varphi \circ g' = \mathsf{U_C}f^\flat) \\
\mathsf{FU_C}f^\flat \circ \gamma &= (f^\flat \text{ is a } (\mathsf{C},\mathsf{F})\text{-coalgebra homomorphism}) \\
\delta \circ \mathsf{U_C}f^\flat &= (\mathsf{U_C}f^\flat = \mathsf{U_C}\varphi \circ g') \\
\delta \circ \mathsf{U_C}\varphi \circ g' &= (\varphi \text{ is a } (\mathsf{C},\mathsf{F})\text{-coalgebra homomorphism}) \\
\mathsf{FU_C}\varphi \circ \xi \circ g'
\end{aligned}$$

together with $\mathsf{FU_C}\varphi$ being a monomorphism (as $\mathsf{U_C}\varphi$ is a monomorphism, while F preserves monomorphisms). Also, the third equivalence exploits the hypothesis that any C-monomorphism (and hence also $\mathsf{U_C}\varphi$) is regular to infer that $\mathsf{U_C}\varphi$ defines an equaliser for h, k. (The dual of 7 of Proposition 2.1.43 is used here.) Finally, the fourth equivalence uses the definitions of l and r.

The following can now be inferred about the expressiveness of abstract coequations with regard to characterising covarieties.

Corollary 3.1.67. *Let* (C,F) *denote an abstract cosignature, such that* $(\mathsf{Epi}(\mathsf{C}),\mathsf{Mono}(\mathsf{C}))$ *defines a factorisation system for* C*, such that all* C*-monomorphisms are regular, and such that* C *has enough injectives and is wellpowered. Then, a full subcategory* $\mathcal{K}$ *of* $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$ *is a covariety if and only if* $\mathcal{K}$ *is definable by a class of* (C,F)*-coequations.*

Remark 3.1.68. The requirement that all C-monomorphisms are regular amongst the hypotheses of Corollary 3.1.67 is directly related to the fact that the definition of covarieties requires closure under homomorphisms whose underlying C-arrows are *arbitrary* monomorphisms. This requirement would not be necessary if a notion of covariety that involved closure under homomorphisms whose underlying C-arrows are *regular* monomorphisms was considered. In this case, however, $(\mathsf{Epi}(\mathsf{C}),\mathsf{RegMono}(\mathsf{C}))$ would have to define a factorisation system for C. Furthermore, the endofunctor F would have to preserve regular monomorphisms, in order to allow this factorisation system to be lifted to $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$. Under these assumptions, [Kur00, Proposition 1.3.3] would result in $\mathsf{U_C}$ creating factorisations w.r.t. $(\mathsf{Epi}(\mathsf{C}),\mathsf{RegMono}(\mathsf{C}))$, while [Kur00, Proposition 1.3.5] would result in $(\mathsf{U_C}^{-1}(\mathsf{Epi}(\mathsf{C})),\mathsf{U_C}^{-1}(\mathsf{RegMono}(\mathsf{C})))$ defining a factorisation system for $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$.

Remark 3.1.69. Stronger characterisability results (see [Kur00, Theorem 2.5.7 and Corollary 2.5.8]), stating the definability of covarieties in terms of *single* modal formulae exist for *bounded categories* (see [Kur00, Definition 1.5.3]). A variant of Corollary 3.1.67 which, under the assumption that $\mathsf{Coalg}(\mathsf{C},\mathsf{F})$ is bounded (e.g. by a cofree coalgebra), states the definability of (C,F)-covarieties in terms of single coequations can also be formulated.

While important from a theoretical point of view, Corollary 3.1.67 is of little practical importance since, unlike in the algebraic case, the coequations used to characterise covarieties do not have finitary syntactic presentations. This observation will become more apparent in Chapter 4, where coalgebraic notions of *coterm* and *coequation*, syntactically dual to the many-sorted algebraic notions of term and equation, will be shown to be insufficiently expressive to yield characterisability results for covarieties.

3.2 Specification of Computational Structures

Dualising the notions of abstract cosignature (after leaving out the requirement that the endofunctor in question preserves pullbacks, see Remark 3.1.3), observer and coequation yields notions of *abstract signature*, *constructor* and *equation*, with results similar to the ones stated in Section 3.1 and marked * holding for categories of algebras. Further constraints on the endofunctors defining abstract signatures are, however, required to guarantee the existence of quotients in the categories of algebras of abstract signatures and their creation by the functors taking algebras of abstract signatures to their carrier. This section outlines the algebraic framework obtained by incorporating such constraints into the approach resulting from the previously-mentioned dualisation.

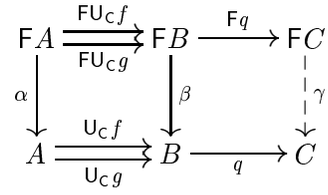
3.2.1 Signatures, Algebras, Initiality and Reachability

Definition 3.2.1. *An* **(abstract) signature** *is a pair* (C, F)*, with* C *a category which is complete, cocomplete*[17] *and regular, and with* $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ *an endofunctor which preserves coequalisers of kernel pairs, epimorphisms and colimits of* ω*-chains.*

Definition 3.2.2. *Let* (C, F) *denote an abstract signature. A* (C, F)-**algebra** *(respectively* (C, F)-**algebra homomorphism***) is an* F*-algebra (* F*-algebra homomorphism).*

For an abstract signature (C, F), the category of (C, F)-algebras and (C, F)-algebra homomorphisms is denoted $\mathsf{Alg}(\mathsf{C}, \mathsf{F})$, while the functor taking (C, F)-algebras to their carrier is denoted $\mathsf{U_C} : \mathsf{Alg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$.

Remark 3.2.3. For an abstract signature (C, F), the fact that F preserves coequalisers of kernel pairs results in $\mathsf{U_C}$ creating coequalisers of congruences (see Definition 2.3.40). by $\mathsf{U_C}$. For, if $f, g : \langle A, \alpha \rangle \to \langle B, \beta \rangle$ denote (C, F)-algebra homomorphisms such that $\mathsf{U_C} f, \mathsf{U_C} g$ define a kernel pair in C, and if $q : B \to C$ denotes a coequaliser for $\mathsf{U_C} f, \mathsf{U_C} g$, the preservation of coequalisers of kernel pairs by F results in $\mathsf{F}q$ defining a coequaliser for $\mathsf{FU_C} f, \mathsf{FU_C} g$.



This, together with $q \circ \beta \circ \mathsf{FU_C} f = q \circ \beta \circ \mathsf{FU_C} g$ (following from the fact that q is a coequaliser for $\mathsf{U_C} f, \mathsf{U_C} g$ together with the fact that f and g are (C, F)-algebra homomorphisms) yield a unique C-arrow $\gamma : \mathsf{F}C \to C$ satisfying $q \circ \beta = \gamma \circ \mathsf{F}q$. That is, q defines a (C, F)-algebra homomorphism from $\langle B, \beta \rangle$ to $\langle C, \gamma \rangle$. The uniqueness of a (C, F)-algebra structure $\langle C, \gamma \rangle$ making q into a (C, F)-algebra homomorphism follows from any such structure having to satisfy $q \circ \beta = \gamma \circ \mathsf{F}q$, together with $\mathsf{F}q$ being an epimorphism (as it is a coequaliser). Also, the fact that q defines a coequaliser for f, g in $\mathsf{Alg}(\mathsf{C}, \mathsf{F})$ follows from q defining a coequaliser for $\mathsf{U_C} f, \mathsf{U_C} g$ in C, together with $\mathsf{F}q$ being an epimorphism.

[17] It would be sufficient to require that C has pullbacks, products, finite colimits and colimits of ω-chains.

For, given a (C, F)-algebra homomorphism $p : \langle B, \beta\rangle \to \langle D, \delta\rangle$ satisfying $p \circ f = p \circ g$, the fact that q defines a coequaliser for $\mathsf{U_C} f, \mathsf{U_C} g$ yields a unique C-arrow $r : C \to D$ satisfying $\mathsf{U_C} p = r \circ q$. Moreover, $r \circ \gamma = \delta \circ \mathsf{F} r$. This follows from: $r \circ \gamma \circ \mathsf{F} q = r \circ q \circ \beta = \mathsf{U_C} p \circ \beta = \delta \circ \mathsf{F U_C} p = \delta \circ \mathsf{F} r \circ \mathsf{F} q$, together with $\mathsf{F} q$ being an epimorphism. That is, r defines a (C, F)-algebra homomorphism from $\langle C, \gamma\rangle$ to $\langle D, \delta\rangle$. This concludes the proof of the fact that $\mathsf{U_C}$ creates coequalisers of congruences.

Proposition 3.2.4. *Let* (C, F) *denote an abstract signature. Then,* $\mathsf{U_C} : \mathsf{Alg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$ *is algebraic.*

Proof. The creation of coequalisers of congruences by $\mathsf{U_C}$ follows by Remark 3.2.3. Also, the existence of a left adjoint to $\mathsf{U_C}$ and the creation of limits by $\mathsf{U_C}$ follow by Proposition 2.3.50 together with Remark 2.3.47 and Proposition 2.3.48. □

Example 3.2.5 (**Many-Sorted Signatures***).* Many-sorted signatures are an instance of the abstract notion of signature. The preservation of coequalisers of kernel pairs, of epimorphisms and of colimits of ω-chains by the endofunctors induced by many-sorted signatures (see Remark 2.3.38) follows by Example 2.1.37.

The constraints in Definition 3.2.1 ensure the existence of an initial object in Alg(C, F) (see e.g. [SP82]).

Proposition 3.2.6. *Let* (C, F) *denote an abstract signature. Then,* Alg(C, F) *has an initial object.*

Also, the existence of limits in C results in the existence of limits in Alg(C, F) and in their preservation by $\mathsf{U_C}$.

Proposition 3.2.7. *Let* (C, F) *denote an abstract signature. Then,* $\mathsf{U_C}$ *creates limits.*

Proof. The conclusion follows by the dual of Proposition 2.4.9. □

Corollary 3.2.8. *For an abstract signature* (C, F)*,* Alg(C, F) *has limits and* $\mathsf{U_C}$ *preserves them.*

Proof. The conclusion follows by the dual of Corollary 2.4.10. □

The existence of quotients in C together with the preservation of coequalisers of kernel pairs by F result in the existence of quotients in Alg(C, F).

Proposition 3.2.9. *Let* (C, F) *denote an abstract signature. Then,* $\mathsf{U_C}$ *creates quotients.*

Proof. $\mathsf{U_C}$ creates kernel pairs and coequalisers of kernel pairs. □

Corollary 3.2.10. *For an abstract signature* (C, F)*,* Alg(C, F) *has quotients and* $\mathsf{U_C}$ *preserves them.*

Proof. By Corollary 3.2.8, kernel pairs exist in Alg(C, F) and are preserved by $\mathsf{U_C}$. Then, since $\mathsf{U_C}$ creates coequalisers of congruences (see Remark 3.2.3), and since coequalisers exist in C, it follows that quotients exist in Alg(C, F) and are preserved by $\mathsf{U_C}$. □

The quotient of a (C, F)-algebra homomorphism defines a homomorphic image of the domain of this homomorphism. And because C is regular, this also yields a subalgebra of the codomain of the given homomorphism.

A notion of *reachability* of algebras, dual to that of observability of coalgebras (see Definition 3.1.15) and of which that of Definition 2.3.11 is an instance can also be defined.

Definition 3.2.11. *Let* (C, F) *denote an abstract signature. A* (C, F)*-algebra* $\langle A, \alpha \rangle$ *is* **reachable** *if and only if the* C*-arrow underlying the unique homomorphism from the initial* (C, F)*-algebra to* $\langle A, \alpha \rangle$ *is an epimorphism.*

3.2.2 Constructors, Equations and Satisfaction (up to Reachability)

Dualising the coalgebraic notions of observer, coequation and coequational satisfaction (Definitions 3.1.23 and 3.1.25) yields algebraic notions of *constructor*, *equation* and *equational satisfaction*.

Definition 3.2.12. *Let* (C, F) *denote an abstract signature. A* (C, F)-**constructor** *is a pair* (K, c)*, with* $\mathsf{K} : \mathsf{C} \to \mathsf{C}$ *a functor which preserves epimorphisms, and with* $c : \mathsf{KU_C} \Rightarrow \mathsf{U_C}$ *a natural transformation. A* (C, F)-**equation** *is a tuple* (K, l, r)*, with* (K, l) *and* (K, r) *denoting* (C, F)*-constructors. A* (C, F)*-algebra* $\langle A, \alpha \rangle$ **satisfies** *a* (C, F)*-equation* (K, l, r) *(written* $\langle A, \alpha \rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}, l, r)$*) if and only if* $l_\alpha = r_\alpha$*.*

For an abstract signature (C, F) and a class E of (C, F)-equations, the full subcategory of Alg(C, F) whose objects satisfy the equations in E is denoted Alg(C, F, E).

Example 3.2.13 (**Many-Sorted Equational Satisfaction***).* The many-sorted algebraic notion of term is an instance of the abstract notion of constructor, while the many-sorted algebraic notions of equation and equational satisfaction are instances of the abstract notions of equation and respectively equational satisfaction. For, if $\mathsf{F}_\Sigma : \mathsf{Set}^S \to \mathsf{Set}^S$ denotes the endofunctor induced by a many-sorted signature Σ with sort set S (see Remark 2.3.38), then Σ-terms $t \in T_\Sigma(\mathcal{V})_s$ with $\mathcal{V}$ consisting of variables $X_1 : s_1, \ldots, X_m : s_m$ induce $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructors (K, t'), with $\mathsf{K} : \mathsf{Set}^S \to \mathsf{Set}^S$ being given by:

$$(\mathsf{K}X)_{s'} = \begin{cases} X_{s_1} \times \ldots \times X_{s_m} & \text{if } \ s' = s \\ \emptyset & \text{otherwise} \end{cases}$$

for $X \in |\mathsf{Set}^S|$, and with $t' : \mathsf{KU} \Rightarrow \mathsf{U}$ (where $\mathsf{U} : \mathsf{Alg}(\mathsf{Set}^S, \mathsf{F}_\Sigma) \to \mathsf{Set}^S$ denotes the functor taking $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-algebras to their carrier) being given by:

$$(t'_{\langle A,\alpha \rangle})_{s'} = \begin{cases} t_A & \text{if } \ s' = s \\ ! : \emptyset \to A_{s'} & \text{otherwise} \end{cases}$$

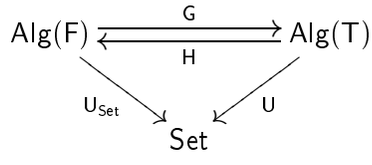
for $\langle A, \alpha \rangle \in |\mathsf{Alg}(\mathsf{Set}^S, \mathsf{F}_\Sigma)|$ with A its associated Σ-algebra. (The fact that K preserves epimorphisms follows by Example 2.1.37.) Also, unconditional Σ-equations e of form $(\forall\, \mathcal{V})\ l = r$ induce $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-equations (K, l', r'), with $\mathsf{K} : \mathsf{Set}^S \to \mathsf{Set}^S$ being as before, and with $l', r' : \mathsf{KU} \Rightarrow \mathsf{U}$ being

the $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructors associated to l and respectively r. Moreover, $\langle A, \alpha\rangle \models_{(\mathsf{Set}^S, \mathsf{F}_\Sigma)} (\mathsf{K}, l', r')$ if and only if $A \models_\Sigma (\forall \mathcal{V})\ l = r$[18].

We conclude this example with a remark on the expressiveness of $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructors. Specifically, we note that any $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructor (K, c) with K of the above form induces a Σ-term $t \in T_\Sigma(\{X_1, \ldots, X_m\})_s$, with t being given by $c_{T_\Sigma(X_1,\ldots,X_m),s}(X_1, \ldots, X_m)$. Furthermore, the $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructor induced by the resulting Σ-term t coincides with (K, c). This follows from: $c_{\langle A,\alpha\rangle}(a_1, \ldots, a_m) = \theta^\#(c_{T_\Sigma(X_1,\ldots,X_m)}(X_1, \ldots, X_m)) = \theta^\#(t) = t_A$ for any $\langle A, \alpha\rangle \in |\mathsf{Alg}(\mathsf{Set}^S, \mathsf{F}_\Sigma)|$ and any $a_i \in A_{s_i}$ for $i = 1, \ldots, m$, with $\theta : \mathcal{V} \to A$ assigning the value a_i to the variable X_i, for $i = 1, \ldots, m$. (The first equality is a consequence of the existence of a $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-algebra homomorphism $\theta^\# : T_\Sigma(X_1, \ldots, X_m) \to \langle A, \alpha\rangle$ extending θ.) The existence of such a close relationship between Σ-terms and $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-constructors is due to the existence of a characterisation of free algebras as sorted sets of terms with variables. We shall see in Chapter 4 that a similar relationship does not hold between a coalgebraically-defined notion of *coterm* over a *many-sorted cosignature* and the abstract notion of observer.

Remark 3.2.14. The notion of constructor introduced in Definition 3.2.12 is a slight generalisation of the notion of *semantic operation* introduced in [Man76] in the context of monads on Set. Given a monad (T, η, μ) on Set together with $X \in |\mathsf{Set}|$, a *semantic operation in X (w.r.t.* T*)* [Man76, Definition 5.3] is a natural transformation of type $(\_)^X \circ \mathsf{U} \Rightarrow \mathsf{U}$, with the functor $\mathsf{U} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{Set}$ taking T-algebras to their carrier, and with the functor $(\_)^X : \mathsf{Set} \to \mathsf{Set}$ taking a set A to the set of functions from X to A. The semantic operations in X are then shown [Man76, Theorem 5.5] to be in one-to-one correspondence with the *syntactic operations in X* [Man76, Definition 5.3], defined as elements of $\mathsf{T}X$.

Now let $(\mathsf{Set}, \mathsf{F})$ denote an abstract signature over Set, and let (T, η, μ) denote the monad induced by F (see Proposition 2.3.50). Also, let $\mathsf{G} : \mathsf{Alg}(\mathsf{F}) \to \mathsf{Alg}(\mathsf{T})$ and $\mathsf{H} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{Alg}(\mathsf{F})$ denote the functors defining the isomorphism of categories $\mathsf{Alg}(\mathsf{T}) \simeq \mathsf{Alg}(\mathsf{F})$, satisfying $\mathsf{GH} = \mathsf{Id}_{\mathsf{Alg}(\mathsf{T})}$ and $\mathsf{HG} = \mathsf{Id}_{\mathsf{Alg}(\mathsf{F})}$, as well as $\mathsf{UG} = \mathsf{U}_{\mathsf{Set}}$ and $\mathsf{U}_{\mathsf{Set}}\mathsf{H} = \mathsf{U}$ (see Proposition 2.3.50).



Then, semantic operations $\alpha : (\_)^X \circ \mathsf{U} \Rightarrow \mathsf{U}$ induce $(\mathsf{Set}, \mathsf{F})$-constructors $((\_)^X, \alpha')$, with $\alpha' : (\_)^X \circ \mathsf{U}_{\mathsf{Set}} \Rightarrow \mathsf{U}_{\mathsf{Set}}$ being given by α_{G}. (However, not any $(\mathsf{Set}, \mathsf{F})$-constructor corresponds to a semantic operation.)

In the case of abstract signatures $(\mathsf{Set}, \mathsf{F}_\Sigma)$ induced by a one-sorted signature Σ, the one-to-one correspondence between Σ-terms with variables in $\mathcal{V}$ and $(\mathsf{Set}, \mathsf{F}_\Sigma)$-constructors of form $((\_)^\mathcal{V}, c)$ outlined in Example 3.2.13 is precisely the one-to-one correspondence between syntactic and semantic operations in $\mathcal{V}$ described in [Man76].

[18] This does not generalise to conditional many-sorted equations, since such equations do not, in general, induce pairs of arrows into the carriers of algebras of the underlying signature.

A notion of *satisfaction of equations up to reachability*, dual to that of satisfaction of coequations up to bisimulation, can also be defined.

Definition 3.2.15. *Let* (C,F) *denote an abstract signature. A* (C,F)*-algebra* $\langle A,\alpha\rangle$ **satisfies** *a* (C,F)*-equation* (K,l,r) **up to reachability** *(written* $\langle A,\alpha\rangle \models^{\mathsf{r}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$*) if and only if* $l_\alpha \circ \mathsf{KU_C}! = r_\alpha \circ \mathsf{KU_C}!$*, with* $! : \langle I,\xi\rangle \to \langle A,\alpha\rangle$ *denoting the unique* (C,F)*-algebra homomorphism from the initial* (C,F)*-algebra to* $\langle A,\alpha\rangle$*.*

Example 3.2.16 (**Many-Sorted Equational Satisfaction up to Reachability***).* Let Σ denote a many-sorted signature with sort set S. A Σ-algebra A is said to *satisfy* a Σ-equation e of form $(\forall\,\mathcal{V})\ l = r$ *up to reachability* (written $A \models^{\mathsf{r}}_{\Sigma} e$) if and only if $\theta^{\#}(l) = \theta^{\#}(r)$ holds for any assignment $\theta : \mathcal{V} \to \mathsf{Im}(!_A)$, where $!_A : T_\Sigma \to A$ denotes the unique Σ-homomorphism from the initial Σ-algebra to A. Now if F_Σ, (K,l',r') and $\langle A,\alpha\rangle$ are as in Example 3.2.13, then $A \models^{\mathsf{r}}_{\Sigma} e$ holds precisely when $\langle A,\alpha\rangle \models^{\mathsf{r}}_{(\mathsf{Set}^S,\mathsf{F}_\Sigma)} (\mathsf{K},l',r')$ does. (This follows from any assignment $\theta : \mathcal{V} \to \mathsf{Im}(!_A)$ admitting a factorisation of form $\theta = !_A \circ \theta'$, for some $\theta' : \mathcal{V} \to T_\Sigma$.)

Standard satisfaction of (K,l,r) by $\langle A,\alpha\rangle$ implies its satisfaction up to reachability by $\langle A,\alpha\rangle$. And if, in addition, $\langle A,\alpha\rangle$ is reachable, then the converse also holds. (In this case, $\mathsf{KU_C}!$ is an epimorphism.)

Similarly to the coalgebraic case, satisfaction up to reachability can be expressed in terms of standard satisfaction by the codomains of the quotients of the unique homomorphisms from the initial algebra.

Proposition 3.2.17. *Let* (C,F) *denote an abstract signature, let* $\langle A,\alpha\rangle$ *denote a* (C,F)*-algebra, and let* (K,l,r) *denote a* (C,F)*-equation. Also, let* $e : \langle A,\alpha\rangle \to \langle E,\eta\rangle$ *denote the quotient of the unique* (C,F)*-algebra homomorphism from the initial* (C,F)*-algebra to* $\langle A,\alpha\rangle$*. Then,* $\langle A,\alpha\rangle \models^{\mathsf{r}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ *if and only if* $\langle E,\eta\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$*.*

Proof. Similar to the proof of Proposition 3.1.38. □

Proving that a (C,F)-equation (K,l,r) holds in a full subcategory $\mathcal{A}$ of $\mathsf{Alg}(\mathsf{C},\mathsf{F})$ can be reduced to exhibiting a functor $\mathsf{S} : \mathcal{A} \to \mathsf{Alg}(\mathsf{C},\mathsf{F})$ together with a natural transformation $\iota : \mathsf{S} \Rightarrow \mathsf{I}_{\mathcal{A}}$, with $\mathsf{I}_{\mathcal{A}}$ denoting the inclusion of $\mathcal{A}$ into $\mathsf{Alg}(\mathsf{C},\mathsf{F})$[19], such that $\mathsf{S}\langle A,\alpha\rangle \models^{\mathsf{r}}_{(\mathsf{C},\mathsf{F})} (\mathsf{K},l,r)$ for each $\langle A,\alpha\rangle \in |\mathcal{A}|$.

Results similar to those marked * in Section 3.1.2 hold for the satisfaction (up to reachability) of equations by algebras of abstract signatures. In particular, equations induce varieties of algebras, both w.r.t. standard satisfaction and w.r.t. satisfaction up to reachability.

3.2.3 Institutions of Computational Structures

Definition 3.2.18. *An* **(abstract) signature morphism** *between abstract signatures* (C,F) *and* (D,G) *is a pair* (U,ξ)*, with* $\mathsf{U} : \mathsf{D} \to \mathsf{C}$ *a functor which preserves colimits*[20] *and has a left adjoint*

[19] The natural transformation ι could, for instance, define a (C,F)-subalgebra of $\langle A,\alpha\rangle$, for each $\langle A,\alpha\rangle \in |\mathcal{A}|$.

[20] It would be sufficient to require that U preserves pushouts, coequalisers and colimits of ω-chains.

L[21], *and with* $\xi : \mathsf{FU} \Rightarrow \mathsf{UG}$ *a natural transformation. If, in addition,* U *lifts limits and colimits*[22], *and if* L *is also a right inverse to* U, *then the signature morphism* (U, ξ) *is called* **strong**.

The quasi-category of abstract signatures and abstract signature morphisms is denoted Sign, while the quasi-category of abstract signatures and strong abstract signature morphisms is denoted SSign. (The composition of abstract signature morphisms $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ and $(\mathsf{V}, \eta) : (\mathsf{D}, \mathsf{G}) \to (\mathsf{E}, \mathsf{H})$ is given by the abstract signature morphism $(\mathsf{UV}, \mathsf{U}\xi \circ \eta_{\mathsf{V}}) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{E}, \mathsf{H})$. Moreover, if (U, ξ) and (V, η) are strong, then so is $(\mathsf{UV}, \mathsf{U}\xi \circ \eta_{\mathsf{V}})$.)

An abstract signature morphism $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ induces a reduct functor $\mathsf{U}_\xi : \mathsf{Alg}(\mathsf{D}, \mathsf{G}) \to \mathsf{Alg}(\mathsf{C}, \mathsf{F})$ on the one hand (with U_ξ taking a (D, G)-algebra $\langle A, \alpha \rangle$ to the (C, F)-algebra $\langle \mathsf{U}A, \mathsf{U}\alpha \circ \xi_A \rangle$), and a translation of (C, F)-constructors and equations to (D, G)-constructors and equations on the other (with the translation of a (C, F)-constructor (K, c) along (U, ξ) being given by the (D, G)-constructor $(\mathsf{LKU}, c^{\#}_{\mathsf{U}_\xi})$[23]). This yields functors $\mathsf{Alg} : \mathsf{Sign} \to \mathsf{CAT}^{\mathsf{op}}$ and respectively $\mathsf{Eqn} : \mathsf{Sign} \to \mathsf{SET}$.

Theorem 3.2.19. $(\mathsf{Sign}, \mathsf{Alg}, \mathsf{Eqn}, \models)$ *is an institution.*

Proof. Similar to the proof of Theorem 3.1.49. □

Example 3.2.20 (**Many-Sorted Signature Morphisms***).* Many-sorted signature morphisms are an instance of the abstract notion of signature morphism. For, if $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$ and $(\mathsf{Set}^{S'}, \mathsf{F}_{\Sigma'})$ denote the abstract signatures induced by the many-sorted signatures (S, Σ) and respectively (S', Σ') (see Example 3.2.5), then many-sorted signature morphisms $\phi : \Sigma \to \Sigma'$ induce abstract signature morphisms (U, η_ϕ), with $\mathsf{U} : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ taking $X' \in |\mathsf{Set}^{S'}|$ to $(X'_{\phi(s)})_{s \in S} \in |\mathsf{Set}^S|$, and with $\eta_\phi : \mathsf{F}_\Sigma \mathsf{U} \Rightarrow \mathsf{U}\mathsf{F}_{\Sigma'}$ being given by:

$$(\eta_{\phi,X})_s(\iota_\sigma(\langle x_1, \ldots, x_m \rangle)) = \iota_{\phi(\sigma)}(\langle x_1, \ldots, x_m \rangle)$$

for $X \in |\mathsf{Set}^{S'}|$, $s \in S$, $\sigma \in \Sigma_{s_1 \ldots s_m, s}$ and $x_i \in X_{\phi(s_i)}$ for $i = 1, \ldots, m$. For, the functor U preserves colimits (see Example 2.1.38) and has a left adjoint L (see Example 2.1.52). And if, in addition, ϕ is injective on sorts, then U lifts limits and colimits (see Example 2.1.38), while L is also a right inverse to U (see Example 2.1.52). Moreover, U_{η_ϕ} agrees with U_ϕ, while the translation of $(\mathsf{Set}^S, \mathsf{F}_\Sigma)$-equations along η_ϕ agrees with the translation of Σ-equations along ϕ.

As in the coalgebraic case, further constraints need to be imposed to abstract signature morphisms in order to obtain an institution w.r.t. the satisfaction of equations up to reachability.

Definition 3.2.21. *An abstract signature morphism* $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ *is* **horizontal** *if and only if* ξ *is a natural epimorphism.*

[21] Consequently, U also preserves limits (see Proposition 2.1.54).

[22] Hence, by Proposition 2.1.35, U preserves limits and colimits.

[23] The preservation of epimorphisms by LKU follows from the preservation of either pushouts or epimorphisms by each of U, K and L.

The quasi-category of abstract signatures and horizontal abstract signature morphisms is denoted HSign.

Theorem 3.2.22. $(\mathsf{HSign}, \mathsf{Alg}\restriction_{\mathsf{HSign}}, \mathsf{Eqn}\restriction_{\mathsf{HSign}}, \models^{\mathsf{r}})$ *is an institution.*

Example 3.2.23 (**Many-Sorted Signature Morphisms Contd.***).* If the many-sorted signature morphism $\phi : \Sigma \to \Sigma'$ is such that $\sigma' \in \Sigma'_{w',\phi(s)}$ implies $\sigma' = \phi(\sigma)$ for some $\sigma \in \Sigma_{w,s}$, then the abstract signature morphism induced by ϕ is horizontal. (In this case, all the components of the natural transformation η_ϕ are epimorphisms.)

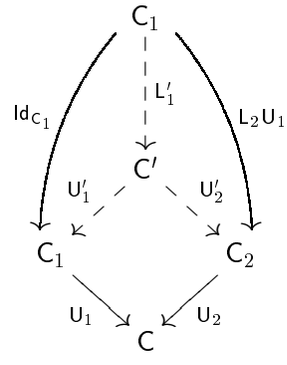
3.2.4 Compositionality Results

Similarly to the coalgebraic case, any functor $\mathsf{U} : \mathsf{D} \to \mathsf{C}$ satisfying the conditions in the definition of abstract signature morphisms induces a lifting of abstract signatures (C, F) over C to abstract signatures $(\mathsf{D}, \bar{\mathsf{F}})$ over D (with $\bar{\mathsf{F}}$ being given by LFU), such that $(\mathsf{U}, \eta_{\mathsf{FU}})$ (where $\eta : \mathsf{Id}_{\mathsf{C}} \Rightarrow \mathsf{UL}$ denotes the unit of the adjunction $\mathsf{U} \vdash \mathsf{L}$) defines a signature morphism from (C, F) to $(\mathsf{D}, \bar{\mathsf{F}})$. (The preservation of epimorphisms and of colimits of ω-chains by each of U, F and L results in their preservation by $\bar{\mathsf{F}}$. Also, the preservation of coequalisers by U and L together with the preservation of kernel pairs by U and the preservation of coequalisers of kernel pairs by F result in the preservation of coequalisers of kernel pairs by $\bar{\mathsf{F}}$.) Then, an arbitrary signature morphism $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{D}, \mathsf{G})$ determines an abstract signature morphism $(\mathsf{Id}_{\mathsf{D}}, \bar{\xi}) : (\mathsf{D}, \bar{\mathsf{F}}) \to (\mathsf{D}, \mathsf{G})$, with $\bar{\xi} : \bar{\mathsf{F}} \Rightarrow \mathsf{G}$ denoting the natural transformation whose components are given by $\bar{\xi}_D = \xi^{\#}_D$ for $D \in |\mathsf{D}|$, such that the following holds: $(\mathsf{Id}_{\mathsf{D}}, \bar{\xi}) \circ (\mathsf{U}, \eta_{\mathsf{FU}}) = (\mathsf{U}, \xi)$. This observation will allow any finite diagram in SSign to be lifted to a diagram all of whose signatures have the same underlying category.

Theorem 3.2.24. SSign *is finitely cocomplete.*

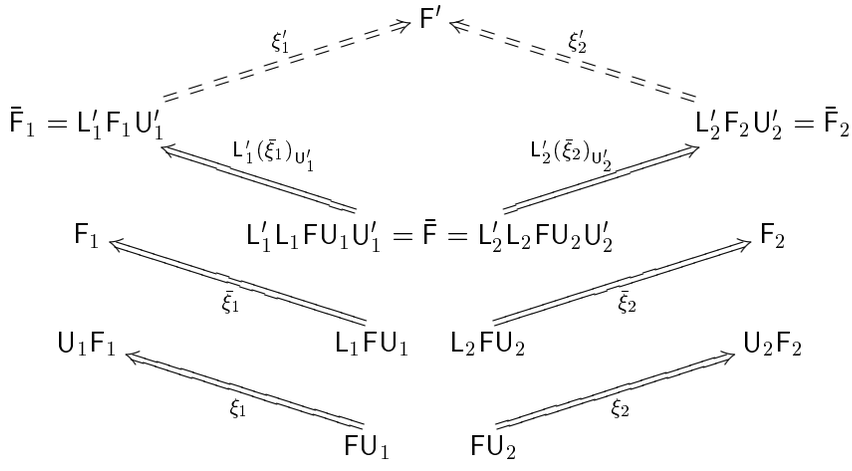
Proof. An initial object in SSign is given by the abstract signature $(1, \mathsf{Id}_1)$, with 1 and Id_1 having the same denotations as in Theorem 3.1.58.

The construction of pushouts in SSign is similar to the construction of pushouts in SCosign (see Theorem 3.1.58). Let $(\mathsf{U}_1, \xi_1) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}_1, \mathsf{F}_1)$ and $(\mathsf{U}_2, \xi_2) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}_2, \mathsf{F}_2)$ denote abstract signature morphisms. Also, let $\mathsf{U}'_1 : \mathsf{C}' \to \mathsf{C}_1$, $\mathsf{U}'_2 : \mathsf{C}' \to \mathsf{C}_2$ define the pullback of U_1, U_2 in CAT, and let $\mathsf{L}'_1 : \mathsf{C}_1 \to \mathsf{C}'$, $\mathsf{L}'_2 : \mathsf{C}_2 \to \mathsf{C}'$ denote the only functors satisfying $\mathsf{U}'_1\mathsf{L}'_1 = \mathsf{Id}_{\mathsf{C}_1}$, $\mathsf{U}'_2\mathsf{L}'_1 = \mathsf{L}_2\mathsf{U}_1$ (resulting from the fact that $\mathsf{U}_1\mathsf{Id}_{\mathsf{C}_1} = \mathsf{U}_1 = \mathsf{U}_2\mathsf{L}_2\mathsf{U}_1$), and similarly for L'_2.



Then, L'_1 and L'_2 define left adjoints, right inverses to U'_1 and respectively U'_2, and moreover, $L'_1 L_1 = L'_2 L_2$. Also, C' has limits and colimits, while U'_1 and U'_2 lift them. (The proof of these statements uses Proposition 2.1.39 and Remark 2.1.40, and is similar to the proof of the corresponding statements in Theorem 3.1.58.) Finally, C' is regular. (Again, this follows similarly to Theorem 3.1.58.)

Now let $\bar{F}, \bar{F}_1, \bar{F}_2 : C' \to C'$ denote the liftings of F, F_1 and F_2 along $U = U_1 U'_1 = U_2 U'_2$, U'_1 and U'_2 respectively. Finally, let $\xi'_1 : \bar{F}_1 \Rightarrow F'$, $\xi'_2 : \bar{F}_2 \Rightarrow F'$ denote the pushout of $L'_1(\bar{\xi}_1)_{U'_1} : \bar{F} \Rightarrow \bar{F}_1$ and $L'_2(\bar{\xi}_2)_{U'_2} : \bar{F} \Rightarrow \bar{F}_2$ in (the quasi-category) $[C', C']$ (see Proposition 2.1.41 and Remark 2.1.42).



The preservation of coequalisers of kernel pairs, of epimorphisms and of colimits of ω^{op}-chains by each of $\bar{F}$, $\bar{F}_1$ and $\bar{F}_2$ results in their preservation by F' (see Proposition 2.1.41).

The pushout of (U_1, ξ_1) and (U_2, ξ_2) is then given by $(U'_1, U'_1\xi'_1) : (C_1, F_1) \to (C', F')$ and $(U'_2, U'_2\xi'_2) : (C_2, F_2) \to (C', F')$. The universality of $((U'_1, U'_1\xi'_1), (U'_2, U'_2\xi'_2))$ is a consequence of the couniversality of (U'_1, U'_2) together with the universality of each of $\bar{\xi}_1$, $\bar{\xi}_2$ and the universality of (ξ'_1, ξ'_2).

The conclusion now follows by the dual of Proposition 2.1.30. □

Remark 3.2.25. Theorem 3.2.24 and Corollary 2.2.6 result in the subcategories of the categories of specifications associated to the institutions $(\mathsf{Sign}, \mathsf{Alg}, \mathsf{Eqn}, \models)$ and $(\mathsf{HSign}, \mathsf{Alg}\restriction_{\mathsf{HSign}}, \mathsf{Eqn}\restriction_{\mathsf{HSign}}, \models^r)$ (see Theorems 3.2.19 and 3.2.22) whose specification morphisms are underlain by strong abstract signature morphisms also being finitely cocomplete.

The following also holds.

Theorem* 3.2.26. *The functor* $\mathsf{Alg}\restriction_{\mathsf{SSign}} : \mathsf{SSign} \to \mathsf{CAT}^{op}$ *preserves finite colimits.*

Proof. Similar to the proof of Theorem 3.1.60. □

3.2.5 Semantic Constructions

The formulation of results similar to Proposition 3.1.33 and Theorem 3.1.64 for abstract algebraic specifications and respectively algebraic specification morphisms requires further constraints on abstract signatures, namely constraints which ensure the existence of arbitrary coequalisers in the

categories of algebras of such signatures. According to [Bor94b, Theorem 4.3.5], imposing some additional constraints to the underlying categories of abstract signatures is sufficient in this sense. Specifically, it is shown there that if the underlying category C of a monad T is complete, cocomplete and regular, and if each regular epimorphism in C has a section, then the category Alg(T) is itself complete, cocomplete and regular. According to Proposition 2.3.50, the category Alg(C, F) with (C, F) an abstract signature is isomorphic to the category Alg(T) for some monad T on C. Consequently, if each regular epimorphism in C has a section, the category Alg(C, F) has coequalisers.

Moreover, the requirement that regular epimorphisms in the underlying categories of abstract signatures have sections is preserved by finite colimits in SSign, provided that the diagrams considered are such that the functors underlying the abstract signature morphisms defining these diagrams also lift sections of regular epimorphisms. For, the initial abstract signature satisfies the above-mentioned property, and this property is preserved by pushouts in SSign of abstract signature morphisms whose underlying functors lift sections of regular epimorphisms. Specifically, given the pushout diagram in SSign obtained in the proof of Theorem 3.2.24, the existence of a section s for each regular epimorphism r in C′ follows from the existence of sections s_1 and s_2 for $\mathsf{U}'_1 r$ and respectively $\mathsf{U}'_2 r$, additionally satisfying $\mathsf{U}_1 s_1 = \mathsf{U}_2 s_2$ (as U_1 preserves sections of regular epimorphisms, while U_2 lifts them).

Provided that coequalisers exist in the categories of algebras of abstract signatures, one can assign suitable denotations to abstract algebraic specifications and algebraic specification morphisms.

Proposition 3.2.27. *Let* (C, F) *denote an abstract signature, such that* Alg(C, F) *has coequalisers. Also, let* $\langle A, \alpha\rangle$ *denote a* (C, F)*-algebra, let* e *denote a* (C, F)*-equation, and let* $(\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha)^e$ *denote the full subcategory of* $\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha$ *whose objects satisfy the equation* e*. Then,* $(\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha)^e$ *has an initial object, which at the same time defines a* (C, F)*-homomorphic image of* $\langle A, \alpha\rangle$*.*

Proof. Say e is of form (K, l, r). Let (T, η, μ) denote the monad induced by (C, F), let $l^{\#}_{\alpha}, r^{\#}_{\alpha} : \langle \mathsf{T}\mathsf{K}A, \alpha'\rangle \to \langle A, \alpha\rangle$ denote the unique extensions of the C-arrows $l_\alpha, r_\alpha : \mathsf{K}A \to A$ to (C, F)-algebra homomorphisms, and let $q : \langle A, \alpha\rangle \to \langle Q, \xi\rangle$ denote a coequaliser for $l^{\#}_{\alpha}, r^{\#}_{\alpha}$. Then, $\mathsf{U}_{\mathsf{C}} q$ is an epimorphism. This follows from q being the coequaliser of its kernel pair (see 7 of Proposition 2.1.43), together with U_{C} preserving kernel pairs and creating (and hence preserving) coequalisers of congruences. It then follows similarly to Proposition 3.1.32 that $\langle Q, \xi\rangle \models_{(\mathsf{C},\mathsf{F})} e$ and moreover, $\langle\langle Q, \xi\rangle, q\rangle$ is initial in $(\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha)^e$. □

Proposition 3.2.28. *Let* (C, F) *denote an abstract signature, such that* Alg(C, F) *has coequalisers. Also, let* $\langle A, \alpha\rangle$ *denote a* (C, F)*-algebra, let* E *denote an enumerable set of* (C, F)*-equations, and let* $(\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha)^E$ *denote the full subcategory of* $\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha$ *whose objects satisfy the equations in* E*. Then,* $(\mathsf{Alg}(\mathsf{C},\mathsf{F})\backslash\alpha)^E$ *has an initial object, which at the same time defines a* (C, F)*-homomorphic image of* $\langle A, \alpha\rangle$*.*

Proof. Similar to the proof of Proposition 3.1.33. □

Theorem 3.2.29. *Let* $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}', \mathsf{F}')$ *denote an abstract signature morphism, such that* $\mathsf{Alg}(\mathsf{C}', \mathsf{F}')$ *has coequalisers. Then,* $\mathsf{U}_\xi : \mathsf{Alg}(\mathsf{C}', \mathsf{F}') \to \mathsf{Alg}(\mathsf{C}, \mathsf{F})$ *has a left adjoint.*

Proof. Similar to the proof of Theorem 3.1.62. □

Theorem 3.2.30. *Let* $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}, E) \to (\mathsf{D}, \mathsf{G}, E')$ *denote an abstract algebraic specification morphism, such that* $\mathsf{Alg}(\mathsf{D}, \mathsf{G})$ *has coequalisers, and such that* E' *is enumerable. Then,* $\mathsf{U}_\xi{\restriction}_{\mathsf{Alg}(\mathsf{D},\mathsf{G},E')} : \mathsf{Alg}(\mathsf{D}, \mathsf{G}, E') \to \mathsf{Alg}(\mathsf{C}, \mathsf{F}, E)$ *has a left adjoint.*

Proof. Similar to the proof of Theorem 3.1.64. □

3.2.6 Expressiveness

Results regarding the expressiveness of abstract equations from the point of view of characterising varieties can be derived essentially by dualising the results in Section 3.1.6. Such a dualisation is possible due to the existence of characterisability results for varieties [BH76][24] dual to those formulated in [Kur00] for covarieties. In particular, a consequence of the results in [BH76] is that varieties (defined here in terms of the factorisation system $(\mathsf{U}_\mathsf{C}^{-1}(\mathsf{Epi}(\mathsf{C})), \mathsf{U}_\mathsf{C}^{-1}(\mathsf{Mono}(\mathsf{C})))$ obtained by lifting the factorisation system $(\mathsf{Epi}(\mathsf{C}), \mathsf{Mono}(\mathsf{C}))$ along U_C) are definable by homomorphic images (i.e. $\mathsf{U}_\mathsf{C}^{-1}(\mathsf{Epi}(\mathsf{C}))$-homomorphisms) with $\mathsf{U}_\mathsf{C}^{-1}(\mathsf{Epi}(\mathsf{C}))$-projective domains. Moreover, in the presence of free algebras, homomorphic images whose domains are given by free algebras over projective C-objects are sufficient to characterise varieties. This observation can be exploited to derive a characterisability result for varieties in terms of abstract equations.

A formula $\varphi : \langle F, \xi\rangle \to \langle B, \beta\rangle$ *holds* [BH76] in a (C, F)-algebra $\langle A, \alpha\rangle$ (written $\langle A, \alpha\rangle \models \varphi$) if and only if any (C, F)-algebra homomorphism $f : \langle F, \xi\rangle \to \langle A, \alpha\rangle$ factors through φ.

$$\begin{array}{ccc} \langle F, \xi\rangle & \xrightarrow{\varphi} & \langle B, \beta\rangle \\ {\scriptstyle f}\downarrow & \swarrow & \\ \langle A, \alpha\rangle & & \end{array}$$

Now given an abstract signature (C, F), any homomorphic image $\varphi : \langle F, \xi\rangle \to \langle B, \beta\rangle$ with $\langle F, \xi\rangle$ a free algebra over some projective C-object X induces an abstract equation (K, l, r), in such a way that $\langle A, \alpha\rangle \models \varphi$ is equivalent to $\langle A, \alpha\rangle \models (\mathsf{K}, l, r)$ for any (C, F)-algebra $\langle A, \alpha\rangle$. Specifically, if $h, k : C \to F$ define a kernel pair for $\mathsf{U}_\mathsf{C}\varphi$, then the functor $\mathsf{K} : \mathsf{C} \to \mathsf{C}$ is given by:

$$\mathsf{K}A = \coprod_{f:X\to A} C$$

for $A \in |\mathsf{C}|$, and:

$$\mathsf{K}a = [\iota_{a\circ f}]_{f:X\to A} : \coprod_{f:X\to A} C \to \coprod_{g:X\to B} C$$

for $(a : A \to B) \in \|\mathsf{C}\|$, while the natural transformations $l, r : \mathsf{K}\mathsf{U}_\mathsf{C} \Rightarrow \mathsf{U}_\mathsf{C}$ are given by:

$$l_{\langle C,\gamma\rangle} = \langle \mathsf{U}_\mathsf{C} f^\# \circ h\rangle_{f:X\to A}$$
$$r_{\langle C,\gamma\rangle} = \langle \mathsf{U}_\mathsf{C} f^\# \circ k\rangle_{f:X\to A}$$

[24] Alternatively, see [AHS90, Chapter 16].

for $\langle A, \alpha \rangle \in |\mathsf{Alg}(\mathsf{C}, \mathsf{F})|$.

Again, under the additional assumption that all C-epimorphisms are regular, one can show that $\langle A, \alpha \rangle \models \varphi$ is equivalent to $\langle A, \alpha \rangle \models (\mathsf{K}, l, r)$, for any (C, F)-algebra $\langle A, \alpha \rangle$. As a result, under assumptions dual to those of Corollary 3.1.67, (C, F)-varieties are definable by classes of (C, F)-equations. (Further assumptions need to be made about $\mathsf{Alg}(\mathsf{C}, \mathsf{F})$ in order to be able to characterise (C, F)-varieties by single (C, F)-equations.)

3.3 Specification of Combined Structures

Sections 3.1 and 3.2 have illustrated how coalgebra and algebra can be used to specify and reason about structures that involve observation and respectively computation. The resulting frameworks are here integrated in order to account for the relationship between computations and observations in structures having both a computational and an observational component.

Our approach builds on the functorial approach to operational semantics in [Tur96], where liftings of *syntactical monads* to categories of coalgebras of *behaviour endofunctors* are used to capture *well-behaved operational semantics*. Here we consider liftings of monads induced by abstract signatures to categories of coalgebras of abstract cosignatures. The algebras of the lifted monads interpret computations on the carriers of particular coalgebras, in such a way that bisimilarity on the underlying coalgebra is preserved by computations, while reachability in the resulting algebra is preserved by observations. Abstract equations and coequations are then used to formalise correctness properties of such algebras, with the associated notions of satisfaction abstracting away bisimilar and respectively unreachable behaviours. A dual approach, involving liftings of the comonads induced by abstract cosignatures to categories of algebras of abstract signatures is also obtained.

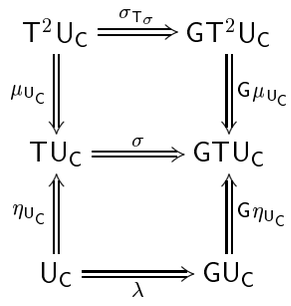
3.3.1 Lifted Signatures and their Models

We begin by noting that abstract signatures (C, F) induce monads (T, η, μ) on C having the property that $\mathsf{Alg}(\mathsf{T}) \simeq \mathsf{Alg}(\mathsf{C}, \mathsf{F})$ (as C has finite coproducts and colimits of ω-chains, while F preserves colimits of ω-chains, see Proposition 2.3.50), and that abstract signature morphisms $(\mathsf{U}, \xi) : (\mathsf{C}, \mathsf{F}) \to (\mathsf{C}', \mathsf{F}')$ induce monad morphisms $(\mathsf{U}, \nu) : (\mathsf{T}, \eta, \mu) \to (\mathsf{T}', \eta', \mu')$ having the property that $\mathsf{U}_\nu : \mathsf{Alg}(\mathsf{T}') \to \mathsf{Alg}(\mathsf{T})$ agrees with $\mathsf{U}_\xi : \mathsf{Alg}(\mathsf{C}', \mathsf{F}') \to \mathsf{Alg}(\mathsf{C}, \mathsf{F})$ (see Proposition 2.3.52).

Following [Tur96], we use natural transformations $\sigma : \mathsf{TU}_\mathsf{C} \Rightarrow \mathsf{GTU}_\mathsf{C}$, with $\mathsf{U}_\mathsf{C} : \mathsf{Coalg}(\mathsf{C}, \mathsf{G}) \to \mathsf{C}$ denoting the functor taking (C, G)-coalgebras to their carrier, and with (T, η, μ) denoting the monad induced by the abstract signature (C, F), to define liftings of the monad T to the category $\mathsf{Coalg}(\mathsf{C}, \mathsf{G})$. Such liftings are specified using *lifted signatures*, while translations between liftings are specified using *lifted signature morphisms*.

Definition 3.3.1. *An* **(abstract) lifted signature** *is a tuple* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$*, with* (C, G) *an abstract cosignature,* (C, F) *an abstract signature and* $\sigma : \mathsf{TU}_\mathsf{C} \Rightarrow \mathsf{GTU}_\mathsf{C}$ *a natural transformation, such that*

the following diagram commutes:

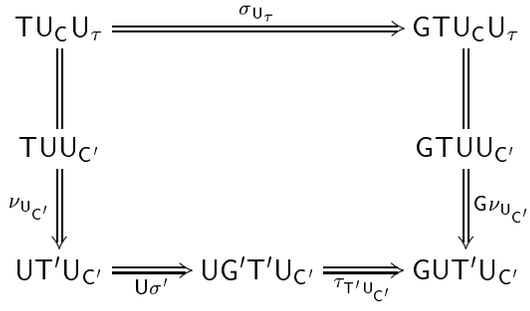


where the natural transformation $\lambda : \mathsf{U_C} \Rightarrow \mathsf{GU_C}$ *is given by:* $\lambda_{\langle C,\gamma\rangle} = \gamma$ *for* $\langle C, \gamma\rangle \in |\mathsf{Coalg(C, G)}|$*, while the functor* $\mathsf{T}_\sigma : \mathsf{Coalg(C, G)} \to \mathsf{Coalg(C, G)}$ *is given by:*

- $\mathsf{T}_\sigma\langle C, \gamma\rangle = \langle \mathsf{T}C, \sigma_\gamma\rangle$ *for* $\langle C, \gamma\rangle \in |\mathsf{Coalg(C, G)}|$
- $\mathsf{U_C T}_\sigma f = \mathsf{TU_C} f$ *for* $f \in \| \mathsf{Coalg(C, G)} \|$[25]

(and consequently $\mathsf{U_C T}_\sigma = \mathsf{TU_C}$*).*

An **(abstract) lifted signature morphism** *from* $(\mathsf{C, G, F}, \sigma)$ *to* $(\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma')$ *is a tuple* (U, τ, ξ) *with* $(\mathsf{U}, \tau) : (\mathsf{C, G}) \to (\mathsf{C}', \mathsf{G}')$ *an abstract cosignature morphism and* $(\mathsf{U}, \xi) : (\mathsf{C, F}) \to (\mathsf{C}', \mathsf{F}')$ *an abstract signature morphism, such that* $\xi : \mathsf{FU} \Rightarrow \mathsf{UF}'$ *is a natural isomorphism, and such that the following diagram commutes:*



where $(\mathsf{U}, \nu) : (\mathsf{T}, \eta, \mu) \to (\mathsf{T}', \eta', \mu')$ *is the monad morphism induced by the signature morphism* (U, ξ)[26]*. A lifted signature morphism* (U, τ, ξ) *is* **strong** *if and only if both* (U, τ) *and* (U, ξ) *are strong.*

Remark 3.3.2. The definition of λ immediately yields $\lambda_{\mathsf{T}_\sigma} = \sigma$.

The components of the natural transformation σ used to define a lifted signature $(\mathsf{C, G, F}, \sigma)$ define $(\mathsf{C, G})$-coalgebra structures on (the carriers of) the free T-algebras over (the carriers of) $(\mathsf{C, G})$-coalgebras. The constraints defining lifted signatures ensure that, for any $(\mathsf{C, G})$-coalgebra $\langle C, \gamma\rangle$,

[25] Naturality of σ ensures that $\mathsf{TU_C} f$ defines a $(\mathsf{C, G})$-coalgebra homomorphism.

[26] The fact that F and F′ preserve epimorphisms, together with the fact that U has a right adjoint (and therefore preserves colimits) ensure that ν is a natural isomorphism (see Remark 2.3.53).

the C-arrows $\eta_C : C \to \mathsf{T}C$ and $\mu_C : \mathsf{T}^2C \to C$ define (C, G)-coalgebra homomorphisms.

$$\begin{array}{ccc} \mathsf{T}^2C & \xrightarrow{\sigma_{\sigma_\gamma}} & \mathsf{G}\mathsf{T}^2C \\ {\scriptstyle \mu_C}\downarrow & & \downarrow{\scriptstyle \mathsf{G}\mu_C} \\ \mathsf{T}C & \xrightarrow{\sigma_\gamma} & \mathsf{G}\mathsf{T}C \\ {\scriptstyle \eta_C}\uparrow & & \uparrow{\scriptstyle \mathsf{G}\eta_C} \\ C & \xrightarrow[\gamma]{} & \mathsf{G}C \end{array}$$

This results in the tuple $(\mathsf{T}_\sigma, \eta, \mu)$ defining a monad on Coalg(C, G). An algebra of this monad is given by a (C, G)-coalgebra $\langle C, \gamma\rangle$ together with a (C, G)-coalgebra homomorphism $\alpha : \mathsf{T}_\sigma\langle C, \gamma\rangle \to \langle C, \gamma\rangle$, additionally satisfying: $\alpha \circ \eta_C = 1_C$ and $\alpha \circ \mu_C = \alpha \circ \mathsf{T}\alpha$. Equivalently, a T_σ-algebra is given by a C-object C carrying both a (C, G)-coalgebra structure $\langle C, \gamma\rangle$ and a T-algebra structure $\langle C, \alpha\rangle$ ((C, F)-algebra structure α'), such that α defines a (C, G)-coalgebra homomorphism from $\langle \mathsf{T}C, \sigma_\gamma\rangle$ to $\langle C, \gamma\rangle$.

$$\begin{array}{ccc} \mathsf{T}C & \xrightarrow{\sigma_\gamma} & \mathsf{G}\mathsf{T}C \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \mathsf{G}\alpha} \\ C & \xrightarrow[\gamma]{} & \mathsf{G}C \end{array}$$

Similarly, a T_σ-algebra homomorphism from $\langle\langle C, \gamma\rangle, \alpha\rangle$ to $\langle\langle D, \delta\rangle, \beta\rangle$ is given by a C-arrow $f : C \to D$ defining both a (C, G)-coalgebra homomorphism from $\langle C, \gamma\rangle$ to $\langle D, \delta\rangle$ and a T-algebra homomorphism from $\langle C, \alpha\rangle$ to $\langle D, \beta\rangle$ ((C, F)-algebra homomorphism from $\langle C, \alpha'\rangle$ to $\langle D, \beta'\rangle$).

The models of a lifted signature (C, G, F, σ) are taken to be the algebras of the lifted monad T_σ. The functor taking T_σ-algebras to their carrier is denoted $\mathsf{U}_{\mathsf{Coalg(C,G)}} : \mathsf{Alg}(\mathsf{T}_\sigma) \to \mathsf{Coalg(C, G)}$.

Remark 3.3.3. Given endofunctors F, G : C → C, the category Bialg(F, G) *of* (F, G)*-bialgebras* has objects given by tuples $\langle C, \alpha, \gamma\rangle$ with $\langle C, \alpha\rangle$ an F-algebra and $\langle C, \gamma\rangle$ a G-coalgebra, and arrows $f : \langle C, \alpha, \gamma\rangle \to \langle D, \beta, \delta\rangle$ given by C-arrows $f : C \to D$ defining both an F-algebra homomorphism from $\langle C, \alpha\rangle$ to $\langle D, \beta\rangle$ and a G-coalgebra homomorphism from $\langle C, \gamma\rangle$ to $\langle D, \delta\rangle$[27]. Then, for a lifted signature (C, G, F, σ), the category $\mathsf{Alg}(\mathsf{T}_\sigma)$ is isomorphic to the full subcategory of Bialg(F, G) whose objects $\langle C, \alpha', \gamma\rangle$ are such that $\gamma \circ \alpha = \mathsf{G}\alpha \circ \sigma_\gamma$ (where $\langle C, \alpha\rangle$ denotes the T-algebra induced by the F-algebra $\langle C, \alpha'\rangle$, as given by the proof of Proposition 2.3.50).

The constraints defining a lifted signature morphism (U, τ, ξ) : (C, G, F, σ) → (C′, G′, F′, σ') ensure that, for any (C′, G′)-coalgebra $\langle C', \gamma'\rangle$, the (C′, G′)-coalgebra structure induced by σ' on $\mathsf{T}'C'$ agrees with the (C, G)-coalgebra structure induced by σ on $\mathsf{T}\mathsf{U}C'$.

$$\begin{array}{ccccc} \mathsf{T}\mathsf{U}C' & & \xrightarrow{\sigma_{\tau_{C'} \circ \mathsf{U}\gamma'}} & & \mathsf{G}\mathsf{T}\mathsf{U}C' \\ {\scriptstyle \nu_{C'}}\downarrow & & & & \downarrow{\scriptstyle \mathsf{G}\nu_{C'}} \\ \mathsf{U}\mathsf{T}'C' & \xrightarrow[\mathsf{U}\sigma'_{\gamma'}]{} & \mathsf{U}\mathsf{G}'\mathsf{T}'C' & \xrightarrow[\tau_{\mathsf{T}'C'}]{} & \mathsf{G}\mathsf{U}\mathsf{T}'C' \end{array}$$

[27] The notion of bialgebra coincides with that of *algebra-coalgebra pair* introduced in [Mal96].

This results in lifted signature morphisms $(\mathsf{U}, \tau, \xi) : (\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma')$ inducing reduct functors $\mathsf{U}_{(\tau,\xi)} : \mathsf{Alg}(\mathsf{T}'_{\sigma'}) \to \mathsf{Alg}(\mathsf{T}_\sigma)$ (with $\mathsf{U}_{(\tau,\xi)}$ taking a $\mathsf{T}'_{\sigma'}$-algebra $\langle\langle C', \gamma'\rangle, \alpha'\rangle$ to the T_σ-algebra $\langle\langle \mathsf{U}C', \tau_{C'} \circ \mathsf{U}\gamma'\rangle, \mathsf{U}\alpha' \circ \nu_{C'}\rangle$). For, if the diagram:

$$\begin{array}{ccc} \mathsf{T}'C' & \xrightarrow{\sigma'_{\gamma'}} & \mathsf{G}'\mathsf{T}'C' \\ {\scriptstyle \alpha'}\downarrow & & \downarrow{\scriptstyle \mathsf{G}'\alpha'} \\ C' & \xrightarrow[\gamma']{} & \mathsf{G}'C' \end{array}$$

commutes in C', then the diagram:

$$\begin{array}{ccc} \mathsf{T}\mathsf{U}C' & \xrightarrow{\sigma_{\tau_{C'} \circ \mathsf{U}\gamma'}} & \mathsf{G}\mathsf{T}\mathsf{U}C' \\ {\scriptstyle \mathsf{U}\alpha' \circ \nu_{C'}}\downarrow & & \downarrow{\scriptstyle \mathsf{G}(\mathsf{U}\alpha' \circ \nu_{C'})} \\ \mathsf{U}C' & \xrightarrow[\tau_{C'} \circ \mathsf{U}\gamma']{} & \mathsf{G}\mathsf{U}C' \end{array}$$

commutes in C. This follows from the commutativity of each of the subdiagrams of the following diagram:

$$\begin{array}{ccccc} \mathsf{T}\mathsf{U}C' & & \xrightarrow{\sigma_{\tau_{C'} \circ \mathsf{U}\gamma'}} & & \mathsf{G}\mathsf{T}\mathsf{U}C' \\ {\scriptstyle \nu_{C'}}\downarrow & & & & \downarrow{\scriptstyle \mathsf{G}\nu_{C'}} \\ \mathsf{U}\mathsf{T}'C' & \xrightarrow[\mathsf{U}\sigma'_{\gamma'}]{} & \mathsf{U}\mathsf{G}'\mathsf{T}'C' & \xrightarrow{\tau_{\mathsf{T}'C'}} & \mathsf{G}\mathsf{U}\mathsf{T}'C' \\ {\scriptstyle \mathsf{U}\alpha'}\downarrow & & \downarrow{\scriptstyle \mathsf{U}\mathsf{G}'\alpha'} & & \downarrow{\scriptstyle \mathsf{G}\mathsf{U}\alpha'} \\ \mathsf{U}C' & \xrightarrow[\mathsf{U}\gamma']{} & \mathsf{U}\mathsf{G}'C' & \xrightarrow{\tau_{C'}} & \mathsf{G}\mathsf{U}C' \end{array}$$

(Commutativity of the top, bottom-left and respectively bottom-right subdiagrams follow from the definition of lifted signature morphisms, $\langle\langle C', \gamma'\rangle, \alpha'\rangle$ being a $\mathsf{T}'_{\sigma'}$-algebra, and respectively the naturality of τ.)

Remark 3.3.4. As a result, lifted signature morphisms also induce reduct functors between the subcategories of bialgebras associated to their target and source signatures (see Remark 3.3.3).

If $(\mathsf{U}, \tau, \xi) : (\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma')$ and $(\mathsf{U}', \tau', \xi') : (\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma') \to (\mathsf{C}'', \mathsf{G}'', \mathsf{F}'', \sigma'')$ are (strong) lifted signature morphisms, then so is $(\mathsf{U}\mathsf{U}', \tau_{\mathsf{U}'} \circ \mathsf{U}\tau', \mathsf{U}\xi' \circ \xi_{\mathsf{U}'})$. The quasi-category of lifted signatures and lifted signature morphisms is denoted LSign, while the quasi-category of lifted signatures and strong lifted signature morphisms is denoted SLSign.

Instances of the notions of lifted signature and lifted signature morphism will be used in Chapter 5 to specify the relationship between structure and functionality in object systems (with polynomial endofunctors of restricted forms being used in this sense).

Some properties of the categories of algebras of lifted signatures are stated in the following.

For a lifted signature $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$, the existence of finite limits in $\mathsf{Coalg}(\mathsf{C}, \mathsf{G})$ (see Corollary 3.1.13) results in the existence of finite limits in $\mathsf{Alg}(\mathsf{T}_\sigma)$.

Proposition 3.3.5. *Let* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ *denote a lifted signature. Then,* $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ *creates finite limits.*

Proof. The conclusion follows by Proposition 2.3.48. □

Corollary 3.3.6. *For a lifted signature* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$, $\mathsf{Alg}(\mathsf{T}_\sigma)$ *has finite limits and* $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ *preserves them.*

Proof. The conclusion follows by Corollary 2.3.49. □

In particular, $\mathsf{Alg}(\mathsf{T}_\sigma)$ has a final object, given by the T_σ-algebra $\langle\langle F, \zeta\rangle, !_{\sigma_\zeta}\rangle$, with $\langle F, \zeta\rangle$ denoting a final (C, G)-coalgebra, and with $!_{\sigma_\zeta} : \langle \mathsf{T}F, \sigma_\zeta\rangle \to \langle F, \zeta\rangle$ denoting the unique (C, G)-coalgebra homomorphism from $\langle \mathsf{T}F, \sigma_\zeta\rangle$ to $\langle F, \zeta\rangle$. The final T_σ-algebra provides an interpretation of arbitrary computations on abstract states.

Also, kernel pairs exist in $\mathsf{Alg}(\mathsf{T}_\sigma)$ and are created by $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$. This yields a T_σ-algebra structure on (C, G)-bisimilarity on the underlying coalgebra of a T_σ-algebra, in such a way that the coalgebra homomorphisms defining the bisimilarity relation become T_σ-algebra homomorphisms. (Recall from Remark 3.1.14 that bisimilarity is given by the kernel pair of the unique coalgebra homomorphism into the final coalgebra.) That is, (C, G)-bisimilarity on the underlying coalgebra of a T_σ-algebra is preserved by the T-algebra structure.

Remark 3.3.7. A similar approach to integrating algebraic and coalgebraic features, also based on [Tur96], is presented in [HK99] (see also [KH, Kur00]). The structures considered there, called (Ω, Ξ)-*structures*, are given by (Ω, Ξ)-bialgebras (see Remark 3.3.3) with $\Omega, \Xi : \mathsf{Set}^n \to \mathsf{Set}^n$, subject to additional constraints which ensure that Ξ-bisimilarity on the underlying coalgebras is compatible with the Ω-structure of the underlying algebras. This compatibility is shown to arise e.g. from *coinductive definitions*, defined as liftings of Ω to Ξ-coalgebras. Such liftings are less general than the ones considered here, as they correspond to natural transformations of type $\mathsf{FU}_\mathsf{C} \Rightarrow \mathsf{GFU}_\mathsf{C}$. In addition, the liftings considered in [HK99, KH, Kur00] are not required to be functorial, that is, no naturality condition is imposed to the arrows $\mathsf{F}C \to \mathsf{GF}C$ with $\langle C, \gamma\rangle \in |\mathsf{Coalg}(\mathsf{G})|$.

On the other hand, the existence of an initial object in $\mathsf{Coalg}(\mathsf{C}, \mathsf{G})$ (following from the existence of an initial object in C, see Corollary 2.4.10) results in the existence of an initial T_σ-algebra (see Remark 2.3.47).

Proposition 3.3.8. *Let* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ *denote a lifted signature. Then,* $\mathsf{Alg}(\mathsf{T}_\sigma)$ *has an initial object.*

An initial object in $\mathsf{Alg}(\mathsf{T}_\sigma)$ is given by the T_σ-algebra $\langle \mathsf{T}_\sigma\langle 0, !_{\mathsf{G}0}\rangle, \mu_0\rangle = \langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle$, where 0 denotes an initial C-object (and consequently $!_{\mathsf{G}0} : 0 \to \mathsf{G}0$ defines an initial (C, G)-coalgebra). The initial T_σ-algebra provides an observational structure on ground computations.

Now observe that each of the categories C, $\mathsf{Coalg}(\mathsf{C}, \mathsf{G})$, $\mathsf{Alg}(\mathsf{T}) \simeq \mathsf{Alg}(\mathsf{C}, \mathsf{F})$ and $\mathsf{Alg}(\mathsf{T}_\sigma)$ have kernel pairs, and that the functors U_C, U'_C (with $\mathsf{U}'_\mathsf{C} : \mathsf{Alg}(\mathsf{T}) \to \mathsf{C}$ taking T-algebras to their carrier) and $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ create as well as preserve kernel pairs (see Propositions 3.1.10, 3.2.7 and 3.3.5, and Corollaries 3.1.11, 3.2.8 and 3.3.6). Consequently, the functor $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})} : \mathsf{Alg}(\mathsf{T}_\sigma) \to \mathsf{Alg}(\mathsf{T})$ (taking

T_σ-algebras to their underlying T-algebras) also creates (and hence preserves) kernel pairs. For, if f denotes an arrow in $\mathsf{Alg}(\mathsf{T}_\sigma)$, and if g, h define a kernel pair for $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}f$, then the preservation of kernel pairs by U'_{C} together with the creation of kernel pairs by U_{C} and $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ yield a kernel pair g', h' for f in $\mathsf{Alg}(\mathsf{T}_\sigma)$ such that $\mathsf{U}'_{\mathsf{C}}\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}g' = \mathsf{U}'_{\mathsf{C}}g$ and $\mathsf{U}'_{\mathsf{C}}\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}h' = \mathsf{U}'_{\mathsf{C}}h$. The creation of kernel pairs by U'_{C} then yields $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}g' = g$ and $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}h' = h$. That is, $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}$ creates kernel pairs.

$$\begin{array}{ccc} \mathsf{Alg}(\mathsf{T}_\sigma) & \xrightarrow{\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}} & \mathsf{Alg}(\mathsf{T}) \\ \Big\downarrow{\scriptstyle \mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}} & & \Big\downarrow{\scriptstyle \mathsf{U}'_{\mathsf{C}}} \\ \mathsf{Coalg}(\mathsf{C},\mathsf{G}) & \xrightarrow[\mathsf{U}_{\mathsf{C}}]{} & \mathsf{C} \end{array}$$

On the other hand, quotients exist in C, $\mathsf{Coalg}(\mathsf{C},\mathsf{G})$ and $\mathsf{Alg}(\mathsf{T})$. Moreover, the functor U'_{C} preserves quotients (see Corollary 3.2.10), while the functor U_{C} creates them (see Proposition 3.1.20). This results in the existence of quotients in $\mathsf{Alg}(\mathsf{T}_\sigma)$ and in their preservation by $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ and $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}$.

Proposition 3.3.9. *Let* $(\mathsf{C},\mathsf{G},\mathsf{F},\sigma)$ *denote a lifted signature. Then,* $\mathsf{Alg}(\mathsf{T}_\sigma)$ *has quotients, while* $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ *and* $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}$ *preserve them.*

Proof. Let f denote an arrow in $\mathsf{Alg}(\mathsf{T}_\sigma)$, and let g, h define its kernel pair in $\mathsf{Alg}(\mathsf{T}_\sigma)$. Then, since $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$ and U_{C} preserve kernel pairs while U'_{C} creates kernel pairs, and since $\mathsf{U}_{\mathsf{C}}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} = \mathsf{U}'_{\mathsf{C}}\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}$, it follows that $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}g, \mathsf{U}_{\mathsf{Alg}(\mathsf{T})}h$ define a kernel pair of $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}f$ in $\mathsf{Alg}(\mathsf{T})$. Now let $q : \langle A,\alpha\rangle \to \langle B,\beta\rangle$ denote a quotient for $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}g, \mathsf{U}_{\mathsf{Alg}(\mathsf{T})}h$ in $\mathsf{Alg}(\mathsf{T})$ (see Corollary 3.2.10). Then, the preservation of quotients by U'_{C} (see Corollary 3.2.10) together with the creation of quotients by U_{C} (see Proposition 3.1.20) yield a quotient $p : \langle A,\gamma\rangle \to \langle B,\delta\rangle$ for $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}g, \mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}h$ in $\mathsf{Coalg}(\mathsf{C},\mathsf{G})$. Moreover, $\mathsf{U}_{\mathsf{C}}p = \mathsf{U}'_{\mathsf{C}}q = r$. To show that r defines an arrow in $\mathsf{Alg}(\mathsf{T}_\sigma)$, it suffices to show that $\langle\langle B,\delta\rangle,\beta\rangle \in |\mathsf{Alg}(\mathsf{T}_\sigma)|$ (as $\langle\langle A,\gamma\rangle,\alpha\rangle \in |\mathsf{Alg}(\mathsf{T}_\sigma)|$). But this follows from $\langle\langle A,\gamma\rangle,\alpha\rangle \in |\mathsf{Alg}(\mathsf{T}_\sigma)|$ together with p and q defining (C,G)-coalgebra and respectively T-algebra homomorphisms, r being an epimorphism (as it is a quotient), and T preserving epimorphisms (as F preserves epimorphisms, see also Remark 2.3.51):

$$\begin{aligned} \delta \circ \beta \circ \mathsf{T}r = &\quad (q \text{ is a T-algebra homomorphism}) \\ \delta \circ r \circ \alpha = &\quad (p \text{ is a } (\mathsf{C},\mathsf{G})\text{-coalgebra homomorphism}) \\ \mathsf{G}r \circ \gamma \circ \alpha = &\quad (\langle\langle A,\gamma\rangle,\alpha\rangle \in |\mathsf{Alg}(\mathsf{T}_\sigma)|) \\ \mathsf{G}r \circ \mathsf{G}\alpha \circ \sigma_\gamma = &\quad (q \text{ is a T-algebra homomorphism}) \\ \mathsf{G}\beta \circ \mathsf{GT}r \circ \sigma_\gamma = &\quad (r = \mathsf{U}_{\mathsf{C}}p, \text{ naturality of } \sigma) \\ \mathsf{G}\beta \circ \sigma_\delta \circ \mathsf{T}r & \end{aligned}$$

Hence, r defines a T_σ-algebra homomorphism $s : \langle\langle A,\gamma\rangle,\alpha\rangle \to \langle\langle B,\delta\rangle,\beta\rangle$. Moreover, s defines a quotient of f in $\mathsf{Alg}(\mathsf{T}_\sigma)$. (This follows from p and q defining quotients in $\mathsf{Coalg}(\mathsf{C},\mathsf{G})$ and respectively $\mathsf{Alg}(\mathsf{T})$.) □

Consequently, each T_σ-algebra homomorphism f has a unique factorisation of form $f = \iota \circ e$, with e defining a homomorphic image of the domain of f, and with ι defining a subalgebra of the codomain of f.

Proposition 3.3.10. *Let* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ *denote a lifted signature, let* f *denote a* T_σ*-algebra homomorphism with quotient* e*, and let* $f = \iota \circ e$ *denote the factorisation of* f *resulting from the universality of* e*. Then,* e *defines a homomorphic image of the domain of* f*, while* ι *defines a* T_σ*-subalgebra of the codomain of* f*.*

Proof. The fact that $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}e$ is a quotient results in it being an epimorphism. That is, e defines a homomorphic image of the domain of f. Also, the fact that $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}e$ is a quotient together with the regularity of C result in $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}\iota$ being a monomorphism (see Proposition 2.1.46). Hence, by Corollary 3.1.12, $\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}\iota$ is also a monomorphism. That is, ι defines a T_σ-subalgebra of the codomain of f. □

Remark 3.3.11. By taking the T_σ-algebra homomorphism f in the statement of Proposition 3.3.10 to be the unique homomorphism from the initial T_σ-algebra to an arbitrary one, one obtains, for each T_σ-algebra $\langle\langle C, \gamma\rangle, \alpha\rangle$, a T_σ-subalgebra of $\langle\langle C, \gamma\rangle, \alpha\rangle$ which is reachable.

3.3.2 Correctness Properties

Once the relationship between computations and observations has been specified by means of a lifted signature, abstract equations and coequations can be used to formalise correctness properties of the specified structures. Specifically, high-level requirements referring to the equivalence of computations can be captured by equations, whereas low-level requirements regarding system invariants can be captured by coequations. Since the interest is in the *observable* result of *ground* computations, the associated notions of satisfaction abstract away bisimilar and respectively unreachable behaviours.

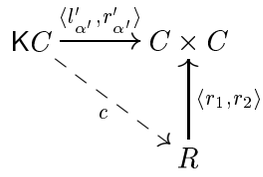
Definition 3.3.12. *Let* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ *denote a lifted signature. A* T_σ*-algebra* $\langle\langle C, \gamma\rangle, \alpha\rangle$ **satisfies** *a* (C, G)*-coequation* (K, l, r) **up to reachability** *(written* $\langle\langle C, \gamma\rangle, \alpha\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r)$*) if and only if* $l_\gamma \circ \mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}! = r_\gamma \circ \mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}!$*, with* $!$ *denoting the unique* T_σ*-algebra homomorphism from the initial* T_σ*-algebra to* $\langle\langle C, \gamma\rangle, \alpha\rangle$*.*

Also, $\langle\langle C, \gamma\rangle, \alpha\rangle$ **satisfies** *a* (C, F)*-equation* (K', l', r') **up to bisimulation** *(written* $\langle\langle C, \gamma\rangle, \alpha\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r')$*) if and only if* $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}!' \circ l'_{\alpha'} = \mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}!' \circ r'_{\alpha'}$*, with* $!'$ *denoting the unique* T_σ*-algebra homomorphism from* $\langle\langle C, \gamma\rangle, \alpha\rangle$ *to the final* T_σ*-algebra, and with* $\langle C, \alpha'\rangle$ *denoting the* (C, F)*-algebra induced by the* T*-algebra* $\langle C, \alpha\rangle$*.*

Remark 3.3.13. Standard satisfaction of coequations (respectively equations) by the underlying coalgebra (algebra) of a T_σ-algebra implies their satisfaction up to reachability (up to bisimulation) by the T_σ-algebra in question. That is, $\langle C, \gamma\rangle \models_{(\mathsf{C},\mathsf{G})} (\mathsf{K}, l, r)$ implies $\langle\langle C, \gamma\rangle, \alpha\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r)$, while $\langle C, \alpha'\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}', l', r')$ implies $\langle\langle C, \gamma\rangle, \alpha\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r')$. Moreover, if the underlying algebra (respectively coalgebra) is reachable (observable), then the converse also holds. For, in this case, $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}!$ (respectively $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}!'$) is an epimorphism (monomorphism).

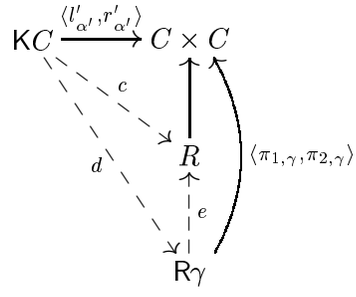
Remark 3.3.14. Since U_C preserves kernel pairs, it follows that $\langle\langle C, \gamma\rangle, \alpha\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r')$ holds if and

only if $\langle l'_{\alpha'}, r'_{\alpha'} \rangle$ factors through $\langle r_1, r_2 \rangle$:



with $\langle R, r_1, r_2 \rangle$ denoting (C, G)-bisimilarity on $\langle C, \gamma \rangle$.

The maximality of bisimilarity amongst the bisimulations on a given coalgebra yields a coinductive technique for proving the satisfaction of equations up to bisimulation. Specifically, proving that a (C, F)-equation holds, up to bisimulation, in a full subcategory $\mathcal{A}$ of $\mathsf{Alg}(\mathsf{T}_\sigma)$ can be reduced to exhibiting a generic (C, G)-bisimulation[28] $\langle \mathsf{R}, \pi_1, \pi_2 \rangle$ on $\mathsf{U}_{\mathsf{Coalg(C,G)}}(\mathcal{A})$, such that $\langle l'_{\alpha'}, r'_{\alpha'} \rangle$ factors through $\langle \pi_{1,\gamma}, \pi_{2,\gamma} \rangle$ for any $\langle \langle C, \gamma \rangle, \alpha \rangle \in |\mathcal{A}|$.



For this, in turn, yields a C-arrow $d : \mathsf{K}C \to \mathsf{R}\gamma$ such that $\langle \pi_{1,\gamma}, \pi_{2,\gamma} \rangle \circ d = \langle l'_{\alpha'}, r'_{\alpha'} \rangle$. Also, the maximality of (C, G)-bisimilarity on $\langle C, \gamma \rangle$ yields a C-arrow $e : \mathsf{R}\gamma \to R$. Then, c is taken to be $e \circ d$.

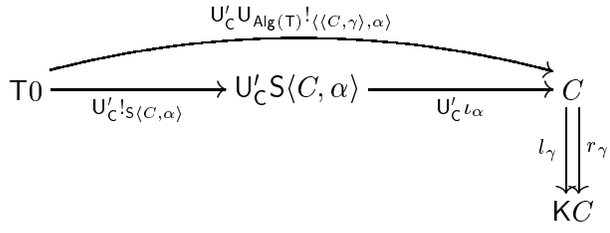
An instance of the abstract notion of equation will be used in Chapter 5 to capture the equivalence of computations yielding system states. The proof technique outlined above will then be used to reduce proving that two computations are observationally equal to exhibiting bisimulation relations which relate the results yielded by those computations (see Section 5.3.2).

Remark 3.3.15. A technique similar to the one outlined above is used in [Jac96a, Jac97] (see also Example 3.1.46) for proving the correctness of coalgebraic refinements. The setting in [Jac96a, Jac97] is mainly coalgebraic. The only algebraic feature refers to an initial state, which does not have to be fully specified in terms of the coalgebraic features. The technique introduced in [Jac96a, Jac97] involves two specifications (*abstract* and *concrete*), and is intended for proving that the concrete specification refines the abstract one. The existence of a refinement relation amounts to the behaviour of the initial state in any model of the abstract specification being *realisable* in any model of the concrete specification. Proving the correctness of a refinement then reduces to exhibiting a bisimulation relation between models of the abstract specification on the one hand and models of the concrete specification on the other, which, in addition, relates the initial state of each abstract model with some reachable state of each concrete model. This is similar to exhibiting

[28] See Definition 3.1.44.

a generic bisimulation, except that, instead of bisimulations *on* (classes of) models, one defines bisimulations between *different* (classes of) models, and instead of requiring the bisimulations to relate the interpretations of the lhs and rhs of an equation, one requires them to relate initial states with reachable states.

Similarly, proving that a (C, G)-coequation holds, up to reachability, in a full subcategory $\mathcal{A}$ of $\mathsf{Alg}(\mathsf{T}_\sigma)$ can be reduced to exhibiting a natural homomorphism $\iota : \mathsf{S} \Rightarrow \mathsf{I}_{\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}(\mathcal{A})}$ (defining, for instance, for each T_σ-algebra in $\mathcal{A}$, a T-subalgebra of its underlying T-algebra) for some endofunctor $\mathsf{S} : \mathsf{U}_{\mathsf{Alg}(\mathsf{T})}(\mathcal{A}) \to \mathsf{Alg}(\mathsf{T})$ (with $\mathsf{I}_{\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}(\mathcal{A})}$ defining the inclusion of $\mathsf{U}_{\mathsf{Alg}(\mathsf{T})}(\mathcal{A})$ into $\mathsf{Alg}(\mathsf{T})$), such that $l_\gamma \circ \mathsf{U}'_{\mathsf{C}}\iota_\alpha = r_\gamma \circ \mathsf{U}'_{\mathsf{C}}\iota_\alpha$ for any $\langle\langle C, \gamma\rangle, \alpha\rangle \in |\mathcal{A}|$.



An instance of the abstract notion of coequation will be used in Chapter 5 to specify invariants on the structure of state-based systems. The proof technique outlined above will then be used to reduce proving that a system invariant holds in reachable states to exhibiting subspaces of the state spaces which contain the reachable states and whose states satisfy the given invariant (see Section 5.3.2).

The following result further justifies the use of inductive and coinductive techniques in proving the satisfaction of coequations up to reachability, and respectively of equations up to bisimulation, by T_σ-algebras.

Proposition 3.3.16. *Let* (C, G, F, σ) *denote a lifted signature. Then, the following hold:*

1. *A* (C, G)*-coequation is satisfied (up to reachability) by the initial* T_σ*-algebra precisely when it is satisfied up to reachability by any* T_σ*-algebra.*
2. *A* (C, F)*-equation is satisfied (up to bisimulation) by the final* T_σ*-algebra precisely when it is satisfied up to bisimulation by any* T_σ*-algebra.*

Proof. The conclusion follows immediately using the naturality of (C, G)-observers and respectively of (C, F)-constructors. □

3.3.3 Institutions

The triviality of the algebraic component of lifted signature morphisms results in the notions of reachability associated to the source and target of such morphisms being essentially the same. This, in turn, yields an institution w.r.t. the satisfaction of coequations up to reachability by algebras of lifted signatures.

Theorem 3.3.17. (LSign, Alg, Coeqn, $\models^{\mathsf{r}}$) *is an institution.*

Proof. Let $(\mathsf{U}, \tau, \xi) : (\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma')$ denote a lifted signature morphism, let $\langle\langle C', \gamma'\rangle, \alpha'\rangle$ denote a $\mathsf{T}'_{\sigma'}$-algebra, and let (K, l, r) denote a (C, G)-coequation. Also, let $! : \mathsf{T}0 \to \mathsf{U}C'$ and $!' : \mathsf{T}'0' \to C'$ denote the C- and respectively C′-arrows underlying the unique T_σ- and $\mathsf{T}'_{\sigma'}$-algebra homomorphisms from the initial T_σ- and $\mathsf{T}'_{\sigma'}$-algebras to $\mathsf{U}_{(\tau,\xi)}\langle\langle C', \gamma'\rangle, \alpha'\rangle$ and $\langle\langle C', \gamma'\rangle, \alpha'\rangle$. Then, the following holds:

$$\begin{aligned}
\langle\langle C', \gamma'\rangle, \alpha'\rangle \models^{\mathsf{r}} \tau(\mathsf{K}, l, r) &\Leftrightarrow \text{(definition of } \models^{\mathsf{r}}) \\
l^\flat_{\mathsf{U}_\tau\gamma'} \circ !' = r^\flat_{\mathsf{U}_\tau\gamma'} \circ !' &\Leftrightarrow (\mathsf{U} \dashv \mathsf{R}) \\
l_{\mathsf{U}_\tau\gamma'} \circ \mathsf{U}!' = r_{\mathsf{U}_\tau\gamma'} \circ \mathsf{U}!' &\Leftrightarrow (\nu_{0'} \text{ is epi}) \\
l_{\mathsf{U}_\tau\gamma'} \circ \mathsf{U}!' \circ \nu_{0'} = r_{\mathsf{U}_\tau\gamma'} \circ \mathsf{U}!' \circ \nu_{0'} &\Leftrightarrow (\mathsf{U}0' = 0,\ \mathsf{U}!' \circ \nu_{0'} = !) \\
l_{\mathsf{U}_\tau\gamma'} \circ ! = r_{\mathsf{U}_\tau\gamma'} \circ ! &\Leftrightarrow \text{(definition of } \models^{\mathsf{r}}) \\
\mathsf{U}_{(\tau,\xi)}\langle\langle C', \gamma'\rangle, \alpha'\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r)
\end{aligned}$$

□

The requirement that the natural transformations $\nu : \mathsf{TU} \Rightarrow \mathsf{UT}'$ induced by the signature morphisms used to define lifted signature morphisms are natural epimorphisms is crucial to the proof of Theorem 3.3.17. Without this requirement, only one implication of the satisfaction condition can be shown to hold (as, in this case, the third equivalence in the above proof becomes an implication). A similar condition needs to be imposed to the cosignature morphisms used to define lifted signature morphisms in order to obtain an institution w.r.t. the satisfaction of equations up to bisimulation by algebras of lifted signatures.

Definition 3.3.18. *A lifted signature morphism* (U, τ, ξ) *is* **horizontal** *if and only if* (U, τ) *is horizontal.*

The quasi-category of lifted signatures and horizontal lifted signature morphisms is denoted HLSign.

The notions of bisimilarity associated to the source and target of horizontal lifted signature morphisms are essentially the same (see Remark 3.1.54). This, in turn, yields an institution.

Theorem 3.3.19. $(\mathsf{HLSign}, \mathsf{Alg}\restriction_{\mathsf{HLSign}}, \mathsf{Eqn}\restriction_{\mathsf{HLSign}}, \models^{\mathsf{b}})$ *is an institution.*

Proof. Similar to the proof of Theorem 3.3.17. □

The conditions defining (horizontal) lifted signature morphisms might appear restrictive at first, since such morphisms are unable to capture development steps which specialise (either the observational or) the computational features of existing components. However, these conditions simply state that the notions of observation and computation must be established *before* beginning to reason about the observational equivalence of computations or the satisfaction of state invariants in reachable states (as changing these notions would usually result in different equivalences holding between computations, and respectively in different invariants being satisfied in reachable states).

Due to the presence of semantic constraints on the algebras of lifted signatures (namely of constraints relating their underlying algebraic and coalgebraic structures), the notions of specification

associated to the institutions $(\mathsf{LSign}, \mathsf{Alg}, \mathsf{Coeqn}, \models^{\mathsf{r}})$ and $(\mathsf{HLSign}, \mathsf{Alg}\restriction_{\mathsf{HLSign}}, \mathsf{Eqn}\restriction_{\mathsf{HLSign}}, \models^{\mathsf{b}})$ play a less important rôle than, for instance, the specifications of $(\mathsf{Cosign}, \mathsf{Coalg}, \mathsf{Coeqn}, \models)$ or those of $(\mathsf{Sign}, \mathsf{Alg}, \mathsf{Eqn}, \models)$ (see Theorems 3.1.49 and 3.2.19). For, restricting the category of algebras of a lifted signature $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ to algebras satisfying some given coequations up to reachability yields a category of models which is:

1. empty, if the final T_σ-algebra does not satisfy the coequations up to reachability,
2. the same as $\mathsf{Alg}(\mathsf{T}_\sigma)$, if the (C, G)-coalgebra underlying the initial T_σ-algebra already satisfies the coequations,
3. a full subcategory of $\mathsf{Alg}(\mathsf{T}_\sigma)$, otherwise. (This subcategory contains the initial, but not the final T_σ-algebra.)

Similarly, restricting the category of algebras of $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ to algebras satisfying some given equations up to bisimulation yields a category of models which is:

1. empty, if the initial T_σ-algebra does not satisfy the equations up to bisimulation,
2. the same as $\mathsf{Alg}(\mathsf{T}_\sigma)$, if the (C, F)-algebra underlying the final T_σ-algebra satisfies the equations,
3. a full subcategory of $\mathsf{Alg}(\mathsf{T}_\sigma)$, otherwise. (This subcategory contains the final, but not the initial T_σ-algebra.)

3.3.4 Compositionality Results

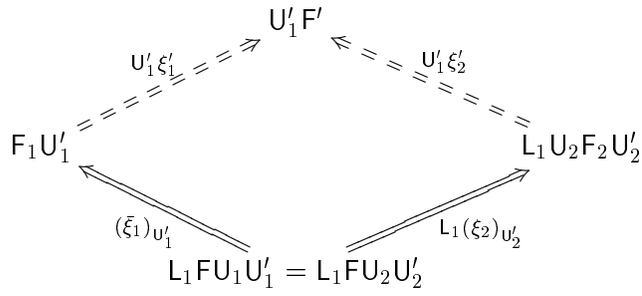
This section combines the compositionality results in Sections 3.1.4 and 3.2.4 in order to derive similar results for the combined framework.

Theorem 3.3.20. SLSign *is finitely cocomplete.*

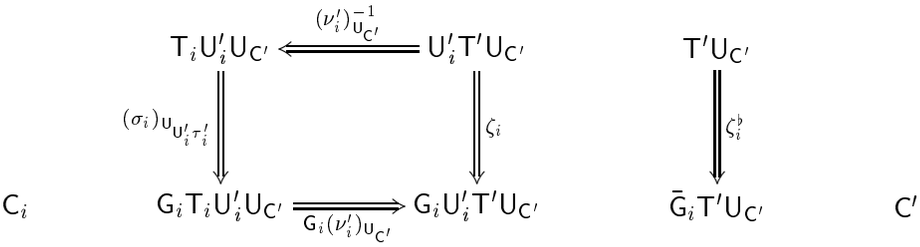
Proof. An initial object in SLSign is given by the lifted signature $(1, \mathsf{Id}_1, \mathsf{Id}_1, 1)$, with 1 denoting a category with one object and one arrow, and with $1 : \mathsf{U}_1 \Rightarrow \mathsf{U}_1$ denoting the natural transformation whose only component is given by the identity arrow (where $\mathsf{U}_1 : \mathsf{Coalg}(1, \mathsf{Id}_1) \to 1$ takes $(1, \mathsf{Id}_1)$-coalgebras to their carrier).

Pushouts in SLSign are constructed from the pushouts of the underlying diagrams in $\mathsf{SCosign}$ and SSign as follows. Let $(\mathsf{U}_i, \tau_i, \xi_i) : (\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{C}_i, \mathsf{G}_i, \mathsf{F}_i, \sigma_i)$ with $i = 1, 2$ denote lifted signature morphisms, and let $(\mathsf{U}'_i, \mathsf{U}'_i\tau'_i) : (\mathsf{C}_i, \mathsf{G}_i) \to (\mathsf{C}', \mathsf{G}')$ with $i = 1, 2$ and $(\mathsf{U}'_i, \mathsf{U}'_i\xi'_i) : (\mathsf{C}_i, \mathsf{F}_i) \to (\mathsf{C}', \mathsf{F}')$ with $i = 1, 2$ denote the pushouts of (U_1, τ_1) and (U_2, τ_2) in $\mathsf{SCosign}$ and respectively of (U_1, ξ_1) and (U_2, ξ_2) in SSign (see the proofs of Theorems 3.1.58 and 3.2.24). Then, the natural transformations $\mathsf{U}'_i\xi'_i : \mathsf{F}_i\mathsf{U}'_i \Rightarrow \mathsf{U}'_i\mathsf{F}'$ with $i = 1, 2$ are natural isomorphisms. For, the preservation of pushouts by U'_1

results in the diagram:

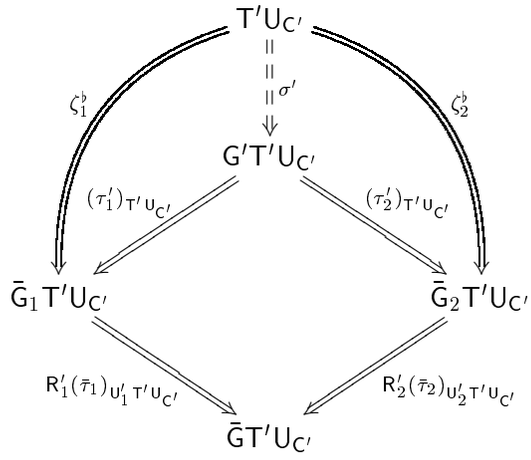


defining a pushout in C_1. (The fact that $U'_1L'_2 = L_1U_2$ is also used here.) This, together with ξ_2 being a natural isomorphism result in $U'_1\xi'_1$ also being a natural isomorphism. The fact that $U'_2\xi'_2$ is a natural isomorphism follows in a similar way. Then, Remark 2.3.53 results in the natural transformations $\nu'_1 : T_1U'_1 \Rightarrow U'_1T'$ and $\nu'_2 : T_2U'_2 \Rightarrow U'_2T'$ defining the monad morphisms induced by the signature morphisms $(U'_1, U'_1\xi'_1)$ and respectively $(U'_2, U'_2\xi'_2)$ (see Proposition 2.3.52) also being natural isomorphisms. Next, one can exploit the couniversality of (G', τ'_1, τ'_2) (see the proof of Theorem 3.1.58) to define a natural transformation $\sigma' : T'U_{C'} \Rightarrow G'T'U_{C'}$ which makes $(U'_i, U'_i\tau'_i, U'_i\xi'_i)$ a lifted signature morphism from $(C_i, G_i, F_i, \sigma_i)$ to (C', G', F', σ'), for $i = 1, 2$. First, the fact that $U_1U'_1\xi'_1 \circ (\xi_1)_{U'_1} = U_2U'_2\xi'_2 \circ (\xi_2)_{U'_2}$ (following from the construction of ξ'_1, ξ'_2, see the proof of Theorem 3.2.24) can be used to show that $U_1\nu'_1 \circ (\nu_1)_{U'_1} = U_2\nu'_2 \circ (\nu_2)_{U'_2}$. Specifically, this follows from $U_1\nu'_1 \circ (\nu_1)_{U'_1} \circ (q_i)_{U_1U'_1} = U_2\nu'_2 \circ (\nu_2)_{U'_2} \circ (q_i)_{U_2U'_2}$ for $i = 0, 1, \ldots$ (where the natural transformations $q_i : F_i \Rightarrow T$ for $i = 0, 1, \ldots$ have the same denotation as in the proof of Proposition 2.3.50), using the colimiting property of T. (The proof of $U_1\nu'_1 \circ (\nu_1)_{U'_1} \circ (q_i)_{U_1U'_1} = U_2\nu'_2 \circ (\nu_2)_{U'_2} \circ (q_i)_{U_2U'_2}$ uses the fact that $U_1\xi'_{1,i} \circ (\xi_{1,i})_{U'_1} = U_2\xi'_{2,i} \circ (\xi_{2,i})_{U'_2}$, where the natural transformations $\xi_{j,i} : F_iU_i \Rightarrow U_iF_{j,i}$ and respectively $\xi'_{j,i} : F_{j,i}U'_i \Rightarrow U'_iF'_i$ for $j = 1, 2$ and $i = 0, 1, \ldots$ are defined similarly to the natural transformations $\xi_i : F_iU \Rightarrow UF'_i$ for $i = 0, 1, \ldots$ in the proof of Proposition 2.3.50. This, in turn, follows by induction on i.) Then, the natural transformations $\zeta_i = G_i(\nu'_i)_{U_{C'}} \circ (\sigma_i)_{U_{U'_i\tau'_i}} \circ (\nu'_i)^{-1}_{U_{C'}} : U'_iT'U_{C'} \Rightarrow G_iU'_iT'U_{C'}$ for $i = 1, 2$ yield natural transformations $\zeta^\flat_i : T'U_{C'} \Rightarrow \bar{G}_iT'U_{C'}$ (where $\bar{G}_i = R'_iG_iU'_i$) for $i = 1, 2$.



Moreover, the following holds: $R'_1(\bar{\tau}_1)_{U'_1T'U_{C'}} \circ \zeta^\flat_1 = R'_2(\bar{\tau}_2)_{U'_2T'U_{C'}} \circ \zeta^\flat_2$ (where the natural transformations $\bar{\tau}_i : G_i \Rightarrow R_iGU_i$ with $i = 1, 2$ are defined similarly to the natural transformations $\bar{\eta}_i : F_i \Rightarrow R_iFU_i$ with $i = 1, 2$ in the proof of Theorem 3.1.58). This follows from $U_1U'_1(R'_1(\bar{\tau}_1)_{U'_1T'U_{C'}} \circ \zeta^\flat_1) = U_2U'_2(R'_2(\bar{\tau}_2)_{U'_2T'U_{C'}} \circ \zeta^\flat_2)$, together with the fact that $R'_1R_1 = R'_2R_2$ is a right adjoint, right inverse to $U_1U'_1 = U_2U'_2$. (The fact that $U_1\nu'_1 \circ (\nu_1)_{U'_1} = U_2\nu'_2 \circ (\nu_2)_{U'_2}$, together with the constraints defining the lifted signature morphisms (U_1, τ_1, ξ_1) and respectively (U_2, τ_2, ξ_2) are used to show

that $U_1U_1'(R_1'(\bar{\tau}_1)_{U_1'T'U_{C'}} \circ \zeta_1^\flat) = U_2U_2'(R_2'(\bar{\tau}_2)_{U_2'T'U_{C'}} \circ \zeta_2^\flat)$.)



The couniversality of (G', τ_1', τ_2') now yields a unique natural transformation $\sigma' : T'U_{C'} \Rightarrow G'T'U_{C'}$ satisfying $(\tau_i')_{T'U_{C'}} \circ \sigma' = \zeta_i^\flat$ for $i = 1, 2$. This, together with the definition of ζ_i yield $U_i'(\tau_i')_{T'U_{C'}} \circ U_i'\sigma' \circ (\nu_i')_{U_{C'}} = G_i(\nu_i')_{U_{C'}} \circ (\sigma_i)_{U_{U_i'\tau_i'}}$ for $i = 1, 2$. That is, $(U_i', U_i'\tau_i', U_i'\xi_i')$ defines a lifted signature morphism from $(C_i, G_i, F_i, \sigma_i)$ to (C', G', F', σ'), for $i = 1, 2$. This concludes the proof. □

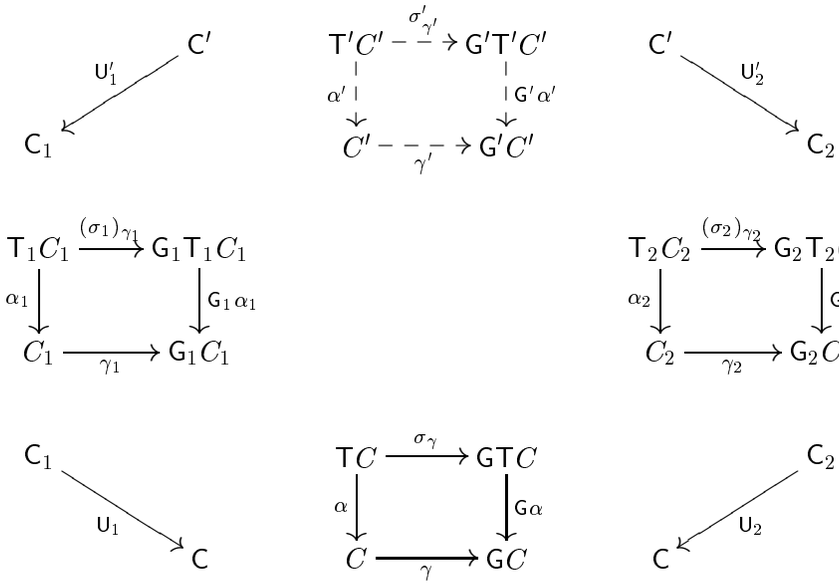
Remark 3.3.21. Theorem 3.3.20 and Corollary 2.2.6 result in the subcategories of the categories of specifications associated to the institutions $(\mathsf{LSign}, \mathsf{Alg}, \mathsf{Coeqn}, \models^{\mathsf{r}})$ and $(\mathsf{HLSign}, \mathsf{Alg}\restriction_{\mathsf{HLSign}}, \mathsf{Eqn}\restriction_{\mathsf{HLSign}}, \models^{\mathsf{b}})$ (see Theorems 3.3.17 and 3.3.19) whose specification morphisms are underlain by strong lifted signature morphisms also being finitely cocomplete.

The following also holds.

Theorem 3.3.22. *The functor* $\mathsf{Alg}\restriction_{\mathsf{SLSign}} : \mathsf{SLSign} \to \mathsf{CAT}^{\mathsf{op}}$ *preserves finite colimits.*

Proof. It suffices to prove that $\mathsf{Alg}\restriction_{\mathsf{SLSign}}$ preserves the initial object and pushouts. First, the category of algebras of the lifted signature $(1, \mathsf{Id}_1, \mathsf{Id}_1, 1)$ has one object and one arrow, and therefore is a final object in CAT. Also, given the pushout diagram in SLSign obtained in the proof of Theorem 3.3.20, the fact that its image under $\mathsf{Alg}\restriction_{\mathsf{SLSign}}$ defines a pullback in CAT follows from any span $\langle H_1, H_2 \rangle$ with source D on $\mathsf{Alg}(C_1, G_1, F_1, \sigma_1), \mathsf{Alg}(C_2, G_2, F_2, \sigma_2)$ in CAT, additionally satisfying $U_{(\tau_1,\xi_1)}H_1 = U_{(\tau_2,\xi_2)}H_2$, inducing a unique functor $H : D \to \mathsf{Alg}(C', G', F', \sigma')$ satisfying $U_{(U_i'\tau_i, U_i'\xi_i)}H = H_i$ for $i = 1, 2$. For $D \in |D|$, HD is constructed as follows. Say $H_iD = \langle\langle C_i, \gamma_i\rangle, \alpha_i\rangle$ for $i = 1.2$. Then, Theorem 3.1.60 yields a unique (C', F')-coalgebra $\langle C', \gamma'\rangle$ satisfying $U_{U_i'\tau_i'}\langle C', \gamma'\rangle = \langle C_i, \gamma_i\rangle$ for $i = 1, 2$. (The fact that pushouts in SLSign are constructed from pushouts in SCosign is used here.) Moreover, the C_1- and C_2-arrows $\alpha_1 \circ (\nu_1')_{C'}^{-1} : U_1'T'C' \to C_1$ and respectively $\alpha_2 \circ (\nu_2')_{C'}^{-1} : U_2'T'C' \to C_2$ are such that their images under U_1 and respectively U_2 coincide. (The fact that $U_1\nu_1' \circ (\nu_1)_{U_1'} = U_2\nu_2' \circ (\nu_2)_{U_2'}$, that each of these is an isomorphism, and that $U_{\nu_1}\alpha_1 = U_{\nu_2}\alpha_2$ are used to show this.) This yields a unique C'-arrow $\alpha' : T'C' \to C'$

satisfying $\mathsf{U}'_i\alpha' = \alpha_i \circ (\nu'_i)^{-1}_{C'}$, that is, $\mathsf{U}_{\nu'_i}\alpha' = \alpha_i$, for $i = 1, 2$.



Next, the couniversality of (G', τ'_1, τ'_2) can be used to show that $\gamma' \circ \alpha' = \mathsf{G}'\alpha' \circ \sigma'_{\gamma'}$. This follows from $(\tau'_i)_{C'} \circ \gamma' \circ \alpha' = (\tau'_i)_{C'} \circ \mathsf{G}'\alpha' \circ \sigma'_{\gamma'}$ for $i = 1, 2$, which, in turn, follows from:

$$\begin{aligned}
\mathsf{U}'_i((\tau'_i)_{C'} \circ \gamma' \circ \alpha') &= \quad (\mathsf{U}_{\mathsf{U}'_i\tau'_i}\langle C', \gamma'\rangle = \langle C, \gamma\rangle)\\
\gamma_i \circ \mathsf{U}'_i\alpha' &= \quad (\text{definition of } \alpha')\\
\gamma_i \circ \alpha_i \circ (\nu'_i)^{-1}_{C'} &= \quad (\langle\langle C_i, \gamma_i\rangle, \alpha_i\rangle \text{ is a } \mathsf{T}_{\sigma_i}\text{-algebra})\\
\mathsf{G}_i\alpha_i \circ (\sigma_i)_{\gamma_i} \circ (\nu'_i)^{-1}_{C'} &= \quad (\text{definition of } \alpha')\\
\mathsf{G}_i\mathsf{U}'_i\alpha' \circ \mathsf{G}_i(\nu'_i)_{C'} \circ (\sigma_i)_{\gamma_i} \circ (\nu'_i)^{-1}_{C'} &= \quad ((\mathsf{U}'_i\tau'_i, \mathsf{U}'_i\xi'_i) \text{ is a lifted signature morphism})\\
\mathsf{G}_i\mathsf{U}'_i\alpha' \circ \mathsf{U}'_i(\tau'_i)_{\mathsf{T}'C'} \circ \mathsf{U}'_i\sigma'_{\gamma'} &= \quad (\text{naturality of } \mathsf{U}'_i\tau'_i)\\
\mathsf{U}'_i((\tau'_i)_{C'} \circ \mathsf{G}'\alpha' \circ \sigma'_{\gamma'})
\end{aligned}$$

using the fact that R'_i is a right adjoint, right inverse to U'_i, for $i = 1, 2$. Hence, $\langle\langle C', \gamma'\rangle, \alpha'\rangle \in |\mathsf{Alg}(\mathsf{T}'_{\sigma'})|$ and moreover, $\mathsf{U}_{(\mathsf{U}'_i\tau'_i, \mathsf{U}'_i\xi'_i)}\langle\langle C', \gamma'\rangle, \alpha'\rangle = \langle\langle C_i, \gamma_i\rangle, \alpha_i\rangle$. The functoriality of $D \mapsto \langle\langle C', \gamma'\rangle, \alpha'\rangle$ follows similarly to the functoriality of $D \mapsto \langle C', \gamma'\rangle$ in the proof of Theorem 3.1.60. This concludes the proof. □

3.3.5 Semantic Constructions

This section is devoted to providing suitable denotations for lifted signatures and lifted signature morphisms.

We first consider lifted signatures. Neither final nor initial algebras provide suitable denotations for lifted signatures, as final algebras are not reachable, while the underlying coalgebras of initial algebras are not observable. However, according to Proposition 3.3.10, the codomain of the quotient of the unique homomorphism from the initial to the final T_σ-algebra is reachable, while its underlying coalgebra is observable. Moreover, this algebra has the property that it satisfies (in the standard

sense) precisely those equations which are satisfied up to bisimulation by the initial algebra, and precisely those coequations which are satisfied up to reachability by the final coalgebra.

Proposition 3.3.23. *Let* $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$ *denote a lifted signature, let* $! : \langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle \to \langle\langle F, \zeta\rangle, !_{\sigma_\zeta}\rangle$ *denote the unique* T_σ*-algebra homomorphism from the initial* T_σ*-algebra to the final one, and let* $e : \langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle \to \langle\langle R, \gamma\rangle, \alpha\rangle$ *denote the quotient of* $!$*. Then, the following hold:*

1. *If* (K, l, r) *denotes a* (C, G)*-coequation, then* $\langle\langle F, \zeta\rangle, !_{\sigma_\zeta}\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r)$ *is equivalent to* $\langle\langle R, \gamma\rangle, \alpha\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r)$*, as well as to* $\langle R, \gamma\rangle \models_{(\mathsf{C},\mathsf{G})} (\mathsf{K}, l, r)$*.*
2. *If* (K', l', r') *denotes a* (C, F)*-equation, then* $\langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r')$ *is equivalent to* $\langle\langle R, \gamma\rangle, \alpha\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r')$*, as well as to* $\langle R, \alpha\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}', l', r')$*.*

Proof. Let $\iota : \langle\langle R, \gamma\rangle, \alpha\rangle \to \langle\langle F, \zeta\rangle, !_{\sigma_\zeta}\rangle$ denote the unique T_σ-algebra homomorphism resulting from the universality of e. It then follows by Proposition 3.3.10 that $\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e$ is an epimorphism, while $\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota$ is a monomorphism.

Then, 1 follows from:

$$
\begin{aligned}
\langle\langle F, \zeta\rangle, !_{\sigma_\zeta}\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r) &\;\Leftrightarrow\; (\text{definition of } \models^{\mathsf{r}}) \\
l_{!_{\sigma_\zeta}} \circ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} ! = r_{!_{\sigma_\zeta}} \circ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} ! &\;\Leftrightarrow\; (! = \iota \circ e,\ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e \text{ is epi}) \\
l_{!_{\sigma_\zeta}} \circ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota = r_{!_{\sigma_\zeta}} \circ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota &\;\Leftrightarrow\; (\text{naturality of } l, r) \\
\mathsf{K U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota \circ l_\gamma = \mathsf{K U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota \circ r_\gamma &\;\Leftrightarrow\; (\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} \iota \text{ is mono, } \mathsf{K} \text{ preserves monos}) \\
l_\gamma = r_\gamma &\;\Leftrightarrow\; (\text{definition of } \models) \\
\langle R, \gamma\rangle \models_{(\mathsf{C},\mathsf{G})} (\mathsf{K}, l, r) &\;\Leftrightarrow\; (\langle R, \alpha\rangle \text{ is reachable, Remark 3.3.13}) \\
\langle\langle R, \gamma\rangle, \alpha\rangle \models^{\mathsf{r}} (\mathsf{K}, l, r) &
\end{aligned}
$$

Also, 2 follows from:

$$
\begin{aligned}
\langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r') &\;\Leftrightarrow\; (\text{definition of } \models^{\mathsf{b}}) \\
\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} ! \circ l'_{\mu_0} = \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} ! \circ r'_{\mu_0} &\;\Leftrightarrow\; (! = \iota \circ e,\ \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{F})} \iota \text{ is mono}) \\
\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e \circ l'_{\mu_0} = \mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e \circ r'_{\mu_0} &\;\Leftrightarrow\; (\text{naturality of } l, r) \\
l'_\alpha \circ \mathsf{K' U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e = r'_\alpha \circ \mathsf{K' U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e &\;\Leftrightarrow\; (\mathsf{U_C U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})} e \text{ is epi, } \mathsf{K}' \text{ preserves epis}) \\
l'_\alpha = r'_\alpha &\;\Leftrightarrow\; (\text{definition of } \models) \\
\langle R, \alpha\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{K}', l', r') &\;\Leftrightarrow\; (\langle R, \gamma\rangle \text{ is observable, Remark 3.3.13}) \\
\langle\langle R, \gamma\rangle, \alpha\rangle \models^{\mathsf{b}} (\mathsf{K}', l', r') &
\end{aligned}
$$

□

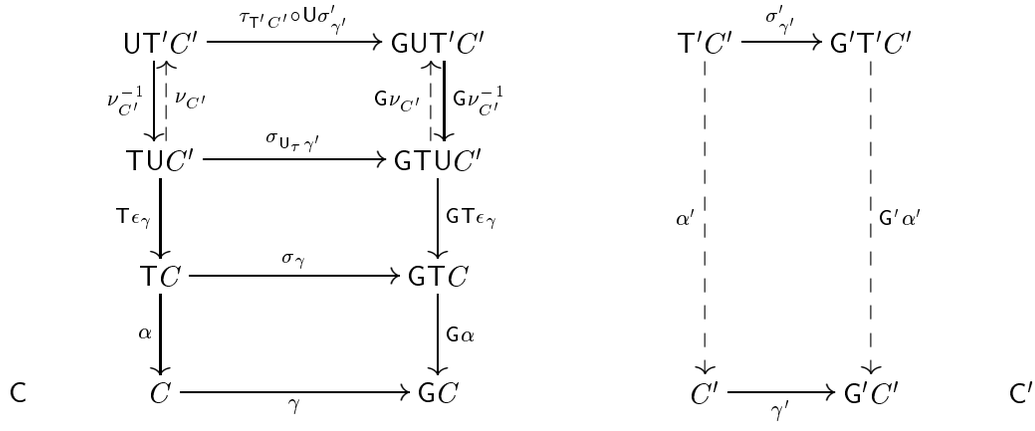
The above property of the codomain of the quotient of the unique homomorphism from the initial to the final T_σ-algebra makes this algebra a suitable denotation for the lifted signature $(\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma)$.

We now consider lifted signature morphisms. A consequence of their definition (and also one of the reasons for this particular definition) is the existence of cofree constructions along such morphisms.

As one would expect, the cofree construction induced by a lifted signature morphism builds on the cofree construction induced by its underlying cosignature morphism.

Theorem 3.3.24. *Let* $(\mathsf{U}, \tau, \xi) : (\mathsf{C}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{C}', \mathsf{G}', \mathsf{F}', \sigma')$ *denote a lifted signature morphism. Then,* $\mathsf{U}_{(\tau,\xi)} : \mathsf{Alg}(\mathsf{T}'_{\sigma'}) \to \mathsf{Alg}(\mathsf{T}_\sigma)$ *has a right adjoint.*

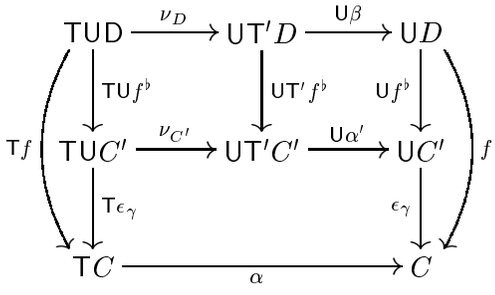
Proof. Let $\langle\langle C, \gamma\rangle, \alpha\rangle \in |\mathsf{Alg}(\mathsf{T}_\sigma)|$, let $\epsilon_\gamma : \mathsf{U}_\tau\langle C', \gamma'\rangle \to \langle C, \gamma\rangle$ denote a couniversal arrow from $\mathsf{U}_\tau : \mathsf{Coalg}(\mathsf{C}', \mathsf{G}') \to \mathsf{Coalg}(\mathsf{C}, \mathsf{G})$ to $\langle C, \gamma\rangle$, and let $\alpha' : \langle \mathsf{T}'C', \sigma'_{\gamma'}\rangle \to \langle C', \gamma'\rangle$ denote the unique (co)extension of $\alpha \circ \mathsf{T}\epsilon_\gamma \circ \nu^{-1}_{C'} : \mathsf{U}_\tau\langle \mathsf{T}'C', \sigma'_{\gamma'}\rangle \to \langle C, \gamma\rangle$ to a $(\mathsf{C}', \mathsf{G}')$-coalgebra homomorphism.



(Commutativity of the inner, and therefore also outer, upper-left square of the above diagram is a consequence of the definition of lifted signature morphisms. Also, commutativity of the middle-left square is a consequence of the naturality of σ. Finally, commutativity of the bottom-left square follows from $\langle\langle C, \gamma\rangle, \alpha\rangle$ being a T_σ-algebra. Consequently, $\alpha \circ \mathsf{T}\epsilon_\gamma \circ \nu^{-1}_{C'}$ defines a (C, G)-coalgebra homomorphism from $\langle \mathsf{UT}'C', \tau_{\mathsf{T}'C'} \circ \mathsf{U}\sigma'_{\gamma'}\rangle$ to $\langle C, \gamma\rangle$.) Then, the fact that α' defines a $(\mathsf{C}', \mathsf{G}')$-coalgebra homomorphism from $\langle \mathsf{T}'C', \sigma'_{\gamma'}\rangle$ to $\langle C', \gamma'\rangle$ translates to $\gamma' \circ \alpha' = \mathsf{G}'\alpha' \circ \sigma'_{\gamma'}$, which amounts to $\langle\langle C', \gamma'\rangle, \alpha'\rangle$ defining a $\mathsf{T}'_{\sigma'}$-algebra. Also, the definition of α' gives $\alpha \circ \mathsf{T}\epsilon_\gamma \circ \nu^{-1}_{C'} = \epsilon_\gamma \circ \mathsf{U}\alpha'$, that is, $\alpha \circ \mathsf{T}\epsilon_\gamma = \epsilon_\gamma \circ \mathsf{U}\alpha' \circ \nu_{C'}$. This makes ϵ_γ into a T-algebra homomorphism from $\mathsf{U}_\nu\langle C', \alpha'\rangle$ to $\langle C, \alpha\rangle$, and therefore also into a T_σ-algebra homomorphism from $\mathsf{U}_{(\tau,\xi)}\langle\langle C', \gamma'\rangle, \alpha'\rangle$ to $\langle\langle C, \gamma\rangle, \alpha\rangle$.

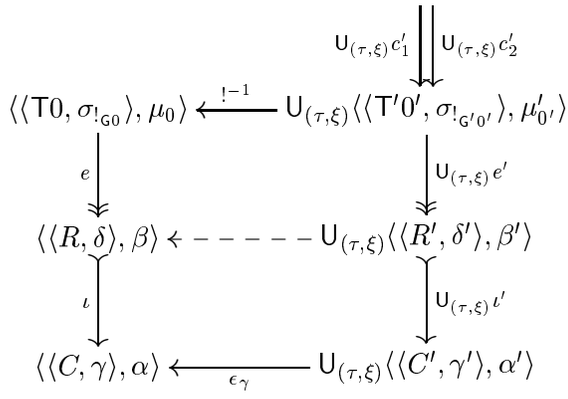
Moreover, ϵ_γ defines a couniversal arrow from $\mathsf{U}_{(\tau,\xi)}$ to $\langle\langle C, \gamma\rangle, \alpha\rangle$. For, given an arbitrary $\mathsf{T}'_{\sigma'}$-algebra $\langle\langle D, \delta\rangle, \beta\rangle$ together with a T_σ-algebra homomorphism $f : \mathsf{U}_{(\tau,\xi)}\langle\langle D, \delta\rangle, \beta\rangle \to \langle\langle C, \gamma\rangle, \alpha\rangle$, the unique (co)extension of the (C, G)-coalgebra homomorphism $f : \langle D, \delta\rangle \to \langle C, \gamma\rangle$ to a $(\mathsf{C}', \mathsf{G}')$-coalgebra homomorphism $f^\flat : \langle D, \delta\rangle \to \langle C', \gamma'\rangle$ also defines a $\mathsf{T}'_{\sigma'}$-algebra homomorphism from $\langle\langle D, \delta\rangle, \beta\rangle$ to $\langle\langle C', \gamma'\rangle, \alpha'\rangle$. This follows from the couniversality of ϵ_γ, after noting that $\epsilon_\gamma \circ \mathsf{U}(f^\flat \circ \beta) = \epsilon_\gamma \circ \mathsf{U}(\alpha' \circ \mathsf{T}'f^\flat)$. The last equality is a consequence of the fact that f defines a T-algebra homomorphism from $\mathsf{U}_\nu\langle D, \beta\rangle$ to $\langle C, \alpha\rangle$, and of the fact that ν is a natural isomorphism:

$$\begin{aligned}
\epsilon_\gamma \circ \mathsf{U}f^\flat \circ \mathsf{U}\beta \circ \nu_D &= \quad (\text{definition of } f^\flat)\\
f \circ \mathsf{U}\beta \circ \nu_D &= \quad (f \text{ is a T-algebra homomorphism})\\
\alpha \circ \mathsf{T}f &= \quad (\text{definition of } f^\flat)\\
\alpha \circ \mathsf{T}\epsilon_\gamma \circ \mathsf{TU}f^\flat &= \quad (\epsilon_\gamma \text{ is a T-algebra homomorphism})\\
\epsilon_\gamma \circ \mathsf{U}\alpha' \circ \nu_{C'} \circ \mathsf{TU}f^\flat &= \quad (\text{naturality of } \nu)\\
\epsilon_\gamma \circ \mathsf{U}\alpha' \circ \mathsf{UT}'f^\flat \circ \nu_D.
\end{aligned}$$



□

Remark 3.3.25. It is precisely the triviality of the algebraic component of lifted signature morphisms that guarantees the existence of $\mathsf{T}'_{\sigma'}$-algebra structures on the cofree coextensions of the carriers of T_σ-algebras along U_τ. Moreover, this triviality also results in the couniversal arrows $\epsilon_\gamma : \mathsf{U}_{(\tau,\xi)}\langle\langle C', \gamma'\rangle, \alpha'\rangle \to \langle\langle C, \gamma\rangle, \alpha\rangle$ *preserving reachability*. That is, if $!_{\langle\langle C,\gamma\rangle,\alpha\rangle} = \iota \circ e$ and $!_{\langle\langle C',\gamma'\rangle,\alpha'\rangle} = \iota' \circ e'$ denote the factorisations of the unique T_σ- and $\mathsf{T}'_{\sigma'}$-homomorphisms from the initial T_σ- and $\mathsf{T}'_{\sigma'}$-algebras to $\langle\langle C, \gamma\rangle, \alpha\rangle$ and $\langle\langle C', \gamma'\rangle, \alpha'\rangle$, as given by Proposition 3.3.10, then $\epsilon_\gamma \circ \mathsf{U}_{(\tau,\xi)}\iota'$ factors through ι. This follows from the commutativity of the outer square of the following diagram[29]:



(where $! : \langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle \to \mathsf{U}_{(\tau,\xi)}\langle\langle \mathsf{T}'0', \sigma_{!_{\mathsf{G}'0'}}\rangle, \mu'_{0'}\rangle$ defines the unique T_σ-algebra homomorphism resulting from the initiality of $\langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle$[30], and where c'_1, c'_2 define a kernel pair for $!_{\langle\langle C',\gamma'\rangle,\alpha'\rangle}$ in $\mathsf{Alg}(\mathsf{T}'_{\sigma'})$), together with $\mathsf{U}_{(\tau,\xi)}c'_1, \mathsf{U}_{(\tau,\xi)}c'_2$ defining a kernel pair for $\mathsf{U}_{(\tau,\xi)}!_{\langle\langle C',\gamma'\rangle,\alpha'\rangle}$ in $\mathsf{Alg}(\mathsf{T}_\sigma)$[31], $\mathsf{U}_{(\tau,\xi)}e'$ defining a quotient for $\mathsf{U}_{(\tau,\xi)}!_{\langle\langle C',\gamma'\rangle,\alpha'\rangle}$ in $\mathsf{Alg}(\mathsf{T}_\sigma)$[32], and ι defining a monomorphism in $\mathsf{Alg}(\mathsf{T}_\sigma)$[33]. This property of the couniversal arrows ϵ_γ results in cofree algebras providing suitable denotations for lifted signature morphisms.

[29] This is a consequence of the initiality of $\langle\langle \mathsf{T}0, \sigma_{!_{\mathsf{G}0}}\rangle, \mu_0\rangle$.

[30] The fact that ! is an isomorphism follows from $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}! = \nu_{0'}$, together with ν being a natural isomorphism.

[31] This follows from the preservation of kernel pairs by $\mathsf{U}_{\mathsf{C}'}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C}',\mathsf{G}')}$ and by U, together with the creation of kernel pairs by $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$.

[32] This follows by a similar argument.

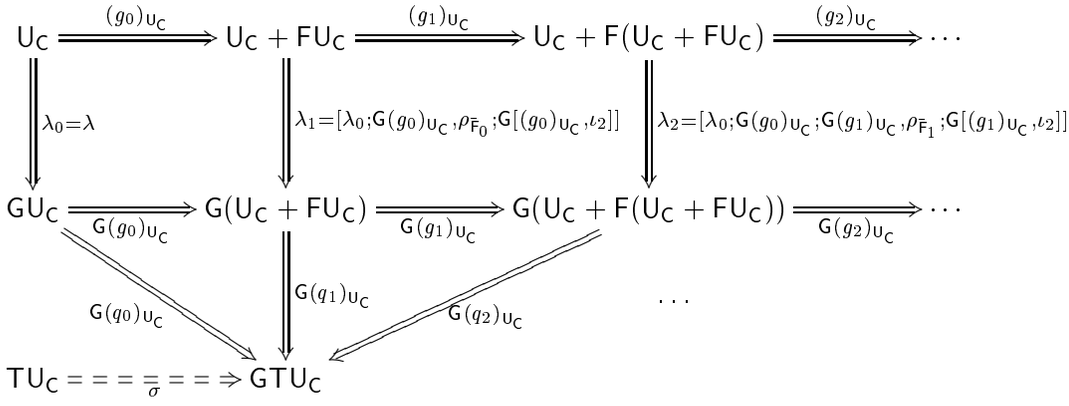
[33] This follows from the creation of pullbacks (and hence the reflection of monomorphisms) by $\mathsf{U}_\mathsf{C}\mathsf{U}_{\mathsf{Coalg}(\mathsf{C},\mathsf{G})}$.

3.3.6 A Particular Case

The natural transformation $\sigma : \mathsf{TU_C} \Rightarrow \mathsf{GTU_C}$ required by the definition of lifted signatures can be given in terms of a natural transformation $\rho : \mathsf{FU_C} \Rightarrow \mathsf{G}(\mathsf{U_C} + \mathsf{FU_C})$, defining the one-step observations of the results yielded by one-step computations as either zero- or one-step computations on observations of their arguments. To see this, note that $\mathsf{TU_C}$ is a colimit object for the following ω-chain:

$$\mathsf{U_C} \xRightarrow{(g_0)_{\mathsf{U_C}}} \mathsf{U_C} + \mathsf{FU_C} \xRightarrow{(g_1)_{\mathsf{U_C}}} \mathsf{U_C} + \mathsf{F}(\mathsf{U_C} + \mathsf{FU_C}) \xRightarrow{(g_2)_{\mathsf{U_C}}} \cdots$$

with $g_0, g_1, \ldots$ being as in Proposition 2.3.50, and with $(q_0)_{\mathsf{U_C}} : \mathsf{U_C} \Rightarrow \mathsf{TU_C}$, $(q_1)_{\mathsf{U_C}} : \mathsf{U_C} + \mathsf{FU_C} \Rightarrow \mathsf{TU_C}$, ... as colimiting arrows. Then, defining $\sigma : \mathsf{TU_C} \Rightarrow \mathsf{GTU_C}$ amounts to making $\mathsf{GTU_C}$ into a cocone on this ω-chain. The following diagram defines such a cocone:



with the natural transformation $\lambda : \mathsf{U_C} \Rightarrow \mathsf{GU_C}$ being given by $\lambda_{\langle C,\gamma\rangle} = \gamma$ for $\langle C, \gamma\rangle \in |\mathsf{Coalg(C, G)}|$, and with the natural transformations $\lambda_0, \lambda_1, \ldots$ being defined inductively by: $\lambda_0 = \lambda$ and $\lambda_{i+1} = [\lambda_0; \mathsf{G}(g_0)_{\mathsf{U_C}}; \ldots; \mathsf{G}(g_i)_{\mathsf{U_C}}, \rho_{\bar{\mathsf{F}}_i}; \mathsf{G}[(g_i)_{\mathsf{U_C}}, \iota_2]]$ for $i = 0, 1, \ldots$, where $\bar{\mathsf{F}}_i : \mathsf{Coalg(C, G)} \to \mathsf{Coalg(C, G)}$ takes $\langle C, \gamma\rangle$ to $\langle \mathsf{F}_i C, (\lambda_i)_\gamma\rangle$, for $i = 0, 1, \ldots$, and where $\mathsf{F}_i : \mathsf{C} \to \mathsf{C}$ with $i = 0, 1, \ldots$ are as in Proposition 2.3.50. (Commutativity of the upper squares follows by induction, using the naturality of ρ.) The colimiting property of $\mathsf{TU_C}$ then yields a natural transformation $\sigma : \mathsf{TU_C} \Rightarrow \mathsf{GTU_C}$ satisfying: $\mathsf{G}(q_i)_{\mathsf{U_C}} \circ \lambda_i = \sigma \circ (q_i)_{\mathsf{U_C}}$ for $i = 0, 1, \ldots$. (That is, the components of $(q_i)_{\mathsf{U_C}}$ define (C, G)-coalgebra homomorphisms from the coalgebras defined by the components of λ_i to the coalgebras defined by the components of σ, for $i = 0, 1, \ldots$.) Moreover, σ satisfies the conditions in the definition of lifted signatures. The fact that $\sigma \circ (q_0)_{\mathsf{U_C}} = \mathsf{G}(q_0)_{\mathsf{U_C}} \circ \lambda$ follows immediately from the colimiting property of $\mathsf{TU_C}$. Also, the fact that $\sigma \circ \mu_{\mathsf{U_C}} = \mathsf{G}\mu_{\mathsf{U_C}} \circ \sigma_{\mathsf{T}_\sigma}$ follows from $\sigma \circ \mu_{\mathsf{U_C}} \circ (q_i)_{\mathsf{TU_C}} = \mathsf{G}\mu_{\mathsf{U_C}} \circ \sigma_{\mathsf{T}_\sigma} \circ (q_i)_{\mathsf{TU_C}}$ for $i = 0, 1, \ldots$, using the colimiting property of $\mathsf{T}^2\mathsf{U_C}$:

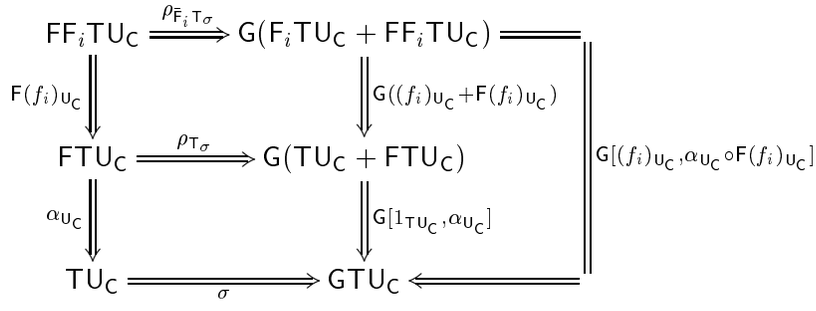
$$\begin{aligned}
\sigma \circ \mu_{\mathsf{U_C}} \circ (q_i)_{\mathsf{TU_C}} &= \quad \text{(definition of } \mu\text{)} \\
\sigma \circ (f_i)_{\mathsf{U_C}} &= \quad \text{(*)} \\
\mathsf{G}(f_i)_{\mathsf{U_C}} \circ (\lambda_i)_{\mathsf{T}_\sigma} &= \quad \text{(definition of } \mu\text{)} \\
\mathsf{G}\mu_{\mathsf{U_C}} \circ \mathsf{G}(q_i)_{\mathsf{TU_C}} \circ (\lambda_i)_{\mathsf{T}_\sigma} &= \quad (\mathsf{TU_C} = \mathsf{U_C}\mathsf{T}_\sigma\text{, definition of } \sigma) \\
\mathsf{G}\mu_{\mathsf{U_C}} \circ \sigma_{\mathsf{T}_\sigma} \circ (q_i)_{\mathsf{TU_C}}
\end{aligned}$$

with $f_i : \mathsf{F}_i\mathsf{T} \Rightarrow \mathsf{T}$ for $i = 0, 1, \ldots$ being as in Proposition 2.3.50.

The second in the above sequence of equalities, that is, $\sigma \circ (f_i)_{\mathsf{U_C}} = \mathsf{G}(f_i)_{\mathsf{U_C}} \circ (\lambda_i)_{\mathsf{T}_\sigma}$ follows by induction on i. For $i = 0$ the following holds: $\sigma \circ (f_0)_{\mathsf{U_C}} = \sigma \circ 1_{\mathsf{TU_C}} = 1_{\mathsf{GTU_C}} \circ \sigma = 1_{\mathsf{GTU_C}} \circ \lambda_{\mathsf{T}_\sigma} = \mathsf{G}(f_0)_{\mathsf{U_C}} \circ (\lambda_0)_{\mathsf{T}_\sigma}$[34]. Now assume $\sigma \circ (f_i)_{\mathsf{U_C}} = \mathsf{G}(f_i)_{\mathsf{U_C}} \circ (\lambda_i)_{\mathsf{T}_\sigma}$, that is, the components of $(f_i)_{\mathsf{U_C}}$ define (C, G)-coalgebra homomorphisms from the coalgebras defined by the components of $(\lambda_i)_{\mathsf{T}_\sigma}$ to the coalgebras defined by the components of σ. By universality of coproducts, showing that $\sigma \circ (f_{i+1})_{\mathsf{U_C}} = \mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ (\lambda_{i+1})_{\mathsf{T}_\sigma}$ amounts to showing that $\sigma \circ (f_{i+1})_{\mathsf{U_C}} \circ \iota_j = \mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ (\lambda_{i+1})_{\mathsf{T}_\sigma} \circ \iota_j$ for $j = 1, 2$. For $j = 1$, the definition of f_{i+1} gives $\sigma \circ (f_{i+1})_{\mathsf{U_C}} \circ \iota_1 = \sigma$. Also, the following holds:

$$\begin{aligned}
\mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ (\lambda_{i+1})_{\mathsf{T}_\sigma} \circ \iota_1 = &\quad \text{(definition of } \lambda_{i+1}\text{)} \\
\mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ \mathsf{G}(g_i)_{\mathsf{U_C T}_\sigma} \circ \ldots \circ \mathsf{G}(g_0)_{\mathsf{U_C T}_\sigma} \circ \lambda_{\mathsf{T}_\sigma} = &\quad (\lambda_{\mathsf{T}_\sigma} = \sigma) \\
\mathsf{G}((f_{i+1})_{\mathsf{U_C}} \circ (g_i)_{\mathsf{TU_C}} \circ \ldots \circ (g_0)_{\mathsf{TU_C}}) \circ \sigma = &\quad \text{(property of } f_0, f_1, \ldots\text{)} \\
\mathsf{G}(f_0)_{\mathsf{U_C}} \circ \sigma = &\quad (\mathsf{G}(f_0)_{\mathsf{U_C}} = 1_{\mathsf{TU_C}}) \\
\sigma &
\end{aligned}$$

Hence, $\sigma \circ (f_{i+1})_{\mathsf{U_C}} \circ \iota_1 = \mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ (\lambda_{i+1})_{\mathsf{T}_\sigma} \circ \iota_1$. It remains to show that $\sigma \circ (f_{i+1})_{\mathsf{U_C}} \circ \iota_2 = \mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ (\lambda_{i+1})_{\mathsf{T}_\sigma} \circ \iota_2$. That is, $\sigma \circ \alpha_{\mathsf{U_C}} \circ \mathsf{F}(f_i)_{\mathsf{U_C}} = \mathsf{G}(f_{i+1})_{\mathsf{U_C}} \circ \mathsf{G}[(g_i)_{\mathsf{U_C T}_\sigma}, \iota_2] \circ \rho_{\bar{\mathsf{F}}_i \mathsf{T}_\sigma}$, or equivalently, $\sigma \circ \alpha_{\mathsf{U_C}} \circ \mathsf{F}(f_i)_{\mathsf{U_C}} = \mathsf{G}[(f_i)_{\mathsf{U_C}}, \alpha_{\mathsf{U_C}} \circ \mathsf{F}(f_i)_{\mathsf{U_C}}] \circ \rho_{\bar{\mathsf{F}}_i \mathsf{T}_\sigma}$. (The previous equivalence is a consequence of $f_0, f_1, \ldots$ satisfying $f_{i+1} \circ (g_i)_{\mathsf{T}} = f_i$ and $f_{i+1} = [1_{\mathsf{T}}, \alpha \circ \mathsf{F} f_i]$ for $i = 0, 1, \ldots$.) But this follows from the commutativity of the following diagram:



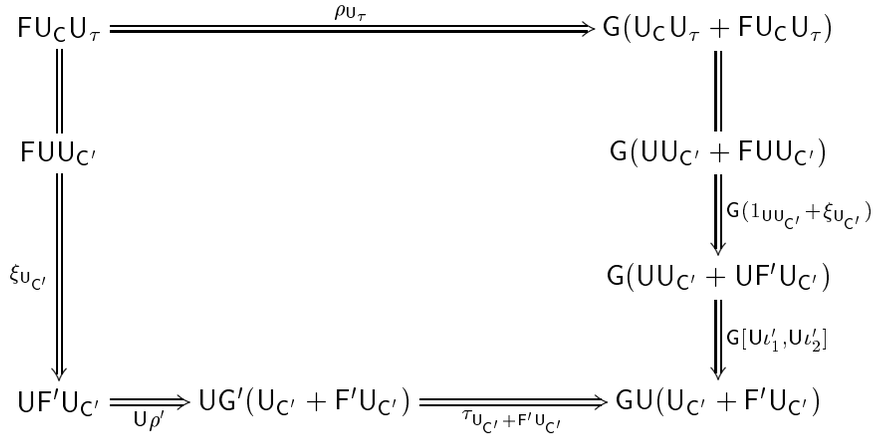
with the commutativity of the top-left square following by the naturality of ρ (by the induction hypothesis, the components of $(f_i)_{\mathsf{U_C}}$ define (C, G)-coalgebra homomorphisms from the coalgebras defined by the components of $(\lambda_i)_{\mathsf{T}_\sigma}$ to the coalgebras defined by the components of σ), the commutativity of the right square following by standard properties of coproducts, and the commutativity of the bottom-left square following by the colimiting property of $\mathsf{FTU_C}$ (the fact that F preserves colimits of ω-chains is used here) from:

$$\begin{aligned}
\sigma \circ \alpha_{\mathsf{U_C}} \circ \mathsf{F}(q_i)_{\mathsf{U_C}} = &\quad \text{(definition of } \alpha\text{)} \\
\sigma \circ (q_{i+1})_{\mathsf{U_C}} \circ \iota_2 = &\quad \text{(definition of } \sigma\text{)} \\
\mathsf{G}(q_{i+1})_{\mathsf{U_C}} \circ \lambda_{i+1} \circ \iota_2 = &\quad \text{(definition of } \lambda_{i+1}\text{)} \\
\mathsf{G}(q_{i+1})_{\mathsf{U_C}} \circ \mathsf{G}[(g_i)_{\mathsf{U_C}}, \iota_2] \circ \rho_{\bar{\mathsf{F}}_i} = &\quad \text{(definition of } \mathsf{TU_C}\text{)} \\
\mathsf{G}[(q_i)_{\mathsf{U_C}}, (q_{i+1})_{\mathsf{U_C}} \circ \iota_2] \circ \rho_{\bar{\mathsf{F}}_i} = &\quad \text{(definition of } \alpha\text{)} \\
\mathsf{G}[1_{\mathsf{TU_C}}, \alpha_{\mathsf{U_C}}] \circ \mathsf{G}((q_i)_{\mathsf{U_C}} + \mathsf{F}(q_i)_{\mathsf{U_C}}) \circ \rho_{\bar{\mathsf{F}}_i} = &\quad \text{(naturality of } \rho\text{)} \\
\mathsf{G}[1_{\mathsf{TU_C}}, \alpha_{\mathsf{U_C}}] \circ \rho_{\mathsf{T}_\sigma} \circ \mathsf{F}(q_i)_{\mathsf{U_C}} &
\end{aligned}$$

[34] Remark 3.3.2 is used here.

for $i = 0, 1, \ldots$. (The fact that the components of $(q_i)_{U_C}$ define (C, G)-coalgebra homomorphisms from the coalgebras defined by the components of λ_i to the coalgebras defined by the components of σ is used for the last equality.)

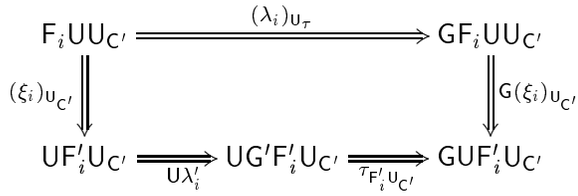
Also, if (C, G, F, σ) and (C', G', F', σ') are the lifted signatures induced by the natural transformations $\rho : FU_C \Rightarrow G(U_C + FU_C)$ and respectively $\rho' : F'U_{C'} \Rightarrow G'(U_{C'} + F'U_{C'})$, and if $(U, \tau) : (C, G) \to (C', G')$ is a cosignature morphism and $(U, \xi) : (C, F) \to (C', F')$ is a signature morphism with ξ a natural isomorphism, such that the following diagram commutes:



then (U, τ, ξ) defines a lifted signature morphism from (C, G, F, σ) to (C', G', F', σ'). The fact that $\tau_{T'U_{C'}} \circ U\sigma' \circ \nu_{U_{C'}} = G\nu_{U_{C'}} \circ \sigma_{U_\tau}$ follows from:

$$
\begin{aligned}
\tau_{T'U_{C'}} \circ U\sigma' \circ \nu_{U_{C'}} \circ (q_i)_{U_CU_\tau} = &\quad \text{(property of } \nu\text{, see proof of Proposition 2.3.52)} \\
\tau_{T'U_{C'}} \circ U\sigma' \circ U(q'_i)_{U_{C'}} \circ (\xi_i)_{U_{C'}} = &\quad \text{(definition of } \sigma'\text{)} \\
\tau_{T'U_{C'}} \circ UG'(q'_i)_{U_{C'}} \circ U\lambda'_i \circ (\xi_i)_{U_{C'}} = &\quad \text{(naturality of } \tau\text{)} \\
GU(q'_i)_{U_{C'}} \circ \tau_{F'_iU_{C'}} \circ U\lambda'_i \circ (\xi_i)_{U_{C'}} = &\quad \text{(*)} \\
GU(q'_i)_{U_{C'}} \circ G(\xi_i)_{U_{C'}} \circ (\lambda_i)_{U_\tau} = &\quad \text{(property of } \nu\text{, see proof of Proposition 2.3.52)} \\
G\nu_{U_{C'}} \circ G(q_i)_{UU_{C'}} \circ (\lambda_i)_{U_\tau} = &\quad \text{(definition of } \sigma\text{, } UU_{C'} = U_CU_\tau\text{)} \\
G\nu_{U_{C'}} \circ \sigma_{U_\tau} \circ (q_i)_{U_CU_\tau} &
\end{aligned}
$$

(with F'_i, q'_i and λ'_i being defined similarly to F_i, q_i and λ_i respectively) for $i = 0, 1, \ldots$, using the colimiting property of TU_CU_τ. The fact that $\tau_{F'_iU_{C'}} \circ U\lambda'_i \circ (\xi_i)_{U_{C'}} = G(\xi_i)_{U_{C'}} \circ (\lambda_i)_{U_\tau}$ follows by induction on i.



For $i = 0$ the following holds: $\tau_{F'_0U_{C'}} \circ U\lambda'_0 \circ (\xi_0)_{U_{C'}} = \tau_{U_{C'}} \circ U\lambda' \circ 1_{UU_{C'}} = \tau_{U_{C'}} \circ U\lambda' = \lambda_{U_\tau} = G1_{UU_{C'}} \circ \lambda_{U_\tau} = G(\xi_0)_{U_{C'}} \circ \lambda_{U_\tau}$. (The definitions of λ and λ' are used here.) Now assume $\tau_{F'_iU_{C'}} \circ U\lambda'_i \circ (\xi_i)_{U_{C'}} = G(\xi_i)_{U_{C'}} \circ (\lambda_i)_{U_\tau}$, that is, the components of $(\xi_i)_{U_{C'}}$ define (C, G)-coalgebra homomorphisms from the coalgebras defined by the components of $(\lambda_i)_{U_\tau}$ to the coalgebras defined by the components

of $\mathsf{U}_\tau\lambda'_i$. Again, showing $\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ (\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} = \mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_{i+1})_{\mathsf{U}_\tau}$ amounts to showing $\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ (\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \iota_j = \mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_{i+1})_{\mathsf{U}_\tau} \circ \iota_j$ for $j = 1, 2$. But this follows from:

$$\begin{aligned}
\mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_{i+1})_{\mathsf{U}_\tau} \circ \iota_1 = &\quad \text{(definition of } \lambda_{i+1}\text{)}\\
\mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{G}(g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}} \circ \ldots \circ \mathsf{G}(g_0)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_0)_{\mathsf{U}_\tau} = &\quad \text{(property of } \xi_0, \xi_1, \ldots\text{)}\\
\mathsf{GU}(g'_i)_{\mathsf{U}_{\mathsf{C}'}} \circ \ldots \circ \mathsf{GU}(g'_0)_{\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_0)_{\mathsf{U}_\tau} = &\quad (\lambda_{\mathsf{U}_\tau} = \tau_{\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda')\\
\mathsf{GU}(g'_i)_{\mathsf{U}_{\mathsf{C}'}} \circ \ldots \circ \mathsf{GU}(g'_0)_{\mathsf{U}_{\mathsf{C}'}} \circ \tau_{\mathsf{F}'_0\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_0 = &\quad \text{(naturality of } \tau\text{)}\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{UG}'(g'_i)_{\mathsf{U}_{\mathsf{C}'}} \circ \ldots \circ \mathsf{UG}'(g'_0)_{\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_0 = &\quad \text{(definition of } \lambda'_{i+1}\text{)}\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ \mathsf{U}\iota'_1 = &\quad \text{(definition of } \xi_{i+1}\text{)}\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ (\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \iota_1 &
\end{aligned}$$

together with:

$$\begin{aligned}
\mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ (\lambda_{i+1})_{\mathsf{U}_\tau} \circ \iota_2 = &\\
\text{(definition of } \lambda_{i+1}\text{)}&\\
\mathsf{G}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{G}[(g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \rho_{\bar{\mathsf{F}}_i\mathsf{U}_\tau} = &\\
(*)&\\
\mathsf{G}[\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \mathsf{G}[\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ \mathsf{G}(1_{\mathsf{U}\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} + \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}}) \circ \mathsf{G}((\xi_i)_{\mathsf{U}_{\mathsf{C}'}} + \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}}) \circ \rho_{\bar{\mathsf{F}}_i\mathsf{U}_\tau} = &\\
\text{(naturality of } \rho\text{, induction hypothesis)}&\\
\mathsf{G}[\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \mathsf{G}[\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ \mathsf{G}(1_{\mathsf{U}\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} + \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}}) \circ \rho_{\mathsf{U}_\tau\bar{\mathsf{F}}'_i} \circ \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}} = &\\
(\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'} = \mathsf{U}_{\mathsf{C}'}\bar{\mathsf{F}}'_i\text{, property of } \rho, \rho'\text{)}&\\
\mathsf{GU}[(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \tau_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}+\mathsf{F}'\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\rho'_{\bar{\mathsf{F}}'_i} \circ \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}} = &\\
\text{(naturality of } \tau\text{)}&\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{UG}'[(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \mathsf{U}\rho'_{\bar{\mathsf{F}}'_i} \circ \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}} = &\\
\text{(definition of } \lambda'_{i+1}\text{)}&\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ \mathsf{U}\iota_2 \circ \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}} = &\\
\text{(definition of } \xi_{i+1}\text{)}&\\
\tau_{\mathsf{F}'_{i+1}\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{U}\lambda'_{i+1} \circ (\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \iota_2 &
\end{aligned}$$

where $\bar{\mathsf{F}}'_i : \mathsf{Coalg}(\mathsf{C}', \mathsf{F}') \to \mathsf{Coalg}(\mathsf{C}', \mathsf{F}')$ takes $\langle C', \gamma' \rangle$ to $\langle \mathsf{F}'_i C', (\lambda'_i)_{\gamma'} \rangle$ for $i = 0, 1, \ldots$, and where $(*)$ uses the fact that $(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ [(g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}}, \iota_2] = [\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ [\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ (1_{\mathsf{U}\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} + \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}}) \circ ((\xi_i)_{\mathsf{U}_{\mathsf{C}'}} + \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}})$, which follows from:

$$\begin{aligned}
(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ [(g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \iota_1 = &\\
(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ (g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}} = &\\
\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}} \circ (\xi_i)_{\mathsf{U}_{\mathsf{C}'}} = &\\
[\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ [\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ (1_{\mathsf{U}\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} + \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}}) \circ ((\xi_i)_{\mathsf{U}_{\mathsf{C}'}} + \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}}) \circ \iota_1 &
\end{aligned}$$

together with:

$$\begin{aligned}(\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ [(g_i)_{\mathsf{U}\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ \iota_2 &= \\ (\xi_{i+1})_{\mathsf{U}_{\mathsf{C}'}} \circ \iota_2 &= \\ \mathsf{U}\iota'_2 \circ \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} \circ \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}} &= \\ [\mathsf{U}(g'_i)_{\mathsf{U}_{\mathsf{C}'}}, \iota_2] \circ [\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ (1_{\mathsf{U}\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}} + \xi_{\mathsf{F}'_i\mathsf{U}_{\mathsf{C}'}}) \circ ((\xi_i)_{\mathsf{U}_{\mathsf{C}'}} + \mathsf{F}(\xi_i)_{\mathsf{U}_{\mathsf{C}'}}) \circ \iota_2\end{aligned}$$

Remark 3.3.26. Let $\sigma : \mathsf{TU_C} \Rightarrow \mathsf{GTU_C}$ denote the natural transformation induced by a natural transformation $\rho : \mathsf{FU_C} \Rightarrow \mathsf{G}(\mathsf{U_C} + \mathsf{FU_C})$. Also, let $\langle C, \gamma \rangle$ denote a (C, G)-coalgebra, let $\alpha : \mathsf{T}C \to C$ define a T-algebra structure on C, and let $\alpha' : \mathsf{F}C \to C$ denote the F-algebra structure induced by α, as given by the proof of Proposition 2.3.50. Then, the diagram:

$$\begin{array}{ccc} \mathsf{F}C & \xrightarrow{\rho_\gamma} & \mathsf{G}(C + \mathsf{F}C) \\ {\scriptstyle \alpha'}\downarrow & & \downarrow{\scriptstyle \mathsf{G}[1_C, \alpha']} \\ C & \xrightarrow[\gamma]{} & \mathsf{G}C \end{array}$$

commutes if and only if the diagram:

$$\begin{array}{ccc} \mathsf{T}C & \xrightarrow{\sigma_\gamma} & \mathsf{GT}C \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \mathsf{G}\alpha} \\ C & \xrightarrow[\gamma]{} & \mathsf{G}C \end{array}$$

commutes. The if direction follows from:

$$\begin{aligned}\gamma \circ \alpha' = &\quad \text{(definition of } \alpha') \\ \gamma \circ \alpha \circ (q_1)_C \circ \iota_2 = &\quad (\gamma \circ \alpha = \mathsf{G}\alpha \circ \sigma_\gamma) \\ \mathsf{G}\alpha \circ \sigma_\gamma \circ (q_1)_C \circ \iota_2 = &\quad \text{(definition of } \sigma) \\ \mathsf{G}\alpha \circ \mathsf{G}(q_1)_C \circ (\lambda_1)_\gamma \circ \iota_2 = &\quad \text{(definition of } \lambda_1) \\ \mathsf{G}\alpha \circ \mathsf{G}(q_1)_C \circ \rho_\gamma = &\quad \text{(property of coproducts)} \\ [\mathsf{G}(\alpha \circ (q_1)_C \circ \iota_1, \mathsf{G}(\alpha \circ (q_1)_C \circ \iota_2)] \circ \rho_\gamma = &\quad (\alpha \circ (q_1)_C \circ \iota_1 = \alpha \circ (q_0)_C = 1_C, \text{ definition of } \alpha') \\ \mathsf{G}[1_C, \alpha'] &\end{aligned}$$

The only if direction uses the definition of α in terms of α' as given by the proof of Proposition 2.3.50. Specifically, $\alpha : \mathsf{T}C \to C$ is the unique C-arrow satisfying $\alpha \circ (q_i)_C = \alpha_i$ for $i = 0, 1, \ldots$, where the C-arrows $\alpha_i : \mathsf{F}_i C \to C$ with $i = 0, 1, \ldots$ are given by: $\alpha_0 = 1_C$ and $\alpha_{i+1} = [1_C, \alpha' \circ \mathsf{F}\alpha_i]$ for $i = 0, 1, \ldots$. Then, the conclusion follows from:

$$\begin{aligned}\gamma \circ \alpha \circ (q_i)_C = &\quad \text{(definition of } \alpha) \\ \gamma \circ \alpha_i = &\quad (*) \\ \mathsf{G}\alpha_i \circ (\lambda_i)_\gamma = &\quad \text{(definition of } \alpha) \\ \mathsf{G}\alpha \circ \mathsf{G}(q_i)_C \circ (\lambda_i)_\gamma = &\quad \text{(definition of } \sigma) \\ \mathsf{G}\alpha \circ \sigma_\gamma \circ (q_i)_C &\end{aligned}$$

for $i = 0, 1, \ldots$, using the colimiting property of T. (The fact that $\gamma \circ \alpha_i = \mathsf{G}\alpha_i \circ (\lambda_i)_\gamma$ for $i = 0, 1, \ldots$ follows by induction, using the naturality of ρ.)

As a result, T_σ-algebras $\langle\langle C, \gamma\rangle, \alpha\rangle$ are in one-to-one correspondence with (F, G)-bialgebras $\langle C, \alpha', \gamma\rangle$ additionally satisfying: $\gamma \circ \alpha' = \mathsf{G}[1_C, \alpha'] \circ \rho_\gamma$. (Remark 3.3.3 is also used here.)

In [TP97], liftings of monads T induced by endofunctors F to categories of G-coalgebras are shown to arise from natural transformations of type $\mathsf{F}(\mathsf{Id}_\mathsf{C} \times \mathsf{G}) \Rightarrow \mathsf{GT}$. Moreover, such natural transformations are shown to be in one-to-one correspondence with natural transformations of type $\mathsf{F}(\mathsf{T} \times \mathsf{GT}) \Rightarrow \mathsf{GT}$. The latter are essentially the same as natural transformations $\mathsf{F}(\mathsf{U}'_\mathsf{C} \times \mathsf{GU}'_\mathsf{C}) \Rightarrow \mathsf{GU}'_\mathsf{C}$ (with $\mathsf{U}'_\mathsf{C} : \mathsf{Alg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$ denoting the functor taking (C, F)-algebras to their carrier), and therefore also determine liftings of the comonad D induced by G to F-algebras. (This follows by an argument dual to the one given above.) The apparent mismatch between the approach in [TP97] and the one presented here (caused by the fact that in [TP97] the emphasis is on the labelled transitions performable by programs over a given language, and consequently the system dynamics is captured by the coalgebraic component, whereas here the emphasis is on the computations performable on system states, and as a result the system dynamics is captured by the algebraic component) is, however, of no importance, since liftings of T to G-coalgebras are shown in [TP97] (see also the concluding remarks in Section 3.3.7) to be in one-to-one correspondence with liftings of D to F-algebras, and therefore either of the two forms of natural transformations can be used to specify either of the two kinds of liftings.

The natural transformations σ arising from natural transformations ρ of the form considered in this section will prove sufficiently general for our purpose. This will become more apparent in Chapter 5, where the specification framework introduced here will be instantiated in order to obtain a formalism for the specification of objects.

3.3.7 A Dual Approach

An approach which involves lifting the algebraic structure of syntactic domains (of programs) to observable behaviours over these syntactic domains can be obtained essentially by dualising the definitions and results of Sections 3.3.1–3.3.6. The resulting approach is briefly outlined in the following. Such an approach will prove suitable for extending data type specifications with definitions of data observers with structured result type. (This will be illustrated in Section 5.2.)

Lifted cosignatures are used in the resulting approach to specify liftings of comonads induced by abstract cosignatures to categories of algebras of abstract signatures, while *lifted cosignature morphisms* are used to specify structure-preserving translations between such liftings. Specifically, lifted cosignatures are defined as tuples $(\mathsf{C}, \mathsf{F}, \mathsf{G}, \sigma)$, with (C, F) an abstract signature, (C, G) an abstract cosignature and $\sigma : \mathsf{FDU}'_\mathsf{C} \Rightarrow \mathsf{DU}'_\mathsf{C}$ a natural transformation (where $\mathsf{U}'_\mathsf{C} : \mathsf{Alg}(\mathsf{C}, \mathsf{F}) \to \mathsf{C}$ denotes the functor taking (C, F)-algebras to their carrier, and where D denotes the comonad induced by the abstract cosignature (C, G)), subject to constraints similar to the ones in Definition 3.3.1. Also, lifted cosignature morphisms are defined as tuples (U, ξ, τ), with (U, ξ) a signature morphism and (U, τ) a cosignature morphism such that τ is a natural isomorphism, again, subject to constraints similar to those in Definition 3.3.1.

The models of a lifted cosignature (C, F, G, σ) are the coalgebras of the comonad D_σ on $Alg(C, F)$ induced by the natural transformation σ. The following diagram summarises the relationship between D_σ-coalgebras on the one hand and (C, F)-algebras and D-coalgebras on the other.

$$\begin{array}{ccc} Coalg(D_\sigma) & \xrightarrow{U_{Coalg(D)}} & Coalg(D) \\ \Big\downarrow{\scriptstyle U_{Alg(C,F)}} & & \Big\downarrow{\scriptstyle U_C} \\ Alg(C, F) & \xrightarrow[U'_C]{} & C \end{array}$$

The relevant properties of the categories and functors appearing in this diagram can also be summarised as follows:

1. $Coalg(D_\sigma)$ has any colimits that exist in $Alg(C, F)$. (This follows by the dual of Corollary 2.3.49.) In particular, $Coalg(D_\sigma)$ has an initial object, whose underlying (C, F)-algebra is an initial (C, F)-algebra.
2. $Coalg(D_\sigma)$ has a final object, whose underlying D-coalgebra is a final D-coalgebra. (This follows similarly to Proposition 3.3.8.)
3. $Coalg(D_\sigma)$ has pullbacks, while $U_{Coalg(D)}$ and $U_{Alg(C,F)}$ create as well as preserve them.
4. $Coalg(D_\sigma)$ has quotients, while $U_{Coalg(D)}$ and $U_{Alg(C,F)}$ create as well as preserve them. (This is a consequence of the existence of quotients in $Alg(C, F)$ and $Coalg(D)$ and of their preservation by U_C and creation by U'_C, and follows similarly to Proposition 3.3.9.)

As in the operational approach, the quotient of the unique homomorphism from the initial to the final D_σ-coalgebra is used to provide a denotation for the lifted cosignature (C, F, G, σ). Since quotients in $Coalg(D_\sigma)$ are preserved by $U_{Alg(C,F)}$, it follows that the codomain of the quotient of a D_σ-coalgebra homomorphism defines a homomorphic image of the domain of the given homomorphism, as well as a subcoalgebra of the codomain of that homomorphism. As a result, the codomain of the quotient of the unique homomorphism from the initial to the final D_σ-coalgebra is observable, while its underlying algebra is reachable. (The fact that U'_C preserves quotients is also used here.)

The notions of satisfaction of equations up to bisimulation and of coequations up to reachability by coalgebras of lifted cosignatures are defined as in Section 3.3.1, and enjoy properties similar to those of their operational counterparts. In particular, the codomains of the quotients of the unique homomorphisms from the initial to the final D_σ-coalgebras satisfy precisely those equations which are satisfied up to bisimulation by the initial D_σ-coalgebras, and precisely those coequations which are satisfied up to reachability by the final D_σ-coalgebras.

Finally, the natural transformations $\sigma : FDU'_C \Rightarrow DU'_C$ required by the definition of lifted cosignatures can be given in terms of natural transformations $\rho : F(U'_C \times GU'_C) \Rightarrow GU'_C$, defining the one-step observations of the results yielded by one-step computations as computations on zero- and one-step observations of their arguments.

A similar approach to specification is taken in [CHM99] (see also [CGRH98]), where suitably-restricted *heterogeneous transition systems* (that is, transition systems whose states carry algebraic structure) are regarded as coalgebras of endofunctors on categories of algebras. Specifically, the

setting in [CHM99] is obtained by taking C to be Set, F : Set $\to$ Set to be the endofunctor induced by a (one-sorted) signature Σ, G : Set $\to$ Set to be given by $\mathcal{P}_c(L \times \_)$[35] for some set L (of *labels*), and σ to be the natural transformation induced by a set of *structural operational semantics rules* over Σ and L. The behaviour induced by these rules is required to be *compositional*, in that transitions on complex states (denoted by Σ-terms) must be derivable from transitions on component states (denoted by the variables appearing in these Σ-terms) using the given rules. Such rules are then shown to induce liftings of the endofunctor $\mathcal{P}_c(L \times \_)$ to categories of Σ-algebras, and this observation is used to obtain conditions under which $\mathcal{P}_c(L \times \_)$-bisimilarity is a congruence w.r.t. the algebraic structure.

Also related to the approach presented here (and indeed, to the one in [CHM99]) is the calculus of communicating systems (CCS) [Mil80]. There, terms over a suitable signature are used to denote processes, and an operational semantics for the resulting process algebra is defined using structural rules over this signature. The observational equivalence of processes, given by *strong bisimulation*, is then shown to be a congruence w.r.t. the algebraic structure. This allows various equivalences between processes to be derived using equational reasoning.

We conclude this section with a few remarks on the relationship between the operational approach described in Sections 3.3.1–3.3.6 and the denotational approach outlined above. In [Tur96] (see also [TP97]), monads T and comonads D are used to specify syntax and respectively behaviour, and *distributive laws* (defined as natural transformations λ : TD $\Rightarrow$ DT subject to certain compatibility conditions involving the unit and multiplication of T and respectively the counit and comultiplication of D) are used to relate the two. It is also shown in [Tur96] that distributive laws λ : TD $\Rightarrow$ DT are in one-to-one correspondence with liftings of T to D-coalgebras (or, equivalently, to G-coalgebras), as well as with liftings of D to T-algebras (or, equivalently, to F-algebras). Consequently, the operational approach to specification described in Sections 3.3.1–3.3.6 and its denotational counterpart outlined in Section 3.3.7 are equally expressive for specifying the relationship between computations and observations in structures having both a computational and an observational component. Nevertheless, both from a semantic point of view and in order to facilitate specification and correctness proofs, it is worth distinguishing between the two kinds of approaches. Again, this will become more apparent in Chapter 5, where the two approaches will be used for the specification of systems with state (with finality being relevant at the semantic level, and with coinduction prevailing as a proof technique) and respectively of data types (with initiality being of interest from a semantic point of view, and with induction being typically used in proofs).

3.3.8 Related Work

This section briefly compares the approach presented in this chapter with similar work in [Fok96] (extending previous work in [Man76], see Remark 3.2.14) on semantic generalisations of the notion of equation.

Notions of *transformer* and *law*, generalising those of term and equation, are introduced in [Fok96] in the context of *dialgebras*. If F, G : A $\to$ B denote arbitrary functors, an (F, G)-*dialgebra* is given by a pair $\langle A, \varphi \rangle$ with A an A-object (called the *carrier* of the dialgebra) and φ : FA $\to$ GA a

[35] The endofunctor $\mathcal{P}_c(L \times \_)$: Set $\to$ Set takes a set X to the set of all countable subsets of $L \times X$.

B-arrow. Also, an (F, G)-*dialgebra homomorphism* from $\langle A, \varphi \rangle$ to $\langle B, \psi \rangle$ is given by an A-arrow $f : A \to B$ additionally satisfying: $\mathsf{G}f \circ \varphi = \psi \circ \mathsf{F}f$. The category of (F, G)-dialgebras and (F, G)-dialgebra homomorphisms is denoted Dialg(F, G). Thus, dialgebras generalise both algebras (which are obtained by taking B = A and G = Id_A) and coalgebras (which are obtained by taking B = A and F = Id_A).

In this setting, a *transformer of type* (F, G) → (H, K) [Fok96], with F, G : A → B and H, K : A → C, is given by a mapping T from (F, G)-dialgebras to (H, K)-dialgebras, additionally satisfying:

$$f : \varphi \to_{\mathsf{F},\mathsf{G}} \psi \;\Rightarrow\; f : T\varphi \to_{\mathsf{H},\mathsf{K}} T\psi$$

that is:

$$\mathsf{G}f \circ \varphi = \psi \circ \mathsf{F}f \;\Rightarrow\; \mathsf{K}f \circ T\varphi = T\psi \circ \mathsf{H}f$$

for any (F, G)-dialgebra homomorphism $f : \langle A, \varphi \rangle \to \langle B, \psi \rangle$. Also, a *law of type* (F, G) → (H, K) is given by a pair of transformers of type (F, G) → (H, K). A law (T, T') is said to *hold* for an (F, G)-dialgebra $\langle A, \varphi \rangle$ if and only if $T\varphi = T'\varphi$.

[Fok96] also shows that transformers of type (F, G) → (H, K) are the same as natural transformations of type HU ⇒ KU, with U : Dialg(F, G) → A denoting the functor taking (F, G)-dialgebras to their carrier, while the satisfaction of a law (T, T') by an (F, G)-dialgebra $\langle A, \varphi \rangle$ amounts to the φ-components of the natural transformations associated to T and T' being the same. In this respect, transformers and laws are generalisations of the notions of observer and abstract coequation introduced in Section 3.1.2, as well as of the dual notions of constructor and abstract equation introduced in Section 3.2.2.

In the following, the use of laws in specifying observational and computational structures on the one hand and combined structures on the other is discussed in comparison with the approaches presented in Sections 3.1 and 3.2, and respectively with the approach presented in Section 3.3.

As far as the notion of observer over an abstract cosignature is concerned, the only reason for using natural transformations of type $\mathsf{HU}_\mathsf{C} \Rightarrow \mathsf{KU}_\mathsf{C}$, as opposed to natural transformations of type $\mathsf{U}_\mathsf{C} \Rightarrow \mathsf{KU}_\mathsf{C}$, in defining it would, in our opinion, be to allow the result type of observers to lie in a less structured category than the category used to define the cosignature. Such an approach would, for instance, allow unconditional hidden Σ-equations of form $(\forall\, Z)\; l = r$, with Σ a destructor hidden signature, to be captured by abstract coequations of form $(\mathsf{H}, \mathsf{K}, l', r')$, with $\mathsf{H} : \mathsf{Set}^S_D \to \mathsf{Set}$ being given by:

$$\mathsf{H}X = X_h$$

for $X \in |\mathsf{Set}^S_D|$ (where h denotes the type of Z), with $\mathsf{K} : \mathsf{Set}^S_D \to \mathsf{Set}$ being given by:

$$\mathsf{K}X = X_s$$

for $X \in |\mathsf{Set}^S_D|$ (where s denotes the type of l and r), and with $l', r' : \mathsf{HU} \Rightarrow \mathsf{KU}$ (where $\mathsf{U} : \mathsf{Coalg}(\mathsf{Set}^S_D, \mathsf{F}_\Sigma) \to \mathsf{Set}^S_D$ denotes the functor taking $(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)$-coalgebras to their carrier) being given by:

$$l'_{\langle C,\gamma \rangle} = l_A : A_h \to A_s$$

for $\langle C, \gamma \rangle \in |\mathsf{Coalg}(\mathsf{Set}^S_D, \mathsf{F}_\Sigma)|$ with A its associated hidden Σ-algebra, and similarly for r'. Under the approach presented in Section 3.1, such equations are captured by (more complex) abstract

coequations of form (K, l', r'), with $\mathsf{K} : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ being given by:

$$(\mathsf{K}X)_h = \begin{cases} X_s & \text{if } \ Z : h \\ 1 & \text{otherwise} \end{cases} \ , \quad h \in H$$
$$(\mathsf{K}X)_v = D_v, \quad v \in V$$

where s denotes the type of l and r, and with $l', r' : \mathsf{U} \Rightarrow \mathsf{KU}$ being given by:

$$(l'_{\langle C,\gamma\rangle})_h = \begin{cases} l_A & \text{if } \ Z : h \\ ! : C_h \to 1 & \text{otherwise} \end{cases} \ , \quad h \in H$$
$$(l'_{\langle C,\gamma\rangle})_v = 1_{D_v} \quad v \in V$$

for $\langle C, \gamma\rangle \in |\mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{F}_\Sigma)|$ with A its associated hidden Σ-algebra, and similarly for r'.

However, the use of natural transformations of type $\mathsf{HU_C} \Rightarrow \mathsf{KU_C}$ does not appear to increase the expressiveness of the notions of observer and coequation, at least not in the case when $\mathsf{H} : \mathsf{C} \to \mathsf{D}$ is a functor having a right adjoint $\mathsf{R} : \mathsf{D} \to \mathsf{C}$ (which is the case in the previous example). For, in this case, natural transformations $c : \mathsf{HU_C} \Rightarrow \mathsf{KU_C}$ are in one-to-one correspondence with natural transformations $c' : \mathsf{U_C} \Rightarrow \mathsf{RKU_C}$ (with $c'_{\langle C,\gamma\rangle} : C \to \mathsf{RK}C$ being given by $c^\flat_{\langle C,\gamma\rangle}$ for any (C, F)-coalgebra $\langle C, \gamma\rangle$). Moreover, the satisfaction of a generalised (C, F)-coequation $(\mathsf{H}, \mathsf{K}, l, r)$, with $l, r : \mathsf{HU_C} \Rightarrow \mathsf{KU_C}$, by a (C, F)-coalgebra $\langle C, \gamma\rangle$ is equivalent to the satisfaction of the (C, F)-coequation $(\mathsf{RK}, l^\flat, r^\flat)$ by $\langle C, \gamma\rangle$. This follows from:

$$\begin{aligned} \langle C, \gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{H}, \mathsf{K}, l, r) \ &\Leftrightarrow \\ l_\gamma = r_\gamma \ &\Leftrightarrow \\ l^\flat_\gamma = r^\flat_\gamma \ &\Leftrightarrow \\ \langle C, \gamma\rangle \models_{(\mathsf{C},\mathsf{F})} (\mathsf{RK}, l^\flat, r^\flat) & \end{aligned}$$

A similar observation can be made about the use of natural transformations of type $\mathsf{KU_C} \Rightarrow \mathsf{HU_C}$ (as opposed to natural transformations of type $\mathsf{KU_C} \Rightarrow \mathsf{U_C}$) in defining the notion of constructor over an abstract signature. A detailed investigation of the properties enjoyed by such generalised notions of observer and constructor constitutes the subject of future work.

The use of laws in specifying the relationship between computations and observations in structures having both a computational and an observational component constitutes an alternative approach to the specification of such structures. This approach is illustrated in [Fok96] using a number of examples of data type specifications. Rather than taking a layered view towards the integration of computational and observational features, such an approach considers both categories of features at the same time. On the one hand, this allows for more freedom in specifying the relationship between the two categories of features, as no restrictions are imposed to the sets of laws used for specification. On the other hand, such an approach does not guarantee either the existence of initial/final models, or the existence of a compatibility between the induced notions of reachability and observational indistinguishability.

4 Coalgebraic Specification

Existing formalisms for the specification of state-based, dynamical systems typically employ an algebraic syntax in formalising and reasoning about state observations [GM97, Cor98, HB99, HK99]. The use of such a syntax prevents these formalisms from accommodating observers whose result type is structured as a coproduct of basic types. Such observers are essential in system specification, as they offer a choice in what can be observed of a system in a particular state. For instance, an observer of type $\mathtt{next} : \mathtt{Stream} \to 1 + \mathtt{Stream}$ (with 1 denoting a one-element type) is needed to specify the fact that streams can terminate at any point, whereas an observer of type $? : \mathtt{Tree} \to \mathtt{Leaf} + \mathtt{Node}$ is needed to classify binary trees into leaves and nodes (as the type of information that can be observed about leaves differs from the type of information that can be observed about nodes). This chapter presents a coalgebraic specification formalism which accommodates observers of the above-mentioned form.

From a syntactic point of view, the approach is dual to the many-sorted algebraic approach to the specification of data types. Observers whose result type is structured as a coproduct of basic types are specified using the notion of *many-sorted cosignature*, while arbitrary state observations are formalised using the notion of *coterm*, which captures the successive evaluation of observers by providing alternatives for proceeding with an evaluation, depending on the type of the result yielded by the most recently evaluated observer. (*Covariables* are used in coterms as place-holders for the possible results of such evaluations.) *Coequations* are then used to constrain the specified behaviours, by requiring different observations of the same state to yield similar results. Coequations may be conditional, this allowing the formulation of a sound and complete deduction calculus for reasoning about the specified behaviours.

From a semantic point of view, however, an approach which simply dualises the many-sorted algebraic approach to specification proves not to be completely satisfactory, as the expressiveness of the resulting formalism does not equal that of many-sorted algebra (or, equivalently, of arbitrary polynomial endofunctors) from the point of view of the structures specifiable within it. This lack of expressiveness is reflected in the fact that the induced notion of observational indistinguishability is not sufficiently fine, and at the same time suggests the need for a data universe w.r.t. which observational properties are specified[1]. Indeed, allowing specifications to be relative to a fixed data universe increases the expressiveness of the formalism to that of extended polynomial endofunctors. And although coequations are still not sufficiently expressive to allow the formulation of a characterisability result similar to Birkhoff's (see Theorem 2.3.28), they succeed in capturing the kinds of constraints one expects to impose when specifying state spaces; these are constraints on the topology of state-based systems, referring to various dependencies between the system components (including the sharing[2] of data or of subcomponents between the system components, or the

[1] This suggestion is strengthened by the presence, in existing specification formalisms, of a similar data universe.

[2] Our approach to sharing is based on the assumption that two systems sharing a component are both subsystems

presence or absence of certain components in some of the system states).

The chapter is structured as follows. Section 4.1 introduces a formalism for the specification of observational structures, whose syntax and semantics dualise the syntax and semantics of many-sorted algebra. Section 4.2 extends this formalism in order to account for the availability of a fixed data universe w.r.t. which observational properties are specified, and then discusses the expressiveness of the resulting approach.

Note The work reported in this chapter was carried out independently of (and prior to) the work described in Chapter 3. Furthermore, this work has partly influenced the abstract coalgebraic approach to specification presented in Section 3.1 – several of the abstract notions introduced in Section 3.1 are generalisations of notions defined in this chapter. This particular ordering of the two chapters is intended to simplify the presentation of the results in the present chapter, by allowing a number of properties of the concrete formalism introduced here to be derived automatically from properties of the abstract framework presented in Section 3.1[3].

4.1 Many-Sorted Coalgebra

This section presents a formalism for the specification of observational structures allowing for a choice in the result type of observers, obtained essentially by dualising the many-sorted algebraic formalism for the specification of data types. A sound and complete deduction calculus for reasoning about the specified behaviours is also formulated.

4.1.1 Cosignatures, Covariables, Coterms and Substitution

Definition 4.1.1. *A* **(many-sorted) cosignature** *is a pair* (S, Δ) *with* S *a set of* **sorts** *and* Δ *an* $S \times S^+$*-sorted set of* **operation symbols** *(where* S^+ *denotes the set of finite, non-empty sequences of sorts). One writes* $\delta : s \to s_1 \dots s_n$ *for* $\delta \in \Delta_{s,s_1 \dots s_n}$*.*

Cosignatures (S, Δ) are abbreviated Δ whenever the context allows it. For a many-sorted cosignature Δ with sort set S, the set $\{\delta \mid \delta \in \Delta_{s,s_1 \dots s_n}, s_1, \dots, s_n \in S\}$ with $s \in S$ is denoted Δ_s. In the following it is only assumed that the sets Δ_s with $s \in S$ are enumerable. In practice however, these sets are usually finite.

The operation symbols of a many-sorted cosignature specify basic ways of observing the states of a given system. An operation symbol $\delta : s \to s_1 \dots s_n$ denotes an operation whose domain is the type denoted by s and whose codomain is given by the coproduct of the types denoted by $s_1, \dots, s_n$.

Example 4.1.2. Lists (finite or infinite) are specified using sorts `1` (denoting a one-element set), `Elt` (for the list elements), `List` and `NeList` (for arbitrary and respectively non-empty lists), and operation symbols `empty?` : `List` $\to$ `1` `NeList` (used to classify lists into empty and non-empty

of a larger system.

[3] Direct proofs of some of these properties can be found in [Cîr99a].

ones), `head` : `NeList` $\to$ `Elt` and `tail` : `NeList` $\to$ `List` (yielding the head and respectively the tail of a non-empty list).

Arbitrary state observations are formalised by the notion of *coterm*, which provides alternatives for proceeding with an observation, depending on the type of the result yielded by the most recently evaluated observer. *Covariables* are used in coterms as place-holders for their potential outputs, in a manner similar to the use of variables as place-holders for the inputs of algebraic terms.

Definition 4.1.3. *Let Δ denote a many-sorted cosignature with sort set S, and let $\mathcal{C}$ denote an S-sorted set (of* **covariables***). The (S-sorted) set $T_\Delta[\mathcal{C}]$ of* **Δ-coterms with covariables from** *$\mathcal{C}$ is the least S-sorted set satisfying:*

1. $Z \in T_\Delta[\mathcal{C}]_s$ *for* $Z \in \mathcal{C}_s$
2. $[t_1, \ldots, t_n]\delta \in T_\Delta[\mathcal{C}]_s$ *for* $\delta \in \Delta_{s,s_1 \ldots s_n}$ *and* $t_i \in T_\Delta[\mathcal{C}]_{s_i}$, $i = 1, \ldots, n$.

Coterms of sort $s \in S$ (elements of $T_\Delta[\mathcal{C}]_s$) specify ways of observing states of type s. Specifically, a coterm of form Z denotes a trivial observation, whose result is precisely the state being observed, whereas a coterm of form $[t_1, \ldots, t_n]\delta$ denotes an observation which amounts to first evaluating the observer denoted by δ and then, depending on whether the result of this evaluation is of the type denoted by s_1, s_2, ... or s_n, continuing with the observation denoted by one of $t_1, t_2, \ldots, t_n$. That is, evaluation begins from the right and proceeds towards the left according to the type of the result yielded by the most recently evaluated observer. It is also worth noting that there are no coterms over an empty set of covariables.

Example 4.1.4. The following are coterms of sort `List`: `[F,N]empty?` (used to observe whether a list is empty or not), `[F,[E]head]empty?` (used to observe the first element of a list, in case it exists), `[F,[[F,[E]head]empty?]tail]empty?` (used to observe the second element, in case it exists), and so on. The result types of these coterms are: `1 NeList`, `1 Elt`, and respectively `1 Elt`[4]. Now consider the second of these coterms, i.e. `[F,[E]head]empty?`. This coterm denotes an observation which, when applied to an element of type `List`, amounts to first evaluating the observer denoted by `empty?` on that element, and then, provided that the result of the evaluation is of type `NeList`, evaluating the observer denoted by `head` on this result.

One writes $Z : s$ for $Z \in \mathcal{C}_s$ and $t : s$ for $t \in T_\Delta[\mathcal{C}]_s$, with $s \in S$. Also, the (S-sorted) set of covariables actually appearing in a coterm $t \in T_\Delta[\mathcal{C}]$ (in general, a subset of $\mathcal{C}$) is denoted $covar(t)$. It then follows by Definition 4.1.3 that $covar(t)$ is finite for any $t \in T_\Delta[\mathcal{C}]$.

Definition 4.1.5. *Let Δ denote a many-sorted cosignature. A coterm $t \in T_\Delta[\mathcal{C}]$ is* **non-identifying** *if it contains at most one occurrence of each covariable in $\mathcal{C}$.*

For a many-sorted cosignature Δ, the S-sorted set of non-identifying Δ-coterms with covariables in $\mathcal{C}$ is denoted $T^1_\Delta[\mathcal{C}]$.

[4] The reason for the last coterm having result type `1 Elt`, rather than `1 1 Elt`, is its use of two occurrences of the same covariable, rather than of two distinct covariables, of sort `1`.

Remark 4.1.6. Since, for a many-sorted cosignature Δ, the sets Δ_s with $s \in S$ are enumerable, and since the sets $T^1_{\Delta,s}$ with $s \in S$ are defined inductively, restricting attention to coterms over some fixed S-sorted set of covariables all of whose components are infinite but enumerable does not reduce the expressiveness of the formalism. We therefore let $\mathcal{C}_0$ denote an S-sorted set of covariables, such that $\mathcal{C}_{0,s}$ is infinite but enumerable for any $s \in S$. Also, for a many-sorted cosignature Δ, we let T^1_Δ denote the set of non-identifying Δ-coterms with covariables in $\mathcal{C}_0$. Then, since both $\mathcal{C}_{0,s}$ with $s \in S$ and Δ_s with $s \in S$ are enumerable, so are the sets $T^1_{\Delta,s}$ with $s \in S$. The elements of T^1_Δ will be referred to simply as *non-identifying Δ-coterms*. The set T^1_Δ will play an important rôle in characterising the elements of final and cofree coalgebras, as well as in proving a completeness result later in this chapter.

Substitution of coterms for covariables is now defined as follows.

Definition 4.1.7. *If $t \in T_\Delta[\{Z_1, \ldots, Z_n\}]_s$ with $Z_i : s_i$ for $i = 1, \ldots, n$, and if $t_i \in T_\Delta[\mathcal{C}]_{s_i}$ for $i = 1, \ldots, n$, the coterm obtained by* **substituting** $t_1, \ldots, t_n$ **for** $Z_1, \ldots, Z_n$ **in** t*, denoted $[t_1/Z_1, \ldots, t_n/Z_n]t$ ($[\overline{t}/\overline{Z}]t$ for short) is defined inductively on the structure of t as follows:*

1. $[\overline{t}/\overline{Z}]Z_i = t_i$ *for* $i = 1, \ldots, n$
2. $[\overline{t}/\overline{Z}]([t'_1, \ldots, t'_m]\delta) = [[\overline{t}/\overline{Z}]t'_1, \ldots, [\overline{t}/\overline{Z}]t'_m]\delta$ *for* $\delta \in \Delta_{s,s'_1 \ldots s'_m}$ *and* $t'_j \in T_\Delta[\{Z_1, \ldots, Z_n\}]_{s'_j}$, $j = 1, \ldots, m$.

Example 4.1.8. Given the cosignature in Example 4.1.2, the coterm obtained by substituting the coterms `F` and `[F,[E]head]empty?` for the covariables `F` and respectively `N` in `[F,[N]tail]empty?` is `[F,[[F,[E]head]empty?]tail]empty?`.

Given $t \in T_\Delta[\{Z_1, \ldots, Z_n\}]$, we write $\underline{t}$ for a coterm with the following properties:

1. $\underline{t} \in T^1_\Delta[\{X_1, \ldots, X_m\}]$
2. $t = [Z_{i_1}/X_1, \ldots, Z_{i_m}/X_m]\underline{t}$, with $Z_{i_1}, \ldots, Z_{i_m} \in \{Z_1, \ldots, Z_n\}$.

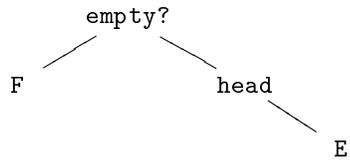
That is, t is obtained from $\underline{t}$ by renaming and possibly identifying some of its covariables. (Note that $\underline{t}$ is only defined up to a bijective renaming of its covariables.)

Remark 4.1.9. Coterms can be represented as trees with the leaves labelled by covariables and with the internal nodes labelled by operation symbols:

1. covariables Z are represented as trees having one node, labelled by Z
2. coterms of form $[t_1, \ldots, t_n]\delta$ are represented as trees having the root labelled by δ and its subtrees given by the trees associated to $t_1, \ldots, t_n$.

A path from the root of the tree associated to a coterm to one of its leaves will be called an *evaluation path* for that coterm.

Example 4.1.10. Given the cosignature in Example 4.1.2, the tree associated to the coterm `[F,[E]head]empty?` is:



with one evaluation path corresponding to an empty list and another corresponding to (the extraction of the first element of) a non-empty list.

4.1.2 Coalgebras, Finality and Bisimilarity

The models of a many-sorted cosignature provide interpretations for its sorts and operation symbols.

Definition 4.1.11. *Let Δ denote a many-sorted cosignature with sort set S. A* **(many-sorted) Δ-coalgebra** *is given by an S-sorted set A together with, for each $\delta : s \to s_1 \dots s_n$ in Δ, a function $\delta_A : A_s \to A_{s_1} + \dots + A_{s_n}$.*

Given Δ-coalgebras A and B, a **Δ-homomorphism** *$f : A \to B$ is given by an S-sorted function $(f_s)_{s \in S}$ with $f_s : A_s \to B_s$ for $s \in S$, additionally satisfying:*

$$[\iota_1 \circ f_{s_1}, \dots, \iota_n \circ f_{s_n}](\delta_A(a)) = \delta_B(f_s(a))$$

for each $\delta : s \to s_1 \dots s_n$ in Δ and each $a \in A_s$, with $s, s_1, \dots, s_n \in S$ (where $\iota_j : B_{s_j} \to B_{s_1} + \dots + B_{s_n}$ for $j = 1, \dots, n$ are the coproduct injections).

For a many-sorted cosignature Δ, the category of Δ-coalgebras and Δ-homomorphisms is denoted Coalg(Δ).

Example 4.1.12. Consider the following coalgebra of the cosignature given in Example 4.1.2:

$$A_{\mathtt{1}} = \{*\}$$
$$A_{\mathtt{Elt}} = \mathbb{N}$$
$$A_{\mathtt{List}} = (\mathbb{N})^*$$
$$A_{\mathtt{NeList}} = (\mathbb{N})^+$$
$$\mathtt{empty?}_A(l) = \begin{cases} \iota_1(*) & \text{if } \ l = \epsilon \\ \iota_2(l) & \text{if } \ l \neq \epsilon \end{cases}$$
$$\mathtt{head}_A(n : l) = n$$
$$\mathtt{tail}_A(n : l) = l$$

with ϵ denoting the empty sequence of natural numbers. This coalgebra provides an implementation of finite lists. An implementation of finite as well as infinite lists can be obtained by replacing the sets of finite (respectively finite, non-empty) sequences of natural numbers in the above definition with the sets of finite or infinite (non-empty, finite or infinite) sequences of natural numbers.

We also note that, at this point, the interpretation of the sort `1` in coalgebras of the list specification can not be constrained to a one-element set. This issue will be dealt with in Section 4.2.

Remark 4.1.13. Many-sorted cosignatures and their coalgebras are instances of the abstract notions of cosignature and coalgebra of an abstract cosignature (see Definitions 3.1.1 and 3.1.4). Specifically, if Δ denotes a many-sorted cosignature with sort set S, and if $\mathsf{G}_\Delta : \mathsf{Set}^S \to \mathsf{Set}^S$ denotes the endofunctor whose components are given by:

$$(\mathsf{G}_\Delta X)_s = \coprod_{\delta \in \Delta_{s,s_1 \ldots s_n}} (X_{s_1} + \ldots + X_{s_n})$$

for $X \in |\mathsf{Set}^S|$ and $s \in S$, then the tuple $(\mathsf{Set}^S, \mathsf{G}_\Delta)$ defines an abstract cosignature whose category of coalgebras is isomorphic to $\mathsf{Coalg}(\Delta)$. For, the category Set^S is complete, cocomplete and regular (see Examples 2.1.31 and 2.1.47), while G_Δ preserves pullbacks and limits of ω^{op}-chains (see Example 2.1.37). Also, Δ-coalgebras A induce Set^S-arrows $\alpha : A \to \mathsf{G}_\Delta A$ (with α_s mapping $a \in A_s$ to $(\delta_A(a))_{\delta \in \Delta_{s,s_1 \ldots s_n}}$, for $s \in S$), and conversely, any such Set^S-arrow defines a Δ-coalgebra structure on its domain; moreover, the two mappings are inverse to each other.

Example 4.1.14. The endofunctor associated to the cosignature given in Example 4.1.2 is of form $\mathsf{G} : \mathsf{Set}^{\{\mathtt{1},\mathtt{Elt},\mathtt{List},\mathtt{NeList}\}} \to \mathsf{Set}^{\{\mathtt{1},\mathtt{Elt},\mathtt{List},\mathtt{NeList}\}}$, with the components given by:

$$\begin{aligned}(\mathsf{G}X)_{\mathtt{1}} &= 1\\ (\mathsf{G}X)_{\mathtt{Elt}} &= 1\\ (\mathsf{G}X)_{\mathtt{List}} &= X_{\mathtt{1}} + X_{\mathtt{NeList}}\\ (\mathsf{G}X)_{\mathtt{NeList}} &= X_{\mathtt{Elt}} + X_{\mathtt{List}}\end{aligned}$$

for $X \in |\mathsf{Set}^{\{\mathtt{1},\mathtt{Elt},\mathtt{List},\mathtt{NeList}\}}|$.

Given a Δ-coalgebra A, a set $\{Z_1, \ldots, Z_n\}$ of covariables with $Z_i : s_i$ for $i = 1, \ldots, n$ and a covariable $Z \in \{Z_1, \ldots, Z_n\}$ with $Z : s$, one writes $\iota_Z : A_s \to A_{s_1} + \ldots + A_{s_n}$ for the corresponding coproduct injection.

The interpretation of Δ-operation symbols by Δ-coalgebras extends to an interpretation of Δ-coterms by Δ-coalgebras.

Definition 4.1.15. *Let Δ denote a many-sorted cosignature, and let $\mathcal{C}$ denote a set of covariables. The* **interpretation** *of a Δ-coterm $t \in T_\Delta[\mathcal{C}]_s$, with $s \in S$, in a Δ-coalgebra A is a function $t_A : A_s \to \coprod_{Z \in \mathcal{C}, Z:s'} A_{s'}$ defined as follows:*

1. *$Z_A = \iota_Z$ for $Z \in \mathcal{C}_s$*
2. *$([t_1, \ldots, t_n]\delta)_A = [(t_1)_A, \ldots, (t_n)_A] \circ \delta_A$ for $\delta \in \Delta_{s,s_1 \ldots s_n}$ and $t_i \in T_\Delta[\mathcal{C}]_{s_i}$, $i = 1, \ldots, n$*

with $[(t_1)_A, \ldots, (t_n)_A] : A_{s_1} + \ldots + A_{s_n} \to \coprod_{Z \in \mathcal{C}, Z:s'} A_{s'}$ denoting the unique Set*-arrow induced by $(t_i)_A : A_{s_i} \to \coprod_{Z \in \mathcal{C}, Z:s'} A_{s'}$ with $i = 1, \ldots, n$.*

Given a many-sorted cosignature Δ with sort set S, the unique Δ-coalgebra structure on the empty S-sorted set yields an initial Δ-coalgebra. However, this coalgebra provides no information about the

behaviours observable using the operations specified by Δ. A characterisation of these behaviours is, instead, given by (the elements of) a final Δ-coalgebra.

The existence of final coalgebras of many-sorted cosignatures is an immediate consequence of Remark 4.1.13 together with Proposition 3.1.9. An alternative proof of the existence of such coalgebras which, in addition, provides a concrete description of their elements is given in the following.

Proposition 4.1.16. *Any many-sorted cosignature Δ admits a final coalgebra.*

Proof. Let F denote the Δ-coalgebra given by:

$$F_s = \{\ \varphi : T^1_{\Delta,s} \to \cup\{\, covar(t) \mid t \in T^1_{\Delta,s} \,\} \mid t \in T^1_{\Delta,s} \Rightarrow \varphi(t) \in covar(t),\\ t, t' \in T^1_{\Delta,s},\ t' = [t_1/Z_1, \ldots, t_n/Z_n]t,\ \varphi(t) = Z_k \Rightarrow \varphi(t') \in covar(t_k)\ \}$$

for $s \in S$, and:

$$\delta_F(\varphi) = \iota_{Z_i}(\varphi'),\ \varphi'(t') = \varphi([Z_1, \ldots, t', \ldots, Z_n]\delta) \text{ for } t' \in T^1_{\Delta,s'} \\ \text{if } \varphi([Z_1, \ldots, Z_n]\delta) = Z_i : s'$$

for $\varphi \in F_s$ and $\delta \in \Delta_{s,s_1 \ldots s_n}$. The correctness of the above definition follows from the observation that, if φ, δ and φ' are as above, and if $t, t' \in T^1_{\Delta,s'}$ are such that $t' = [t_1/Z'_1, \ldots, t_m/Z'_m]t$ and $\varphi'(t) = Z'_k$ (or, equivalently, $\varphi([Z_1, \ldots, t, \ldots, Z_n]\delta) = Z'_k$), then $\varphi \in F_s$ gives $\varphi([Z_1, \ldots, [t_1/Z'_1, \ldots, t_m/Z'_m]t, \ldots, Z_n]\delta) \in covar(t_k)$, that is, $\varphi'(t') \in covar(t_k)$. Thus, the elements of F are suitably-restricted mappings from non-identifying coterms to covariables appearing in them (with, for each such coterm, the choice of covariable determining an evaluation path[5] for that coterm).

Then, F is a final Δ-coalgebra. For, given an arbitrary Δ-coalgebra A, one can define a Δ-homomorphism $f : A \to F$ by: $f(a)(t) = Z_i$ if $t_A(a) \in \iota_{Z_i}(A_{s_i})$, for $t \in T^1_{\Delta,s}$, $a \in A_s$ and $s \in S$. The fact that $f(a) \in F_s$ for any $a \in A_s$ and $s \in S$ follows from the observation that $t_A(a) \in \iota_{Z_i}(A_{s_i})$ implies $([t_1/Z_1, \ldots, t_n/Z_n]t)_A(a) \in \iota_Z(A_{s'})$ for some $Z \in covar(t_i)_{s'}$, for any $t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_s$ and any suitable Δ-coterms $t_i \in T^1_{\Delta,s_i}$ with $i = 1, \ldots, n$. The fact that f is a Δ-homomorphism follows in a similar way. Moreover, any Δ-homomorphism from A to F is necessarily defined in this way. □

Example 4.1.17. The carrier of the final coalgebra of the cosignature in Example 4.1.2 is given by: $F_{\mathtt{1}} = \{*\}$, $F_{\mathtt{Elt}} = \{*\}$, $F_{\mathtt{NeList}} = \mathbb{N}^* \cup \{\infty\}$ and $F_{\mathtt{List}} = \mathbb{N} \cup \{\infty\}$ (with lists of length n being denoted by n, for $n \in \mathbb{N}$, and with infinite lists being denoted by ∞). The reason for this rather unexpected result is that no observers have been provided for the type `Elt`; as a result, list elements can not be distinguished from each other, and all that can be observed about a list is its length.

Remark 4.1.18. For $C \in |\mathsf{Set}^S|$, a cofree Δ-coalgebra A over C is given by:

$$A_s = \{\ \varphi : T^1_{\Delta,s} \to \cup\{\, covar(t) \times C \mid t \in T^1_{\Delta,s} \,\} \mid t \in T^1_{\Delta,s} \Rightarrow \pi_1(\varphi(t)) \in covar(t),\\ t, t' \in T^1_{\Delta,s},\ t' = [t_1/Z_1, \ldots, t_n/Z_n]t,\ \pi_1(\varphi(t)) = Z_k \Rightarrow \pi_1(\varphi(t')) \in covar(t_k)\\ \text{and moreover, } \pi_2(\varphi(t')) = \pi_2(\varphi(t)) \text{ if } t_k \text{ is a covariable }\}$$

[5] See Remark 4.1.9.

for $s \in S$ (where the product $covar(t) \times C$ is taken in Set^S), and:

$$\delta_A(\varphi) = \iota_{Z_i}(\varphi'),\ \ \varphi'(t') = \varphi([Z_1, \ldots, t', \ldots, Z_n]\delta) \text{ for } t' \in T^1_{\Delta,s'}$$
$$\text{if } \varphi([Z_1, \ldots, Z_n]\delta) = \langle Z_i, c_i \rangle \text{ with } Z_i : s'$$

for $\varphi \in A_s$ and $\delta \in \Delta_{s,s_1 \ldots s_n}$. This can be shown similarly to Proposition 4.1.16.

The next result gives a characterisation of the notion of bisimilarity associated to (the endofunctors induced by) many-sorted cosignatures.

Proposition 4.1.19. *Let Δ denote a many-sorted cosignature with sort set S, and let A denote a Δ-coalgebra. Then, given $s \in S$, two states $a, a' \in A_s$ are bisimilar if and only if for any $t \in T^1_{\Delta,s}$, there exist $s' \in S$ and $Z \in covar(t)_{s'}$ such that $t_A(a), t_A(a') \in \iota_Z(A_{s'})$.*

Proof. Recall from Remark 3.1.14 that bisimilarity on an arbitrary coalgebra is given by the kernel of its unique homomorphism into the final coalgebra. Then, according to the proof of Proposition 4.1.16, a and a' are bisimilar (i.e. $f_s(a) = f_s(a')$, with $f : A \to F$ denoting the unique Δ-homomorphism from A to the final Δ-coalgebra) precisely when $t_A(a), t_A(a') \in \iota_Z(A_{s'})$ for some $Z \in covar(t)_{s'}$ and $s' \in S$, for any $t \in T^1_{\Delta,s}$. □

That is, two states of a given coalgebra are bisimilar if and only if, when observed using any non-identifying coterm, the results yielded correspond to the same evaluation path for that coterm.

Example 4.1.20. The notion of bisimilarity associated to the cosignature given in Example 4.1.2 relates two states of a coalgebra if and only if they denote lists with the same number of elements. A finer notion of bisimilarity, which distinguishes between lists with different elements, could be obtained either by providing suitable observers for the sort `Elt` (as already noted in Example 4.1.17), or by fixing the interpretation of the sort `Elt`. (The latter alternative will be considered in Section 4.2.)

The characterisation of bisimilarity given by Proposition 4.1.19 together with the observation that bisimilarity on final coalgebras coincides with the equality relation (see Remark 3.1.14) yield a coinductive technique for proving the equality of observations on the elements of final coalgebras.

Corollary 4.1.21. *Let Δ denote a many-sorted cosignature with sort set S, let F denote a final Δ-coalgebra, and let $l, r \in T_\Delta[\{Z_1, \ldots, Z_n\}]_s$ with $Z_1 : s_1, \ldots, Z_n : s_n$. Then, given $\varphi \in F_s$, $l_F(\varphi) = r_F(\varphi)$ holds if and only if $([t_1/Z_1, \ldots, t_n/Z_n]l)_F(\varphi)$ and $([t_1/Z_1, \ldots, t_n/Z_n]r)_F(\varphi)$ are both in $\iota_Z(F_{s'})$ for some $Z : s'$ and $s' \in S$, for any $t_i \in T^1_{\Delta,s_i}$ with $i = 1, \ldots, n$.*

Proof. The **only if** direction is straightforward. For the **if** direction, it suffices to show that $l_F(\varphi) \sim_F r_F(\varphi)$, with $\sim_F$ denoting Δ-bisimilarity on F (see Remark 3.1.14). Taking $t_i = Z_i$ for $i = 1, \ldots, n$ gives $l_F(\varphi), r_F(\varphi) \in \iota_{Z_i}(F_{s_i})$ for some $i \in \{1, \ldots, n\}$. Then, for any $t \in T^1_{\Delta,s_i}$, taking $t_j = Z_j$ for $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$ and $t_i = t$ in the hypothesis gives $t_F(l_F(\varphi)) = t_F(r_F(\varphi))$. Proposition 4.1.19 now yields $l_F(\varphi) \sim_{F,s_i} r_F(\varphi)$, while Remark 3.1.14 yields $l_F(\varphi) = r_F(\varphi)$. □

4.1.3 The Lawvere Category of a Many-Sorted Cosignature

To each many-sorted cosignature one can associate a category whose objects denote types and whose arrows denote type observations. The construction of this category is dual to that of the Lawvere category associated to an algebraic signature (see Definition 2.3.33).

Definition 4.1.22. *The* **Lawvere category** L^{Δ} *associated to a many-sorted cosignature* Δ *is the least category with finite coproducts which has the sorts in* S *as objects and the operation symbols of* Δ *as arrows.*

That is, the objects of L^{Δ} are of the form $+_{i\in I}s_i$ with I finite and $s_i \in S$ for each $i \in I$, while the arrows of L^{Δ} are of one of the following forms:

- $\delta : s \to s_1 + \ldots + s_n$ with $\delta \in \Delta_{s,s_1\ldots s_n}$
- $\iota_i : s_i \to s_1 + \ldots + s_n$ with $i \in \{1, \ldots, n\}$
- $[l_1, \ldots l_n] : s_1 + \ldots + s_n \to x$ with $l_i : s_i \to x$ for $i = 1, \ldots, n$
- a composition of such arrows.

Remark 4.1.23. The following closure rules can be used to generate $\mathsf{L}_{\Delta}(s, s_1 + \ldots + s_n)$, with $s, s_1, \ldots, s_n \in S$ and $n \in \mathbb{N}^*$:

- $(\iota_i : s_i \to s_1 + \ldots + s_n) \in \mathsf{L}_{\Delta}(s_i, s_1 + \ldots + s_n)$ for $i \in \{1, \ldots, n\}$
- $([l_1, \ldots, l_n]\delta : s \to s'_1 + \ldots + s'_m) \in \mathsf{L}_{\Delta}(s, s'_1 + \ldots + s'_m)$ for $\delta \in \Delta_{s,s_1\ldots s_n}$ and $l_i \in \mathsf{L}_{\Delta}(s_i, s'_1 + \ldots + s'_m)$ for $i = 1, \ldots, n$.

This is a consequence of the fact that $[l_1, \ldots, l_n] \circ \iota_i = l_i$ for any $l_1, \ldots, l_n$ as above and any $i \in \{1, \ldots, n\}$.

Proposition 4.1.24. *Let* Δ *denote a many-sorted cosignature with sort set* S*. There exists a one-to-one correspondence between coterms* $t \in T_{\Delta}[\{Z_1, \ldots, Z_n\}]_s$ *with* $Z_i : s_i$*,* $i = 1, \ldots, n$ *and arrows* $[\![t]\!]$ *in* $\mathsf{L}^{\Delta}(s, s_1 + \ldots + s_n)$*. Moreover,* $[\![[t_1/Z_1, \ldots, t_n/Z_n]t]\!] = [[\![t_1]\!], \ldots, [\![t_n]\!]] \circ [\![t]\!]$ *for any* $t \in T_{\Delta}[\{Z_1, \ldots, Z_n\}]_s$ *and any* $t_i \in T_{\Delta}[\mathcal{C}]_{s_i}$ *for* $i = 1, \ldots, n$*.*

Proof. For $s \in S$, $s_i \in S$ for $i = 1, \ldots, n$ and $n \in \mathbb{N}^*$, let $[\![\_]\!]_s : T_{\Delta}[\{Z_1, \ldots, Z_n\}]_s \to \mathsf{L}_{\Delta}(s, s_1 + \ldots + s_n)$ be given by:

1. $[\![Z_i]\!]_{s_i} = \iota_i$ for $i \in \{1, \ldots, n\}$
2. $[\![[t_1, \ldots, t_k]\delta]\!]_s = [[\![t_1]\!]_{s_1}, \ldots, [\![t_k]\!]_{s_k}] \circ \delta$ for $\delta \in \Delta_{s,s_1\ldots s_k}$ and $t_i \in T_{\Delta}[\{Z_1, \ldots, Z_n\}]_{s_i}$, $i = 1, \ldots, k$.

We first use structural induction to show that for any $l \in \mathsf{L}_{\Delta}(s, s_1 + \ldots + s_n)$, there exists $t \in T_{\Delta}[\{Z_1, \ldots, Z_n\}]_s$ with $Z_i : s_i$ for $i = 1, \ldots, n$ such that $[\![t]\!]_s = l$:

1. If $l = \iota_i$, then $[\![Z_i]\!]_{s_i} = l$.

2. If $l = [l_1, \ldots, l_k] \circ \delta$ with $\delta \in \Delta_{s,s'_1 \ldots s'_k}$ and $[\![t_i]\!] = l_i$ for $i = 1, \ldots, k$, then $[\![[t_1, \ldots, t_k]\delta]\!]_s = [[\![t_1]\!]_{s'_1}, \ldots, [\![t_k]\!]_{s'_k}] \circ \delta = [l_1, \ldots, l_k] \circ \delta = l$.

Hence, $[\![\_]\!]$ is surjective.

We now use structural induction on t to show that $[\![t]\!]_s = [\![t']\!]_s$ implies $t = t'$:

1. $t = Z_i$

   Then, $[\![t]\!]_{s_i} = [\![t']\!]_{s_i}$ together with the definition of $[\![\_]\!]_{s_i}$ yield $[\![t']\!]_{s_i} = \iota_i$, which, by the definition of $[\![\_]\!]_{s_i}$ and of L_Δ, can only hold if $t' = Z_i$. Hence, $t = t'$.

2. $t = [t_1, \ldots, t_k]\delta$

   Then, $[\![t]\!]_s = [\![t']\!]_s$ together with the definition of $[\![\_]\!]_s$ yield $[\![t']\!]_s = [[\![t_1]\!]_{s_1}, \ldots, [\![t_k]\!]_{s_k}]\delta$. The definitions of $[\![\_]\!]_s$ and L_Δ give $t' = [t'_1, \ldots, t'_k]\delta$ for some $t'_1, \ldots, t'_k$ satisfying $[\![t_i]\!]_{s_i} = [\![t'_i]\!]_{s_i}$ for $i = 1, \ldots, k$. The induction hypothesis then gives $t_i = t'_i$ for $i = 1, \ldots, k$, which implies $[t_1, \ldots, t_k]\delta = [t'_1, \ldots, t'_k]\delta$, i.e. $t = t'$.

Hence $[\![\_]\!]$ is injective. Therefore, $[\![\_]\!]$ is a one-to-one correspondence.

The fact that $[\![[t_1/Z_1, \ldots, t_n/Z_n]t]\!] = [[\![t_1]\!], \ldots, [\![t_n]\!]] \circ [\![t]\!]$ follows, again, by structural induction on t. □

As in many-sorted algebra, there exists a correspondence between coalgebras of a many-sorted cosignature and functors on its Lawvere category.

Proposition 4.1.25. *For a many-sorted cosignature Δ, Δ-coalgebras A are in one-to-one correspondence with coproduct-preserving functors* $\mathsf{A} : \mathsf{L}^\Delta \to \mathsf{Set}$, *while Δ-homomorphisms $f : A \to B$ are in one-to-one correspondence with natural transformations* $\mathsf{f} : \mathsf{A} \Rightarrow \mathsf{B}$. *Furthermore,* $\mathsf{A}[\![t]\!] = t_A$ *for any Δ-coalgebra A and any Δ-coterm t.*

Proof (sketch). Any coproduct-preserving functor $\mathsf{A} : \mathsf{L}^\Delta \to \mathsf{Set}$ defines a Δ-coalgebra A (with the carrier of A being given by $A_s = \mathsf{A}s$ for $s \in S$, and with the operations of A being given by $\delta_A = \mathsf{A}\delta$ for $\delta \in \Delta_{s,s_1 \ldots s_n}$), and conversely. □

4.1.4 Coalgebraic Equational Specification

In algebraic specification, many-sorted equations are used to constrain the interpretation of terms by algebras. A similar approach proves suitable for constraining state observations, provided that one's interest is in relating different observations of the same state. This section presents such an approach, illustrating the kinds of constraints specifiable within it.

A first approximation of the notion of coequation is given by a pair of coterms of the same sort. Satisfaction of a coequation by a coalgebra then corresponds to the coalgebra providing identical interpretations for the two coterms. For instance, a coequation of form:

```
[[F,[E]head]empty?]tail = [E]head
```

constrains the interpretation of the sort `NeList` in coalgebras A satisfying it to constant, infinite lists, by requiring the observations:

$[\iota_{\mathtt{F}}, \iota_{\mathtt{E}} \circ \mathtt{head}_A] \circ \mathtt{empty?}_A \circ \mathtt{tail}_A : A_{\mathtt{NeList}} \to A_1 + A_{\mathtt{Elt}}$
$\iota_{\mathtt{E}} \circ \mathtt{head}_A : A_{\mathtt{NeList}} \to A_1 + A_{\mathtt{Elt}}$

(with $\iota_{\mathtt{F}} : A_1 \to A_1 + A_{\mathtt{Elt}}$ and $\iota_{\mathtt{E}} : A_{\mathtt{Elt}} \to A_1 + A_{\mathtt{Elt}}$ denoting the two injections) to yield similar results on any non-empty list.

However, due to the presence of choice in the result types of observers, reasoning about the satisfaction of coequations by algebras may involve some case analysis on the possible evaluation paths of coterms. For instance, in order to infer the satisfaction of the coequation:

```
[[F,N]empty?]tail = [[F',N]empty?]tail
```

(constraining $\mathtt{empty?}_A \circ \mathtt{tail}_A$ to always yield a result of type `NeList`, by requiring the observations:

$[\iota_{\mathtt{F}}, \iota_{\mathtt{N}}] \circ \mathtt{empty?}_A \circ \mathtt{tail}_A : A_{\mathtt{NeList}} \to A_1 + A_1 + A_{\mathtt{NeList}}$
$[\iota_{\mathtt{F}'}, \iota_{\mathtt{N}}] \circ \mathtt{empty?}_A \circ \mathtt{tail}_A : A_{\mathtt{NeList}} \to A_1 + A_1 + A_{\mathtt{NeList}}$

to yield similar results on any non-empty list) from the satisfaction of the coequation:

```
[[F,[E]head]empty?]tail = [E]head
```

a case analysis on the possible evaluation paths for `[[F,N]empty?]tail` should be carried out. Specifically, the satisfaction of the above coequation would follow from its satisfaction both under the assumption that the evaluation path corresponds to the covariable `N` (in which case the proof is trivial), and under the assumption that the evaluation path corresponds to the covariable `F` (in which case a contradiction can be inferred from the assumption together with the satisfaction of the initial coequation, and consequently the assumption is invalidated). It turns out that, in order to obtain a complete deduction calculus for coequations, such assumptions must be incorporated in the notion of coequation. In this case, the satisfaction of the coequation:

```
[[F,N]empty?]tail = [[F',N]empty?]tail
```

would follow from the satisfaction of the following two (conditional) coequations:

```
[[F,N]empty?]tail = [[F',N]empty?]tail if ([[F,N]empty?]tail,N)
[[F,N]empty?]tail = [[F',N]empty?]tail if ([[F,N]empty?]tail,F)
```

The presence of conditions in the above coequations is intended to specify the fact that the interpretations of the lhs and rhs of the coequations are only required to coincide on non-empty lists satisfying these conditions (i.e. on non-empty lists on which the evaluation of `[[F,N]empty?]tail` yields a result of type `NeList`, respectively `1`).

The above discussion justifies the following definition.

Definition 4.1.26. *Let Δ denote a many-sorted cosignature with sort set S, and let $s \in S$. A* **Δ-coequation of sort** *s is a tuple (l, r, C), with $l, r \in T_\Delta[\mathcal{C}]_s$, and with $C = \{(t_1, \mathcal{C}'_1), \ldots, (t_n, \mathcal{C}'_n)\}$ for some $t_i \in T_\Delta[\mathcal{C}_i]_s$ and $\mathcal{C}'_i \subseteq \mathcal{C}_i$, with $i = 1, \ldots, n$. A Δ-coalgebra A* **satisfies** *a Δ-coequation e of the above form (written $A \models_\Delta e$) if and only if $l_A(a) = r_A(a)$ holds whenever $a \in A_s$ is such that $(t_i)_A(a) \in \iota_{Z_i}(A_{s_i})$ for some $Z_i \in (\mathcal{C}'_i)_{s_i}$, for $i = 1, \ldots, n$ (in which case a is said to* **satisfy** *the conditions C).*

The coequation (l, r, C) (with C being as in Definition 4.1.26) is alternatively denoted $l = r$ if $(t_1, \mathcal{C}'_1), \ldots, (t_n, \mathcal{C}'_n)$. Also, if $\mathcal{C}'_i = \{Z_i\}$, one writes (t_i, Z_i) as a shorthand for $(t_i, \mathcal{C}'_i)$. Finally, given a set E of Δ-coequations together with $s \in S$, one writes E_s for the subset of E consisting of coequations of sort s.

Remark 4.1.27. The notion of coequation could have alternatively been defined as a tuple of form $((l, r), \{(t_1, Z_1), \ldots, (t_n, Z_n)\})$, with $t_i \in T_\Delta[\mathcal{C}_i]$ and $Z_i \in \mathcal{C}_i$ for $i = 1, \ldots, n$. The notion of coequation in Definition 4.1.26 acts as a shorthand for *sets* of coequations of the above form – a coequation of form $l = r$ if $(t_1, \mathcal{C}'_1), \ldots, (t_n, \mathcal{C}'_n)$ is semantically equivalent to the following set of coequations: $\{\ l = r$ if $(t_1, Z_1), \ldots, (t_n, Z_n) \mid Z_1 \in \mathcal{C}'_1, \ldots, Z_n \in \mathcal{C}'_n\ \}$.

Example 4.1.28. Given the cosignature in Example 4.1.2, the coequation:

```
[[F,[E]head]empty?]tail = [E]head
```

constrains the interpretation of `NeList` to constant, infinite lists, while the coequation:

```
[[F,[E]head]empty?]tail = [E]head if ([[F,N]empty?]tail,N)
```

constrains the interpretation of `NeList` to constant, finite or infinite lists. (The presence of a condition in the second coequation allows the constraint in the main part of the coequation to only be imposed on non-empty lists having at least two elements). Similarly, the coequation:

```
[[F,[[F',[E]head]empty?]tail]empty?]tail = [E]head
```

constrains the interpretation of `NeList` to alternating, infinite lists, while the coequation:

```
[[F,[[F',[E]head]empty?]tail]empty?]tail = [E]head
     if ([[F,[[F',N]empty?]tail]empty?]tail,N)
```

constrains the interpretation of `NeList` to alternating, finite or infinite lists.

Example 4.1.29. An alternative specification of lists can be given using sorts `1`, `Elt` and `List`, and operation symbols `first` : `List` $\to$ `1 Elt` and `rest` : `List` $\to$ `1 List`, subject to constraints captured by the following coequations:

```
[Z,L]rest = [Z,L']rest if ([Z,E]first,Z)
[Z,E]first = [Z,E']first if ([Z,L]rest,Z)
```

formalising the fact that a list is either empty, in which case it has neither head nor tail, or non-empty, in which case it has both a head and a tail.

Remark 4.1.30. The notions of Δ-coterm, Δ-coequation and satisfaction of Δ-coequations by Δ-coalgebras are instances of the abstract notions of observer, coequation and coequational satisfaction (see Definitions 3.1.23 and 3.1.25). For, given a Δ-coterm $t \in T_\Delta[\mathcal{C}]_s$ with $\mathcal{C}$ a finite set of covariables[6], one can define a $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-observer (K, t') having the property that t_A coincides with $t'_{\langle A,\alpha\rangle}$ for each Δ-coalgebra A with $\langle A, \alpha\rangle$ the associated $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-coalgebra. Specifically,

[6] For instance, $\mathcal{C}$ can be taken to be $covar(t)$.

$\mathsf{K} : \mathsf{Set}^S \to \mathsf{Set}^S$ is given by:

$$(\mathsf{K}X)_s = \coprod_{s'' \in S} \coprod_{Z \in \mathcal{C}_{s''}} X_{s''}$$
$$(\mathsf{K}X)_{s'} = 1, \quad \text{for } s' \in S \setminus \{s\}$$

for $X \in |\mathsf{Set}^S|$, while $t' : \mathsf{U} \Rightarrow \mathsf{KU}$ (with $\mathsf{U} : \mathsf{Coalg}(\mathsf{Set}^S, \mathsf{G}_\Delta) \to \mathsf{Set}^S$ denoting the functor taking $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-coalgebras to their carrier) is given by:

$$(t'_{\langle A,\alpha\rangle})_s(a) = t_A(a)$$
$$(t'_{\langle A,\alpha\rangle})_{s'}(a') = *, \quad \text{for } s' \in S \setminus \{s\}$$

for each $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-coalgebra $\langle A, \alpha\rangle$ with A its corresponding Δ-coalgebra. (The fact that K preserves monomorphisms follows by Example 2.1.37.)

Also, given a Δ-coterm $t \in T_\Delta[\mathcal{C}]_s$, with $\mathcal{C}$ a finite set of covariables, together with some conditions C of form $(t_1, \mathcal{C}_1), \ldots, (t_n, \mathcal{C}_n)$ for the sort s, one can define a $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-observer (K, t'), with $\mathsf{K} : \mathsf{Set}^S \to \mathsf{Set}^S$ being given by:

$$(\mathsf{K}X)_s = 1 + \coprod_{s'' \in S} \coprod_{Z \in \mathcal{C}_{s''}} X_{s''}$$
$$(\mathsf{K}X)_{s'} = 1, \quad \text{for } s' \in S \setminus \{s\}$$

for $X \in |\mathsf{Set}^S|$, and with $t' : \mathsf{U} \Rightarrow \mathsf{KU}$ being given by:

$$(t'_{\langle A,\alpha\rangle})_s(a) = \begin{cases} \iota_2(t_A(a)) & \text{if } C \text{ holds in } a \\ \iota_1(*) & \text{otherwise} \end{cases}$$
$$(t'_{\langle A,\alpha\rangle})_{s'}(a') = *, \quad \text{for } s' \in S \setminus \{s\}$$

for each $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-coalgebra $\langle A, \alpha\rangle$ with A its corresponding Δ-coalgebra. (The fact that K preserves monomorphisms follows, again, by Example 2.1.37. Also, note that K only depends on $\mathcal{C}$.) Now, if e denotes a Δ-coequation of form $l = r$ if C, and if (K, l') and (K, r') denote the $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-observers induced by the Δ-coterms l and r together with the conditions C, then $A \models_\Delta e$ is equivalent to $\langle A, \alpha\rangle \models_{(\mathsf{Set}^S, \mathsf{G}_\Delta)} (\mathsf{K}, l', r')$.

However, unlike in the algebraic case (see Example 3.2.13), $(\mathsf{Set}^S, \mathsf{G}_\Delta)$-observers whose type is of the previous form do not, in general, yield Δ-coterms. The reason for this is that, while Σ-terms (with Σ a many-sorted signature) can be identified with the elements of certain free Σ-algebras, a similar observation does not hold for Δ-coterms.

Given $\delta \in \Delta_{s, s_1 \ldots s_n}$ together with $i \in \{1, \ldots, n\}$ and conditions C of form $(t_1, \mathcal{C}'_1), \ldots, (t_m, \mathcal{C}'_m)$ for the sort s_i, one writes $[Z_1, \ldots, C, \ldots, Z_n]\delta$ as a shorthand for $([Z_1, \ldots, t_j, \ldots, Z_n]\delta, \mathcal{C}'_j \cup \{Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n\})_{j=1,\ldots,m}$, with $\{Z_1, \ldots, Z_n\} \cap covar(t_j) = \emptyset$ for $j = 1, \ldots, m$. That is, the conditions $[Z_1, \ldots, C, \ldots, Z_n]\delta$ require the result yielded by (the interpretation of) δ to satisfy the conditions C whenever the evaluation path for $[Z_1, \ldots, Z_n]\delta$ corresponds to the covariable $Z_i : s_i$. Also, given $t \in T_\Delta[\{Z_1, \ldots, Z_n\}]_s$ with $Z_k : s_k$ for $k = 1, \ldots, n$, and given i and

C as before, one writes $[Z_1/Z_1, \ldots, C/Z_i, \ldots, Z_n/Z_n]t$ for $([Z_1/Z_1, \ldots, t_j/Z_i, \ldots, Z_n/Z_n]t, \mathcal{C}'_j \cup \{Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n\})_{j=1,\ldots,m}$. This notation will be used in Section 4.1.6 in formulating a deduction calculus for coequations and in proving its completeness for the satisfaction of coequations by coalgebras.

While in many-sorted algebra equations of form $X = X'$ are only satisfied by algebras whose corresponding carrier is a one-element set, here coequations of form $Z = Z'$ are only satisfied by coalgebras whose corresponding carrier is empty. More generally, coequations of form $l = r$ with $covar(l) \neq covar(r)$ constrain the result type of l and r to the type of a covariable appearing in both l and r. Amongst such coequations, of particular interest are those with l and r being the same up to a renaming of their covariables.

Definition 4.1.31. *Let Δ denote a many-sorted cosignature, let $t \in T_\Delta[\mathcal{C}]_s$ for some set $\mathcal{C}$ of covariables and some $s \in S$, and let $\mathcal{C}' \subseteq \mathcal{C}$. The coequation:*

$$\underline{t} = [y_1/X_1, \ldots, y_m/X_m]\underline{t}$$

where:

- *$\underline{t} \in T^1_\Delta[\{X_1, \ldots, X_m\}]$ is such that $t = [Z_{i_1}/X_1, \ldots, Z_{i_m}/X_m]\underline{t}$*
- *$y_j = \begin{cases} X_j & \text{if } Z_{i_j} \in \mathcal{C}' \\ Y_j \neq X_j & \text{if } Z_{i_j} \notin \mathcal{C}' \end{cases}$ for $j = 1, \ldots, m$*

is called a **type constraint** *for t and is denoted $c(t, \mathcal{C}')$.*

$c(t, \mathcal{C}')$ constrains the result type of t to the type of a covariable in $\mathcal{C}'$: given a Δ-coalgebra A, $c(t, \mathcal{C}')$ holds in a state $a \in A_s$ if and only if $t_A(a) \in \iota_Z(A_{s'})$ for some $Z \in \mathcal{C}'_{s'}$.

If $\mathcal{C}' = \{Z\}$, $c(t, \mathcal{C}')$ is alternatively denoted $c(t, Z)$.

We note that, since $\underline{t}$ is only defined up to a bijective renaming of its covariables, so are the type constraints for t. Consequently, the covariables X_j and Y_j with $j = 1, \ldots, m$ can be arbitrarily chosen, provided that they are all distinct. This observation will be used when proving a completeness result for the satisfaction of coequations by coalgebras.

Remark 4.1.32. If $t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]$ and $i \in \{1, \ldots, n\}$, then $c(t, Z_i)$ has the form:

$$t = [Y_1/Z_1, \ldots, Z_i/Z_i, \ldots, Y_n/Z_n]t$$

Example 4.1.33. Given the cosignature in Example 4.1.2, $c($`[[F,N]empty?]tail,N`$)$ has the form:

```
[[F,N]empty?]tail = [[F',N]empty?]tail
```

and constrains the interpretation of the sort `NeList` in coalgebras satisfying this coequation to infinite lists (by requiring the interpretation of `tail` in such a coalgebra to always yield a non-empty list).

The next result states some immediate properties of the notion of satisfaction of coequations[7].

[7] A slightly weaker version of this result can be obtained as an instance of Proposition 3.1.28, by exploiting the correspondence between many-sorted cosignatures and abstract cosignatures, and between many-sorted coequations and abstract coequations.

Proposition 4.1.34. *Let A and B denote Δ-coalgebras, let $f : A \to B$ denote a Δ-homomorphism, and let e denote a Δ-coequation. Then:*

1. *$A \models_\Delta e$ implies $\mathsf{Im}(f) \models_\Delta e$.*
2. *If all the components of f corresponding to sorts of covariables appearing in e are injective, then $B \models_\Delta e$ implies $A \models_\Delta e$.*

Proof (sketch). The fact that $t_{\mathsf{Im}(f)}(f_s(a)) = [\iota_1 \circ f_{s_1}, \ldots, \iota_n \circ f_{s_n}](t_A(a))$ for each $s \in S$, $t \in T_\Delta[\{Z_1, \ldots, Z_n\}]_s$ with $Z_1 : s_1, \ldots, Z_n : s_n$ and $a \in A_s$ is used. (This is a consequence of the definition of Δ-homomorphisms.) $\square$

Corollary 4.1.35. *Let Δ denote a many-sorted cosignature, and let E denote a set of Δ-coequations. The full subcategory of* Coalg(Δ) *whose objects satisfy the coequations in E is a covariety.*

The fact that many-sorted cosignatures and their coalgebras and coequations are instances of the corresponding abstract notions introduced in Section 3.1 also results in the existence of largest subcoalgebras satisfying given sets of coequations (see Proposition 3.1.33). The next result provides a concrete description of such subcoalgebras.

Proposition 4.1.36. *Let Δ denote a many-sorted cosignature, let E denote a set of Δ-coequations, and let A denote a Δ-coalgebra. Then, the carrier of the largest Δ-subcoalgebra A_E of A satisfying the coequations in E is given by:*

$$\begin{aligned} A_{E,s} = \{\ a \in A_s \mid\ & l_A(t_A(a)) = r_A(t_A(a)) \text{ whenever} \\ & t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_s, i \in \{1, \ldots, n\} \text{ and } (l = r \text{ if } C) \in E_{s_i} \\ & \text{are such that } t_A(a) \in \iota_{Z_i}(A_{s_i}) \text{ and } C \text{ holds in } t_A(a)\ \},\quad s \in S \end{aligned}$$

Proof. To show that the S-sorted set $(A_{E,s})_{s \in S}$ defines a Δ-subcoalgebra of A, let $a \in A_{E,s}$ and $\delta \in \Delta_{s,s_1 \ldots s_m}$. Say $\delta_A(a) \in \iota_j(A_{s_j})$ with $j \in \{1, \ldots, m\}$. Then, given $t' \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_{s_j}$ and $(l = r \text{ if } C) \in E$ such that C holds in $t'_A(\delta_A(a))$, taking $t = [X_1, \ldots, X_{j-1}, t', X_{j+1}, \ldots, X_m]\delta$ in the definition of A_E gives $l_A(t'_A(\delta_A(a))) = r_A(t'_A(\delta_A(a)))$. That is, $\delta_A(a) \in \iota_j(A_{E,s_j})$.

Also, given an arbitrary (Δ, E)-subcoalgebra A' of A, the inclusion $\iota_{A'}$ of A' into A factors through the inclusion ι_{A_E} of A_E into A. (Proposition 4.1.34 gives $\mathsf{Im}(\iota_{A'}) \models_\Delta E$, which, together with the definition of A_E, gives $\mathsf{Im}(\iota_{A'}) \subseteq A_E$.) Hence, A_E is the largest (Δ, E)-subcoalgebra of A. $\square$

4.1.5 An Institution of Many-Sorted Coalgebras

Translations between many-sorted cosignatures are captured by *many-sorted cosignature morphisms*.

Definition 4.1.37. *Let Δ and Δ' denote many-sorted cosignatures with sort sets S and respectively S'. A* **(many-sorted) cosignature morphism** *from Δ to Δ' consists of a function $\phi : S \to S'$ together with an $S \times S^+$-sorted function $(\phi_{s,w})_{s \in S, w \in S^+}$, with $\phi_{s,w} : \Delta_{s,w} \to \Delta'_{\phi(s),\phi^+(w)}$ for $s \in S$ and $w \in S^+$ (with ϕ^+ denoting the pointwise extension of $\phi : S \to S'$ to a function from S^+ to S'^+).*

The category of many-sorted cosignatures and cosignature morphisms is denoted Cosign.

Cosignature morphisms $\phi : \Delta \to \Delta'$ induce reduct functors $\mathsf{U}_\phi : \mathsf{Coalg}(\Delta') \to \mathsf{Coalg}(\Delta)$, taking Δ'-coalgebras A' to the Δ-coalgebras whose carrier is given by $A = (A'_{\phi(s)})_{s \in S}$, and whose operations are given by $\delta_A = \phi(\delta)_{A'}$ for $\delta \in \Delta$.

Remark 4.1.38. Many-sorted cosignature morphisms are an instance of the abstract notion of cosignature morphism (see Definition 3.1.47). For, many-sorted cosignature morphisms $\phi : (S, \Delta) \to (S', \Delta')$ induce abstract cosignature morphisms (U, η_ϕ), with $\mathsf{U} : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ taking $X \in |\mathsf{Set}^{S'}|$ to $(X_{\phi(s)})_{s \in S} \in |\mathsf{Set}^S|$, and with $\eta_\phi : \mathsf{UG}_{\Delta'} \to \mathsf{G}_\Delta \mathsf{U}$ being given by:

$$(\eta_{\phi,X})_s((x_{\delta'})_{\delta' \in \Delta'_{\phi(s),s'_1 \ldots s'_n}}) = (x_{\phi(\delta)})_{\delta \in \Delta_{s,s_1 \ldots,s_m}}$$

for $X \in |\mathsf{Set}^{S'}|$ and $s \in S$. The functor U preserves pullbacks and limits of ω^{op}-chains (see Example 2.1.38) and has a right adjoint R (see Example 2.1.52). And if, in addition, ϕ is injective on sorts, then U lifts pullbacks and limits of ω^{op}-chains (see Example 2.1.38), while R is also a right inverse to U (see Example 2.1.52). Moreover, U_{η_ϕ} agrees with U_ϕ, while the translation along ϕ of Δ-coequations agrees with the translation along η_ϕ of the induced G_Δ-coequations. (The existence of a one-to-one correspondence between Δ-algebras and G_Δ-coalgebras, and of a correspondence between Δ-coequations and G_Δ-coequations, see Remarks 4.1.13 and 4.1.30, are used here.)

An immediate consequence of the above observation is the existence of cofree coalgebras w.r.t. the reduct functors induced by many-sorted cosignature morphisms.

Proposition 4.1.39. *Let $\phi : \Delta \to \Delta'$ denote a many-sorted cosignature morphism. Then,* $\mathsf{U}_\phi : \mathsf{Coalg}(\Delta') \to \mathsf{Coalg}(\Delta)$ *has a right adjoint.*

Proof. The conclusion follows from Remark 4.1.38 and Theorem 3.1.62. □

On the one hand, the mapping from many-sorted cosignatures to their categories of coalgebras extends to a functor $\mathsf{Coalg} : \mathsf{Cosign} \to \mathsf{CAT}^{\mathsf{op}}$. On the other hand, the translation of sorts and operation symbols along many-sorted cosignature morphisms extends to a translation of coterms and hence of coequations, yielding a functor $\mathsf{Coeqn} : \mathsf{Cosign} \to \mathsf{SET}$. Moreover, the functors Coalg and Coeqn agree with the restrictions to Cosign of the corresponding abstract functors described in Section 3.1.3. Then, Proposition 3.1.48 yields the following.

Theorem 4.1.40. $(\mathsf{Cosign}, \mathsf{Coalg}, \mathsf{Coeqn}, \models)$ *is an institution.*

This institution will be called *many-sorted coalgebra*[8]. Its specifications and specification morphisms will be referred to as *(many-sorted) coalgebraic specifications* and respectively *(many-sorted) coalgebraic specification morphisms*.

[8] The use of this terminology is justified by the syntactic duality w.r.t. many-sorted algebra.

Example 4.1.41. Extending the list specification given in Example 4.1.29 with operation symbols `odd_select` : `List` → `List` and `even_select` : `List` → `List` (denoting the operations which select the elements situated in odd and respectively even positions in a list) and coequations:

```
[[Z,E]first]odd_select = [Z,E]first
[[Z,L]rest]odd_select = [Z,[Z,[L]odd_select]rest]rest
          if ([Z,[T,L]rest]rest,{Z,L})
[[Z,L]rest]odd_select = [Z,L]rest
          if ([Z,[T,L]rest]rest,{T})

[[Z,E]first]even_select = [Z,[Z,E]first]rest
[[Z,L]rest]even_select = [Z,[Z,[L]even_select]rest]rest
```

yields an inclusion cosignature morphism which at the same time defines a coalgebraic specification morphism from the list specification given in Example 4.1.29 to the extended list specification given above. We note that the use of case analysis in the definition of `odd_select` is required to ensure that, in coalgebras of the target specification, the list invariant (given by the two coequations in Example 4.1.29) is preserved by `odd_select`. (The preservation of the list invariant amounts to the largest subcoalgebras w.r.t. `first` and `rest` which satisfy the list invariant also defining subcoalgebras w.r.t. `odd_select`.) The way in which the definition of `odd_select` ensures the preservation of the list invariant is to "stop" when an empty list is reached[9] rather than follow the link from an empty list to its (non-existent) tail, as this would possibly result in a list which has a head but does not have a tail. Note that no case analysis is necessary in the definition of `even_select`.

As in the abstract setting, suitable denotations for coalgebraic specifications and coalgebraic specification morphisms are provided by final and respectively cofree coalgebras.

Proposition 4.1.42. *Any coalgebraic specification* (Δ, E) *admits a final coalgebra.*

Proof (sketch). A final (Δ, E)-coalgebra is obtained as the largest (Δ, E)-subalgebra (see Proposition 4.1.36) of the final Δ-coalgebra (see Proposition 4.1.16). □

Proposition 4.1.43. *Let* $\phi : (\Delta, E) \to (\Delta', E')$ *denote a coalgebraic specification morphism. Then,* $U_\phi{\restriction}_{\mathsf{Coalg}(\Delta', E')}$: $\mathsf{Coalg}(\Delta', E') \to \mathsf{Coalg}(\Delta, E)$ *has a right adjoint.*

Proof (sketch). Propositions 4.1.39 and 4.1.36 are used. □

Corollary 4.1.44. *Let* (Δ, E) *denote a coalgebraic specification. Then, for any* $C \in |\mathsf{Set}^S|$*, there exists a cofree* (Δ, E)*-coalgebra over* C*.*

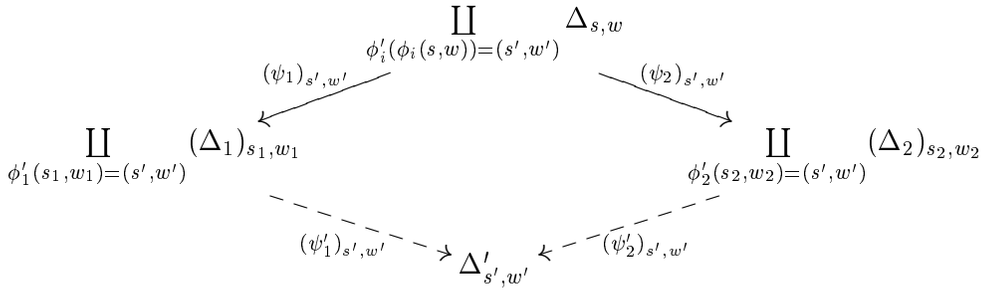
We conclude this section with a discussion on the existence of finite colimits in Cosign. We begin by noting that Theorem 3.1.58 does not guarantee the existence of such colimits, as only many-sorted cosignature morphisms whose sort components are injective induce strong abstract cosignature morphisms. Theorem 3.1.58 does not even immediately result in the existence of finite colimits in

[9] An empty list is characterised by the absence of a head and a tail.

the category of many-sorted cosignatures and cosignature morphisms whose sort components are injective. For this, one would have to prove that the colimiting cone in the category of abstract cosignatures and strong abstract cosignature morphisms belongs to the subcategory whose objects correspond to many-sorted cosignatures and whose arrows correspond to many-sorted cosignature morphisms which are injective on sorts. Nevertheless, one can show directly that finite colimits exist in Cosign, and moreover, that the colimits of finite diagrams which only involve cosignature morphisms whose sort components are injective agree with the colimits of the corresponding diagrams in the category of abstract cosignatures and strong abstract cosignature morphisms.

Theorem 4.1.45. Cosign *is finitely cocomplete.*

Proof (sketch). An initial object in Cosign is given by the cosignature with no sorts and no operation symbols. Also, given many-sorted cosignature morphisms $\phi_1 : (S, \Delta) \to (S_1, \Delta_1)$ and $\phi_2 : (S, \Delta) \to (S_2, \Delta_2)$, their pushout in Cosign is given by the many-sorted cosignature morphisms $\phi_1' : (S_1, \Delta_1) \to (S', \Delta')$ and $\phi_2' : (S_2, \Delta_2) \to (S', \Delta')$, with $\phi_1' : S_1 \to S'$ and $\phi_2' : S_2 \to S'$ defining a pushout for $\phi_1 : S \to S_1$ and $\phi_2 : S \to S_2$ in Set, and with $(\phi_i')_{s_i,w_i} : (\Delta_i)_{s_i,w_i} \to \Delta'_{\phi_i'(s_i),\phi_i'^+(w_i)}$ being given by $(\phi_i')_{s_i,w_i} = (\psi_i')_{\phi_i'(s_i),\phi_i'^+(w_i)} \circ \iota_{s_i,w_i}$ for $(s_i, w_i) \in S_i \times S_i^+$, for $i = 1, 2$, where $(\psi_i')_{s',w'} : \coprod_{\phi_i'(s_i,w_i)=(s',w')} (\Delta_i)_{s_i,w_i} \to \Delta'_{s',w'}$ for $i = 1, 2$ define a pushout for $(\psi_i)_{s',w'} : \coprod_{\phi_i'(\phi_i(s,w))=(s',w')} \Delta_{s,w} \to \coprod_{\phi_i'(s_i,w_i)=(s',w')} (\Delta_i)_{s_i,w_i}$, $(\psi_i)_{s',w'}(\iota_{s,w}(\delta)) = \iota_{\phi_i(s),\phi_i^+(w)}(\phi_i(\delta))$ for $i = 1, 2$, for $(s', w') \in S' \times S'^+$:



□

Corollary 2.2.6 now yields the following.

Corollary 4.1.46. *The category of many-sorted coalgebraic specifications and coalgebraic specification morphisms is finitely cocomplete.*

Remark 4.1.47. The constructions of finite colimits in Cosign (see the proof of Theorem 4.1.45) and respectively in the category of abstract cosignatures and strong abstract cosignature morphisms (see the proof of Theorem 3.1.58) can be used to show that these colimits agree on diagrams only involving many-sorted cosignature morphisms whose sort components are injective. On the one hand, the abstract cosignature associated to the initial many-sorted cosignature is an initial abstract cosignature. Also, if the many-sorted cosignature morphisms ϕ_1 and ϕ_2 considered in the proof of Theorem 4.1.45 are injective on sorts, then so are ϕ_1' and ϕ_2', and moreover, the abstract cosignature morphisms associated to ϕ_1' and ϕ_2' define a pushout for $(\mathsf{U}_1, \eta_{\phi_1})$ and $(\mathsf{U}_2, \eta_{\phi_2})$ in the category of

abstract cosignatures and strong abstract cosignature morphisms (where $(\mathsf{U}_1, \eta_{\phi_1})$ and $(\mathsf{U}_2, \eta_{\phi_2})$ denote the abstract cosignature morphisms induced by ϕ_1 and respectively ϕ_2).

A consequence of the preceding remark is that the restriction of Coalg : Cosign $\to$ CAT$^{\mathsf{op}}$ to many-sorted cosignature morphisms whose sort components are injective preserves finite colimits. However, the construction of finite colimits in Cosign given in the proof of Theorem 4.1.45 together with the observation that, if $\phi_1' : S_1 \to S'$, $\phi_2' : S_2 \to S'$ define a pushout in Set of $\phi_1 : S \to S_1$, $\phi_2 : S \to S_2$, then $\mathsf{U}_{\phi_1'} : \mathsf{Set}^{S'} \to \mathsf{Set}^{S_1}$, $\mathsf{U}_{\phi_2'} : \mathsf{Set}^{S'} \to \mathsf{Set}^{S_2}$ define a pullback in CAT of $\mathsf{U}_{\phi_1} : \mathsf{Set}^{S_1} \to \mathsf{Set}^{S}$, $\mathsf{U}_{\phi_2} : \mathsf{Set}^{S_2} \to \mathsf{Set}^{S}$ can be used to show that in fact a stronger result holds.

Theorem 4.1.48. *The functor* Coalg : Cosign $\to$ CAT$^{\mathsf{op}}$ *preserves finite colimits.*

4.1.6 Deduction

We are now in the position to formulate a sound and complete deduction calculus for coequations. We consider the following deduction rules:

$$[\,\textbf{base}\,]\ \frac{}{E \vdash e}\quad e \in E$$

$$[\,\textbf{cond}\,]\ \frac{}{E \vdash c(t, \mathcal{C}) \text{ if } (t, \mathcal{C})}$$

$$[\,\textbf{weakening}\,]\ \frac{E \vdash t = t' \text{ if } C}{E \vdash t = t' \text{ if } C, C'}$$

$$[\,\textbf{reflexivity}\,]\ \frac{}{E \vdash t = t}$$

$$[\,\textbf{symmetry}\,]\ \frac{E \vdash t = t' \text{ if } C}{E \vdash t' = t \text{ if } C}$$

$$[\,\textbf{transitivity}\,]\ \frac{E \vdash t = t' \text{ if } C,\quad E \vdash t' = t'' \text{ if } C}{E \vdash t = t'' \text{ if } C}$$

$$[\,\textbf{closure}\,]\ \frac{E \vdash t_1 = t_1' \text{ if } C_1, \ldots, E \vdash t_n = t_n' \text{ if } C_n}{E \vdash [t_1, \ldots, t_n]\delta = [t_1', \ldots, t_n']\delta \text{ if } [C_1, \ldots, Z_n]\delta, \ldots, [Z_1, \ldots, C_n]\delta}$$

$$[\,\textbf{substitution}\,]\ \frac{E \vdash t = t' \text{ if } C}{E \vdash [t_1/Z_1, \ldots, t_n/Z_n]t = [t_1/Z_1, \ldots, t_n/Z_n]t' \text{ if } C}$$

$$[\,\textbf{contradiction}\,]\ \frac{E \vdash t = t' \text{ if } C}{E \vdash l = r \text{ if } C}$$

for $t, t' \in T_\Delta[\mathcal{C}]_s,\ covar(t) \cap covar(t') = \emptyset,\ l, r \in T_\Delta[\mathcal{C}']_s$

$$[\,\textbf{case}\,]\ \frac{E \vdash t = t' \text{ if } C, (t_0, \mathcal{C}_1)\ , \ldots,\ E \vdash t = t' \text{ if } C, (t_0, \mathcal{C}_n)}{E \vdash t = t' \text{ if } C}$$

for $t, t' \in T_\Delta[\mathcal{C}']_s,\ t_0 \in T_\Delta[\mathcal{C}]_s,\ \mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$

Letting $\vdash_\Delta \subseteq \mathcal{P}(\mathsf{Coeqn}(\Delta)) \times \mathsf{Coeqn}(\Delta)$ be given by: $E \vdash_\Delta e$ if and only if $E \vdash e$ can be inferred

using a finite number of applications of the above rules, for $E \subseteq \text{Coeqn}(\Delta)$, $e \in \text{Coeqn}(\Delta)$ and $\Delta \in |\text{Cosign}|$ yields an entailment system (see Definition 2.2.7).

Proposition 4.1.49. $(\text{Cosign}, \text{Coeqn}, \vdash)$ *is an entailment system.*

Theorem 4.1.50 (Soundness). *Let (Δ, E) denote a coalgebraic specification, and let e denote a Δ-coequation. Then, $E \vdash e$ implies $E \models_\Delta e$.*

Proof. We use induction on the structure of the proof of $E \vdash e$ to show that $E \models_\Delta e$.

If the last rule applied is **base**, then $E \models_\Delta e$ follows from the definition of $A \models_\Delta E$ for a Δ-coalgebra A. If the last rule applied is **weakening**, then $E \models_\Delta t = t'$ if C, C' follows from the fact that if C, C' holds in a state $a \in A_s$ of some Δ-coalgebra A, then C holds in a. If the last rule applied is **cond** or **reflexivity**, then $E \models_\Delta e$ follows by any Δ-coalgebra (and hence any (Δ, E)-coalgebra) satisfying any coequation of form $c(t, \mathcal{C})$ if $(t, \mathcal{C})$, respectively $t = t$. If the last rule applied is one of **symmetry**, **transitivity** or **substitution**, then $E \models_\Delta e$ follows from the induction hypothesis by using standard properties of equality. If the last rule applied is **closure**, then for any (Δ, E)-coalgebra A and any $a \in A_s$ satisfying $[C_1, \ldots, Z_n]\delta, \ldots, [Z_1, \ldots, C_n]\delta$, say $\delta_A(a) \in \iota_{Z_i}(A_{s_i})$ with $i \in \{1, \ldots, n\}$, the satisfaction of $[Z_1, \ldots, C_i, \ldots, Z_n]\delta$ by a implies the satisfaction of C_i by $\delta_A(a)$, which yields $(t_i)_A(\delta_A(a)) = (t_i')_A(\delta_A(a))$ (by the induction hypothesis); that is, $([t_1, \ldots, t_n]\delta)_A(a) = ([t_1', \ldots, t_n']\delta)_A(a)$. If the last rule applied is **contradiction**, then $E \models_\Delta l = r$ if C follows from the fact that for a (Δ, E)-coalgebra A, there are no states $a \in A_s$ satisfying C (as they would then have to satisfy $t_A(a) = t'_A(a)$). Finally, if the last rule applied is **case**, $E \models_\Delta t = t'$ if C follows from one of the conditions $(t_0, \mathcal{C}_1), \ldots, (t_0, \mathcal{C}_n)$ holding in any state $a \in A_s$ satisfying C, for any (Δ, E)-coalgebra A. □

The completeness proof requires some preliminary results.

Lemma 4.1.51. *Let Δ denote a many-sorted cosignature, and let E denote a set of Δ-coequations. If $E \vdash l = r$ if $C, (t, \mathcal{C})$ and $E \vdash c(t, \mathcal{C})$ if C, C', then $E \vdash l = r$ if C, C'.*

Proof (sketch). If $\mathcal{C}' = covar(t) \setminus \mathcal{C}$, then the soundness of the **weakening** and **contradiction** rules gives:

$$E \vdash l = r \text{ if } C, C', (t, \mathcal{C})$$

and:

$$E \vdash l = r \text{ if } C, C', (t, \mathcal{C}')$$

The conclusion then follows by the soundness of the **case** rule. □

The next two lemmas will prove crucial to the completeness proof. The former states that whenever a set of coequations is inconsistent w.r.t a given sort and a set of conditions for that sort, a contradiction for the given conditions can be syntactically derived from the coequations, while the latter states that if two coterms constrained to the same covariable can not be proved equal under certain conditions, then the two coterms are distinguished by a state satisfying the given conditions, in a coalgebra satisfying the specification.

Lemma 4.1.52. *Let (Δ, E) denote a coalgebraic specification, and let F_E denote a final (Δ, E)-coalgebra. Also, let $s \in S$, and let C denote some conditions for the sort s. If $E \not\vdash l = r$ if C for any $l, r \in T_\Delta[\mathcal{C}]_s$ with $covar(l) \cap covar(r) = \emptyset$, then $F^C_{E,s} = \{\varphi \in F_{E,s} \mid C \text{ holds in } \varphi\} \neq \emptyset$.*

Proof. We define an ω^{op}-chain in Set whose limit object L has the following properties:

(a) $L \neq \emptyset$ implies $F^C_{E,s} \neq \emptyset$

(b) if $L = \emptyset$ then $E \vdash l = r$ if C with $l, r \in T_\Delta[\mathcal{C}]_s$, $covar(l) \cap covar(r) = \emptyset$.

Then, $F^C_{E,s} = \emptyset$ gives $L = \emptyset$, which, in turn, gives $E \vdash l = r$ if C for some $l, r \in T_\Delta[\mathcal{C}]_s$ with $covar(l) \cap covar(r) = \emptyset$, yielding a contradiction. Hence, $F^C_{E,s} \neq \emptyset$.

We begin by recalling that the set $T^1_{\Delta,s}$ is enumerable (see Remark 4.1.6); say $T^1_{\Delta,s} = \{t_1, t_2, \ldots\}$. We then consider the following ω^{op}-chain:

$$C_1 \xleftarrow{p_1} C_2 \xleftarrow{p_2} C_3 \xleftarrow{p_3} \cdots$$

where:

$$\begin{aligned} C_n = \{\ & (Z_{t_1}, \ldots, Z_{t_n}) \mid Z_{t_i} \in covar(t_i) \text{ for } i \in \{1, \ldots, n\}, \\ & E \not\vdash l = r \text{ if } C, (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n}) \\ & \text{for any } l, r \in T_\Delta[\mathcal{C}]_s \text{ with } covar(l) \cap covar(r) = \emptyset\ \} \end{aligned}$$

and:

$$p_n(Z_{t_1}, \ldots, Z_{t_{n+1}}) = (Z_{t_1}, \ldots, Z_{t_n})$$

for $n = 1, 2, \ldots$. A limit object L for this ω^{op}-chain is given by:

$$\begin{aligned} L = \{\ & (Z_{t_i})_{i \in \{1,2,\ldots\}} \mid Z_{t_i} \in covar(t_i) \text{ for } i \in \{1, 2, \ldots\}, \\ & E \not\vdash l = r \text{ if } C, (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n}) \\ & \text{for any } l, r \in T_\Delta[\mathcal{C}]_s \text{ with } covar(l) \cap covar(r) = \emptyset \text{ and any } n\ \} \end{aligned}$$

To show (a), let $(Z_{t_i})_{i \in \{1,2,\ldots\}} \in L$, and let $\varphi : T^1_{\Delta,s} \to \cup\{\, covar(t) \mid t \in T^1_{\Delta,s}\,\}$ be given by $\varphi(t_i) = Z_{t_i}$ for $i = 1, 2, \ldots$.

To show that $\varphi \in F_s$, let $t_i, t_j \in T^1_{\Delta,s}$ be such that $t_j = [t'_1/Z_1, \ldots, t'_n/Z_n]t_i$. If $Z_{t_i} = Z_k$, we must show that $Z_{t_j} \in covar(t'_k)$. Suppose $Z_{t_j} \notin covar(t'_k)$. Then:

$$E \vdash t_j = [t'_1/Z_1, \ldots, t'_n/Z_n]t_i$$

(following by **reflexivity**) together with:

$$E \vdash t_i = [U_1/Z_1, \ldots, Z_k/Z_k, \ldots, U_n/Z_n]t_i \text{ if } C, (t_1, Z_{t_1}), \ldots, (t_i, Z_{t_i})$$

and:

$$E \vdash t_j = [V_1/Z'_1, \ldots, Z_{t_j}/Z_{t_j}, \ldots, V_m/Z'_m]t_j \text{ if } C, (t_1, Z_{t_1}), \ldots, (t_j, Z_{t_j})$$

(both following by **cond** and **weakening**) yield (by **substitution** followed by **weakening** and then by **transitivity**):

$$E \vdash [V_1/Z'_1, \ldots, Z_{t_j}/Z_{t_j}, \ldots, V_m/Z'_m]t_j = [U_1/Z_1, \ldots, t'_k/Z_k, \ldots, U_n/Z_n]t_i$$
$$\text{if } C, (t_1, Z_{t_1}), \ldots, (t_N, Z_{t_N})$$

with $N = max(i, j)$. But this contradicts the definition of L, as the lhs and rhs of the last coequation have no covariable in common. Hence, $Z_{t_j} \in covar(t'_k)$.

To show that $\varphi \in F_{E,s}$, let $t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_s$, $i \in \{1, \ldots, n\}$ and $(t_i = t'_i \text{ if } C_i) \in E$ be such that $t_F(\varphi) \in \iota_{Z_i}(F_{s_i})$, $t_i, t'_i \in T_\Delta[\mathcal{C}_i]_{s_i}$ and C_i holds in $t_F(\varphi)$. According to Proposition 4.1.36, we must show that $(t_i)_F(t_F(\varphi)) = (t'_i)_F(t_F(\varphi))$. However, by Corollary 4.1.21, it suffices to show that, for any coterms $u_1, \ldots, u_q$ of suitable sort, $l_F(\varphi)$ and $r_F(\varphi)$ are both in $\iota_Z(F_{s'})$ for some $Z : s'$, where:

$$l = [u_1/U_1, \ldots, u_q/U_q][Z_1/Z_1, \ldots, t_i/Z_i, \ldots, Z_n/Z_n]t$$
$$r = [u_1/U_1, \ldots, u_q/U_q][Z_1/Z_1, \ldots, t'_i/Z_i, \ldots, Z_n/Z_n]t$$

with $\{U_1, \ldots, U_q\} = \mathcal{C}_i \cup \{Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n\}$.

Let $l = [V_{i_1}/X_1, \ldots, V_{i_m}/X_m]\underline{l}$ with $\underline{l} \in T^1_\Delta[\{X_1, \ldots, X_m\}]_s$, and $r = [V_{j_1}/Y_1, \ldots, V_{j_p}/Y_p]\underline{r}$ with $\underline{r} \in T^1_\Delta[\{Y_1, \ldots, Y_p\}]_s$. From:

$$E \vdash t_i = t'_i \text{ if } C_i$$

(following by **base**) we can infer, by successive applications of the **closure** rule, followed by **substitution**:

$$E \vdash l = r \text{ if } [Z_1/Z_1, \ldots, C_i/Z_i, \ldots, Z_n/Z_n]t$$

We claim that if $Z_{\underline{l}} = X_k$ and $Z_{\underline{r}} = Y_l$, then $V_{i_k} = V_{j_l}$. For, if this was not the case, **cond** together with **substitution** would yield:

$$E \vdash [S_1/V_1, \ldots, V_{i_k}/V_{i_k}, \ldots, S_o/V_o]l = [T_1/V_1, \ldots, V_{j_l}/V_{j_l}, \ldots, T_o/V_o]r$$
$$\text{if } [Z_1/Z_1, \ldots, C_i/Z_i, \ldots, Z_n/Z_n]t, (\underline{l}, X_k), (\underline{r}, Y_l)$$

with $V_{i_k} \neq V_{j_l}$, which would then yield:

$$E \vdash [S_1/V_1, \ldots, V_{i_k}/V_{i_k}, \ldots, S_o/V_o]l = [T_1/V_1, \ldots, V_{j_l}/V_{j_l}, \ldots, T_o/V_o]r$$
$$\text{if } (t_1, Z_{t_1}), \ldots, (t_N, Z_{t_N})$$

for N sufficiently large (the fact that $[Z_1/Z_1, \ldots, C_i/Z_i, \ldots, Z_n/Z_n]t$ holds in φ together with Lemma 4.1.51 are used here). But this would contradict the definition of L. Hence, $V_{i_k} = V_{j_l} = Z : s'$ and $l_F(\varphi), r_F(\varphi) \in \iota_Z(F_{s'})$. This gives $\varphi \in F_{E,s}$.

In addition, $\varphi \in F^C_{E,s}$. For, if this was not the case, the conditions in C would contradict the conditions $(t_1, Z_{t_1}), \ldots, (t_N, Z_{t_N})$ for N sufficiently large (by Lemma 4.1.51), yielding $E \vdash l = r$ if $C, (t_1, Z_{t_1}), \ldots, (t_N, Z_{t_N})$ with $covar(l) \cap covar(r) = \emptyset$. This concludes the proof of (a).

To show (b), assume $L = \emptyset$. Then, for any $Z \in C_1$, there exists $n_Z \in \{2, \ldots\}$ such that $Z \notin \mathsf{Im}(p_1 \circ \ldots \circ p_{n_Z})$. For, if $Z \in C_1$ was such that $Z \in \mathsf{Im}(p_1 \circ \ldots \circ p_n)$ for any $n \in \{2, \ldots\}$, then also $Z \in \mathsf{Im}(l_1)$ (with $l_1 : L \to C_1$ denoting the corresponding arrow of the limiting cone), which

would contradict the assumption that $L = \emptyset$. Now let $n' = max\{n_Z \mid Z \in C_1\}$. It then follows by **weakening** and **contradiction** that $E \vdash l = r$ if $C, (t_1, Z_1), \ldots, (t_{n'}, Z_{n'})$ for any choice of $Z_1 \in covar(t_1), \ldots, Z_{n'} \in covar(t_{n'})$, with $l, r \in T_\Delta[\mathcal{C}]_s$ being such that $covar(l) \cap covar(r) = \emptyset$. Then, successive applications of the **case** rule yield $E \vdash l = r$ if C with $l, r \in T_\Delta[\mathcal{C}]_s$ being such that $covar(l) \cap covar(r) = \emptyset$, thus contradicting the hypothesis. This concludes the proof of (b). □

Lemma 4.1.53. *Let (Δ, E) denote a coalgebraic specification, let $l, r \in T_\Delta[\mathcal{C}]_s$ for some set $\mathcal{C}$ of covariables and some $s \in S$, and let $Z \in \mathcal{C}$. Also, let A_E denote a cofree (Δ, E)-coalgebra over the S-sorted set $(\{*, *'\})_{s \in S}$. If $E \not\vdash l = r$ if $C, (l, Z), (r, Z)$, then there exists $\varphi \in A_{E,s}^{C,(l,Z),(r,Z)}$ such that $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$.*

Proof. We begin by recalling that the cofree (Δ, E)-coalgebra A_E over the S-sorted set $(\{*, *'\})_{s \in S}$ has elements given by functions $\varphi : T^1_{\Delta,s} \to \cup\{\, covar(t) \times (\{*, *'\})_{s \in S} \mid t \in T^1_{\Delta,s} \,\}$, additionally satisfying:

1. $t \in T^1_{\Delta,s}$ implies $\pi_1(\varphi(t)) \in covar(t)$;
2. $t, t' \in T^1_{\Delta,s}$, $t' = [t_1/Z_1, \ldots, t_n/Z_n]t$ and $\pi_1(\varphi(t)) = Z_k$ imply $\pi_1(\varphi(t')) \in covar(t_k)$ and moreover, $\pi_2(\varphi(t')) = \pi_2(\varphi(t))$ if t_k is a covariable;
3. $(t_i)_A(t_A(\varphi)) = (t'_i)_A(t_A(\varphi))$ holds whenever $\varphi \in A_s$, $t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_s$, $i \in \{1, \ldots, n\}$ and $(t_i = t'_i \text{ if } C_i) \in E$ are such that $t_A(\varphi) \in \iota_{Z_i}(A_{s_i})$, $t_i, t'_i \in T_\Delta[\mathcal{C}_i]_{s_i}$ and C_i holds in $t_A(\varphi)$, with A denoting the cofree Δ-coalgebra over $(\{*, *'\})_{s \in S}$.

(This is a consequence of Remark 4.1.18 together with Corollary 4.1.44, see also Proposition 4.1.36.)

The proof is similar to that of Lemma 4.1.52. We define an ω^{op}-chain in Set whose limit object is a non-empty set provided that $E \not\vdash l = r$ if $C, (l, Z), (r, Z)$, and then use an element of the limit object to construct $\varphi \in A_{E,s}^{C,(l,Z),(r,Z)}$ with $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$.

Consider the following ω^{op}-chain:

$$S_1 \xleftarrow{p_1} S_2 \xleftarrow{p_2} S_3 \xleftarrow{p_3} \cdots$$

where:

$$\begin{aligned} S_n = \{\ (Z_{t_1}, \ldots, Z_{t_n}) \mid\ & Z_{t_i} \in covar(t_i) \text{ for } i \in \{1, \ldots, n\}, \\ & E \not\vdash l = r \text{ if } C, (l, Z), (r, Z), (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n}) \ \} \end{aligned}$$

and:

$$p_n(Z_{t_1}, \ldots, Z_{t_{n+1}}) = (Z_{t_1}, \ldots, Z_{t_n})$$

for $n = 1, 2, \ldots$. A limit object L for this ω^{op}-chain is given by:

$$\begin{aligned} L = \{\ (Z_{t_i})_{i \in \{1,2,\ldots\}} \mid\ & Z_{t_i} \in covar(t_i) \text{ for } i \in \{1, 2, \ldots\}, \\ & E \not\vdash l = r \text{ if } C, (l, Z), (r, Z), (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n}) \text{ for any } n \ \} \end{aligned}$$

We claim that:

(a) $S_n \neq \emptyset$ for any $n \in \{1, 2, \ldots\}$

(b) $L \neq \emptyset$

To show (a), assume $S_n = \emptyset$ for some $n \in \{1, 2, \ldots\}$. That is:

$$E \vdash l = r \text{ if } C, (l, Z), (r, Z), (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n})$$

for any $Z_{t_i} \in covar(t_i)$ with $i = 1, \ldots, n$. It then follows by **case** that:

$$E \vdash l = r \text{ if } C, (l, Z), (r, Z)$$

But this contradicts the hypothesis. Therefore $S_n \neq \emptyset$ for any $n \in \{1, 2, \ldots\}$.

To show (b), assume $L = \emptyset$. Hence, for any $Z \in S_1$, there exists $n_Z \in \{2, \ldots\}$ such that $Z \notin \mathsf{Im}(p_1 \circ \ldots \circ p_n)$. If $n' = max\{n_Z \mid Z \in S_1\}$, it follows by **weakening** that:

$$E \vdash l = r \text{ if } C, (l, Z), (r, Z), (t_1, Z_{t_1}), \ldots, (t_{n'}, Z_{t_{n'}})$$

for any choice of $Z_{t_1} \in covar(t_1), \ldots, Z_{t_{n'}} \in covar(t_{n'})$. Hence, by **case**:

$$E \vdash l = r \text{ if } C, (l, Z), (r, Z)$$

Again, this contradicts the hypothesis. Hence, $L \neq \emptyset$.

We now fix $(Z_{t_i})_{i=1,2,\ldots} \in L$, and use it to define $\varphi \in A_{E,s}^{C,(l,Z),(r,Z)}$ such that $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$. Say $\mathcal{C} = \{Z_1, \ldots, Z_n\}$. Then let $l = [Z_{i_1}/X_1, \ldots, Z_{i_m}/X_m]\underline{l}$ with $\underline{l} \in T_\Delta^1[\{X_1, \ldots, X_m\}]_s$, and $r = [Z_{j_1}/Y_1, \ldots, Z_{j_p}/Y_p]\underline{r}$ with $\underline{r} \in T_\Delta^1[\{Y_1, \ldots, Y_p\}]_s$. Also, let $k \in \{1, \ldots, m\}$ be such that $X_k = Z_{\underline{l}}$, and $l \in \{1, \ldots, p\}$ be such that $Y_l = Z_{\underline{r}}$. One can immediately infer that $Z_{i_k} = Z$ and $Z_{j_l} = Z$; for if, say, $Z_{i_k} \neq Z$, then the conditions (l, Z) and $(\underline{l}, X_k)$ would contradict each other, yielding:

$$E \vdash l' = r' \text{ if } C, (l, Z), (r, Z), (t_1, Z_{t_1}), \ldots, (t_N, Z_{t_N})$$

with $covar(l') \cap covar(r') = \emptyset$ for N sufficiently large. Finally, for $n \in \{1, 2, \ldots\}$, let C_n stand for $(t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n})$.

Now define:

$$\begin{aligned} T = \{\ t \in T_{\Delta,s}^1 \mid\ & \text{there exists } n \in \{1, 2, \ldots\} \text{ such that} \\ & E \vdash t = [Y_1/Y_1, \ldots, Z_t/Y_l, \ldots, Y_p/Y_p]\underline{r} \text{ if } C, (l, Z), (r, Z), C_n\ \} \end{aligned}$$

where $\{Y_1, \ldots, Y_p\} \cap covar(t) = \emptyset$ for any $t \in T_{\Delta,s}^1$. That is, T consists of Δ-coterms whose interpretation must agree with that of $\underline{r}$ on any state in $A_{E,s}^{C,(l,Z),(r,Z)}$ which, in addition, satisfies the conditions (t_i, Z_i), with $i = 1, 2, \ldots$. Then let $\varphi \in A_{E,s}^{C,(l,Z),(r,Z)}$ be given by: $\varphi(t) = \langle Z_t, c_t \rangle$ for $t \in T_{\Delta,s}^1$, where:

$$c_t = \begin{cases} * & \text{if } \ t \notin T \\ *' & \text{if } \ t \in T \end{cases}$$

Note that if, say, $Z : s'$, then $t \in T$ gives $Z_t : s'$ (as $Z_{\underline{r}} = Y_l : s'$).

We now claim that:

(c) $\varphi \in A_{E,s}^{C,(l,Z),(r,Z)}$

(d) $r_{A_E}(\varphi) \neq l_{A_E}(\varphi)$

Proving (c) amounts to proving that $\varphi \in A_{E,s}$, and that $C, (l, Z), (r, Z)$ holds in φ.

The proof of $\varphi \in A_s$, with A denoting the cofree Δ-coalgebra over $(\{*, *'\}_s)_{s \in S}$ is similar to the proof of $\varphi \in F_s$ in Lemma 4.1.52. In addition, here we must show that if $t_i, t_j \in T^1_{\Delta,s}$ are such that $t_j = [t'_1/Z'_1, \ldots, t'_n/Z'_n]t_i$, $Z_{t_i} = Z'_k$ and $t'_k = Z_{t_j}$, then either t_i and t_j are both in T, or none of them is in T. One can distinguish two cases:

1. $t_i \in T$. That is:

   $$E \vdash t_i = [Y_1/Y_1, \ldots, Z'_k/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

   for some $n_0 \in \{1, 2, \ldots\}$. Then, **substitution** yields:

   $$E \vdash t_j = [t'_1/Z'_1, \ldots, t'_n/Z'_n][Y_1/Y_1, \ldots, Z'_k/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

   This, together with:

   $$\{Z'_1, \ldots, Z'_{k-1}, Z'_{k+1}, \ldots, Z'_n\} \cap \{Y_1, \ldots, Y_{l-1}, Y_{l+1}, \ldots, Y_m\} = \emptyset$$

   and $t'_k = Z_{t_j}$ yields:

   $$E \vdash t_j = [Y_1/Y_1, \ldots, Z_{t_j}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

   That is, $t_j \in T$.

2. $t_j \in T$. That is:

   $$E \vdash t_j = [Y_1/Y_1, \ldots, Z_{t_j}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

   for some $n_0 \in \{1, 2, \ldots\}$. Then, $t_j = [t'_1/Z'_1, \ldots, t'_n/Z'_n]t_i$ gives:

   $$E \vdash [t'_1/Z'_1, \ldots, t'_n/Z'_n]t_i = [Y_1/Y_1, \ldots, Z_{t_j}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

   But $Z_{t_i} = Z'_k$ together with **substitution** and **cond** yield:

   $$E \vdash [t'_1/Z'_1, \ldots, t'_n/Z'_n]t_i = [Z'_1/Z'_1, \ldots, t'_k/Z'_k, \ldots, Z'_n/Z'_n]t_i \text{ if } C, (l, Z), (r, Z), C_N$$

   for N sufficiently large. Also, $t'_k = Z_{t_j}$. Hence, by **transitivity**:

   $$E \vdash [Z'_1/Z'_1, \ldots, Z_{t_j}/Z'_k, \ldots, Z'_n/Z'_n]t_i = [Y_1/Y_1, \ldots, Z_{t_j}/Y_l, \ldots, Y_m/Y_m]\underline{r}$$
   $$\text{if } C, (l, Z), (r, Z), C_N$$

   Finally, substituting Z_{t_i} for Z_{t_j} yields:

   $$E \vdash t_i = [Y_1/Y_1, \ldots, Z_{t_i}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_N$$

   That is, $t_i \in T$.

Hence, either both t_i and t_j belong to T, or neither of them does, and therefore $c_{t_i} = c_{t_j}$. This concludes the proof of $\varphi \in A_s$.

The proof of $\varphi \in A_{E,s}$ is, again, similar to the proof of $\varphi \in F_{E,s}$ in Lemma 4.1.52. In addition, here we must show that given $t \in T^1_\Delta[\{Z'_1, \ldots, Z'_n\}]_s$, $i \in \{1, \ldots, n\}$ and $(t_i = t'_i \text{ if } C_i) \in E$ such that $t_A(\varphi) \in \iota_{Z'_i}(A_{s_i})$, $t_i, t'_i \in T_\Delta[\mathcal{C}_i]_{s_i}$ and C_i holds in $t_A(\varphi)$, then either both $\underline{l}'$ and $\underline{r}'$ are in T, or none of them is (where l' and r' are defined similarly to l and r from Lemma 4.1.52).

Suppose $\underline{l}' \in T$. On the one hand,

$$E \vdash l' = r' \text{ if } [Z'_1/Z'_1, \ldots, C_i/Z'_i, \ldots, Z'_n/Z'_n]t$$

(following by successive applications of the **closure** rule) together with the fact that the condition $[Z'_1/Z'_1, \ldots, C_i/Z'_i, \ldots, Z'_n/Z'_n]t$ holds in φ yield:

$$E \vdash l' = r' \text{ if } C_N$$

for N sufficiently large (Lemma 4.1.51 is used here). That is:

$$E \vdash [W_{i_1}/U_1, \ldots, W_{i_q}/U_q]\underline{l}' = [W_{j_1}/V_1, \ldots, W_{j_r}/V_r]\underline{r}' \text{ if } C_N$$

One can immediately infer that if $q_0 \in \{1, \ldots, q\}$ and $r_0 \in \{1, \ldots, r\}$ are defined by $Z_{\underline{l}'} = U_{q_0}$ and respectively $Z_{\underline{r}'} = V_{r_0}$, then $W_{i_{q_0}} = W_{j_{r_0}}$.

On the other hand, $\underline{l}' \in T$ gives:

$$E \vdash \underline{l}' = [Y_1/Y_1, \ldots, Z_{l'}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

for some $n_0 \in \{1, 2, \ldots\}$.

The last two statements, together with:

$$E \vdash \underline{l}' = [W_{i_1}/U_1, \ldots, U_{q_0}/U_{q_0}, \ldots, W_{i_q}/U_q]\underline{l}' \text{ if } C_N$$

and:

$$E \vdash \underline{r}' = [W_{j_1}/V_1, \ldots, V_{r_0}/V_{r_0}, \ldots, W_{j_r}/V_r]\underline{r}' \text{ if } C_N$$

(both following by **cond** for N sufficiently large) can then be used to infer:

$$E \vdash \underline{r}' = [Y_1/Y_1, \ldots, Z_{r'}/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_N$$

That is, $\underline{r}' \in T$. This concludes the proof of $\varphi \in A_{E,s}$.

It remains to prove that $C, (l, Z), (r, Z)$ holds in φ. If this was not the case, the condition $C, (l, Z), (r, Z), C_N$ would be contradictory for N sufficiently large. This, in turn, would yield $E \vdash l = r$ if $C, (l, Z), (r, Z), C_N$, contradicting the definition of L.

We have therefore proved (c). To prove (d), it suffices to show that $\underline{l} \notin T$. Then, since $\underline{r} \in T$, the claim follows from $* \neq *'$. We show that $\underline{l} \in T$ yields a contradiction. If $\underline{l} \in T$, then:

$$E \vdash \underline{l} = [Y_1/Y_1, \ldots, X_k/Y_l, \ldots, Y_m/Y_m]\underline{r} \text{ if } C, (l, Z), (r, Z), C_{n_0}$$

for some $n_0 \in \{1, 2, \ldots\}$. This, together with:

$$E \vdash l = [X_1/X_1, \ldots, Z/X_k, \ldots, X_m/X_m]\underline{l} \text{ if } C_N$$

and:

$$E \vdash r = [Y_1/Y_1, \ldots, Z/Y_l, \ldots, Y_p/Y_p]\underline{r} \text{ if } C_N$$

for N sufficiently large (both following by **transitivity** and **cond**) can then be used to infer:

$$E \vdash l = r \text{ if } C, (l, Z), (r, Z), C_N$$

for N sufficiently large. But this contradicts the fact that $(Z_{t_i})_{i=1,2,\ldots} \in L$. Hence, $\underline{l} \notin T$. This concludes the proof of (d).

We have therefore constructed $\varphi \in A_{E,s}$ such that $C, (l, Z), (r, Z)$ holds in φ, but $r_{A_E}(\varphi) \neq l_{A_E}(\varphi)$. This concludes the proof. □

Theorem 4.1.54 (Completeness). *Let (Δ, E) denote a coalgebraic specification, and let e denote a Δ-coequation. Then, $E \models_\Delta e$ implies $E \vdash e$.*

Proof. Let e be of form $l = r$ if C with $l, r \in T_\Delta[\mathcal{C}]_s$, let F_E denote a final (Δ, E)-coalgebra, and let A_E denote a cofree (Δ, E)-coalgebra over the S-sorted set $(\{*, *'\})_{s \in S}$. We distinguish the following cases.

1. $F^C_{E,s} = \emptyset$.

   Then, $E \vdash e$ follows immediately by Lemma 4.1.52.

2. $F^C_{E,s} \neq \emptyset$.

   We assume that $E \nvdash e$, and show that this yields a contradiction. From $E \nvdash e$ one can immediately infer that there exist $Z \in \mathcal{C}_{s'}$ and $Z' \in \mathcal{C}_{s''}$ with $s', s'' \in S$ such that $E \nvdash l = r$ if $C, (l, Z), (r, Z')$ (otherwise $E \vdash l = r$ if C would follow by **case**). We now distinguish two subcases.

   (a) $Z \neq Z'$.

   We have: $E \nvdash l' = r'$ if $C, (l, Z), (r, Z')$ for any $l', r' \in T_\Delta[\mathcal{C}']_s$ with $covar(l') \cap covar(r') = \emptyset$ (otherwise **contradiction** could be applied to infer that $E \vdash l = r$ if $C, (l, Z), (r, Z')$). Lemma 4.1.52 then gives $\varphi \in F_{E,s}$ such that $C, (l, Z), (r, Z')$ holds in φ. That is, φ satisfies the conditions C but $l_{F_E}(\varphi) \neq r_{F_E}(\varphi)$ (as $l_{F_E}(\varphi) \in \iota_Z(F_{E,s'})$ and $r_{F_E}(\varphi) \in \iota_{Z'}(F_{E,s''})$, with $Z \neq Z'$). Hence, $F_E \not\models_\Delta l = r$ if C.

   (b) $Z = Z'$.

   Since $E \nvdash l = r$ if $C, (l, Z), (r, Z)$, it follows by Lemma 4.1.53 that there exists $\varphi \in A_{E,s}$ such that $C, (l, Z), (r, Z)$ holds in φ but $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$. Hence, $A_E \not\models_\Delta l = r$ if C.

   In both of the above subcases one can infer that $E \not\models_\Delta e$, which contradicts the hypothesis. Hence, $E \vdash e$.

This concludes the proof of completeness. □

Example 4.1.55. Given the cosignature in Example 4.1.2, the fact that:

$$E \vdash \texttt{[[F,N]empty?]tail = [[F',N]empty?]tail}$$

(capturing the fact that applying `tail` to a non-empty list always yields a non-empty list) with $E = \{\ \texttt{[[F,[E]head]empty?]tail = [E]head}\ \}$ (requiring non-empty lists to have a second element, and this element to be equal to their first element) follows by **case-analysis** from:

$$E \vdash \texttt{[[F,N]empty?]tail = [[F',N]empty?]tailif ([[F,N]empty?]tail,N)}$$

following directly by **cond**, together with:

$$E \vdash \texttt{[[F,N]empty?]tail = [[F',N]empty?]tailif ([[F,N]empty?]tail,F)}$$

following by **contradiction** from:

$$E \vdash \texttt{[E]head = [E']head if ([[F,N]empty?]tail,F)}$$

The previous statement follows by **transitivity** from:

$$E \vdash \texttt{[[F,[E]head]empty?]tail = [[F,N']empty?]tailif ([[F,N]empty?]tail,F)}$$

and:

$$E \vdash \texttt{[[F,[E']head]empty?]tail = [[F,N']empty?]tailif ([[F,N]empty?]tail,F)}$$

(both following by **cond** followed by **substitution**), together with:

$$E \vdash \texttt{[[F,[E]head]empty?]tail = [E]headif ([[F,N]empty?]tail,F)}$$

and:

$$E \vdash \texttt{[[F,[E']head]empty?]tail = [E']headif ([[F,N]empty?]tail,F)}$$

(both following by **base** followed by **weakening**).

Example 4.1.56. Given the cosignature in Example 4.1.29, the fact that:

$$E \vdash \texttt{[Z,L]rest = [Z',L]rest}$$

with:

$$E = \{\ \texttt{[Z,[Z,E]first]rest = [Z',E]first}\ \}$$

follows by **case-analysis** from:

$$E \vdash \texttt{[Z,L]rest = [Z',L]rest if ([Z,L]rest,L)}$$

following directly by **cond**, together with:

$$E \vdash \texttt{[Z,L]rest = [Z',L]rest if ([Z,L]rest,Z)}$$

following by **contradiction** from:

$$E \vdash \texttt{[Z,[Z,E]first]rest = [Z',E']first if ([Z,L]rest,Z)}$$

The last statement follows by **transitivity** from:

$$E \vdash \texttt{[Z,[Z,E]first]rest = [Z,[Z,E']first]rest if ([Z,L]rest,Z)}$$

following by **cond** and **substitution**, together with:

$$E \vdash \texttt{[Z,[Z,E']first]rest = [Z',E']first if ([Z,L]rest,Z)}$$

following by **base** and **weakening**.

We conclude this section by noting that a sound and complete deduction calculus could have also been formulated for the alternative notion of coequation described in Remark 4.1.27.

4.2 Coalgebraic Specification over a Fixed Data Universe

The fact that the components of arbitrary polynomial endofunctors on Set^S can be written as coproducts of finite products of projection functors[10] results in many-sorted signatures being at least as expressive as polynomial endofunctors from the point of view of the structures specifiable with them[11]. However, a similar result can not be stated about many-sorted cosignatures, as the components of polynomial endofunctors on Set^S can not, in general, be written as products of finite coproducts of projection functors[12].

An expressiveness equal to that of arbitrary polynomial endofunctors can, however, be achieved by allowing a fixed data universe to be used for specification. For, this yields a notion of cosignature which is able to capture extended polynomial endofunctors whose components have the form of products of finite coproducts of constant/projection functors. Also, by allowing a change in the underlying category, arbitrary polynomial endofunctors can be transformed into endofunctors of the above-mentioned form. As a result, the new notion of cosignature is as expressive as arbitrary polynomial endofunctors from the point of view of the structures specifiable with it.

This section extends the approach in the previous section in order to achieve this expressiveness. The extension assumes the existence of a fixed data universe, in the form of a set V of *visible sorts*, and of a V-sorted set D of *data values*. (The particular choice of V and D is determined by the polynomial endofunctor under consideration.) Furthermore, it is assumed that $D_v \neq \emptyset$ for each $v \in V$.

4.2.1 Destructor Cosignatures and Cosignature Morphisms

Definition 4.2.1. *A* **destructor cosignature over** V *is a pair* (H, Δ) *with* H *a set of* **hidden sorts** *and* Δ *a* $V \cup H$*-sorted cosignature such that* $\Delta_v = \emptyset$ *for each* $v \in V$*.*

A **cosignature morphism** *between destructor cosignatures* (H, Δ) *and* (H', Δ') *over* V *is a many-sorted cosignature morphism* $\phi : (V \cup H, \Delta) \to (V \cup H', \Delta')$ *such that* $\phi|_V = \iota_V : V \to V \cup H'$ *and such that* $\phi(H) \subseteq H'$*.*

The category of destructor cosignatures over V and destructor cosignature morphisms is denoted

[10] This is a consequence of the existence of a Set-theoretic isomorphism $C \simeq \coprod_{c \in C} 1$ with C an arbitrary set and 1 a one-element set (defining an empty product) on the one hand, and of the distributivity of products over coproducts in Set on the other.

[11] Many-sorted signatures are actually more expressive than polynomial endofunctors, as they also cover endofunctors whose components have the form of *infinite* coproducts of finite products of projection functors.

[12] This is due to the fact that constant functors can not be written as products of finite coproducts of projection functors, as well as to the fact that, in Set, coproducts do not distribute over products.

Cosign_V. Whenever possible, destructor cosignatures (H, Δ) over V are abbreviated Δ, while the set $V \cup H$ is denoted S.

Example 4.2.2. The many-sorted cosignature used in Example 4.1.2 to specify finite and infinite lists can be regarded as a destructor cosignature over `1` and `Elt`.

Now recall from Theorem 4.1.45 that finite colimits exist in the category of many-sorted cosignatures and many-sorted cosignature morphisms. To prove a similar result for destructor cosignatures and their morphisms, it suffices to show that finite colimits in Cosign preserve destructor cosignatures and their morphisms. But this follows immediately from the construction of finite colimits in Cosign given in the proof of Theorem 4.1.45. Hence, the following holds.

Theorem 4.2.3. Cosign_V *is finitely cocomplete.*

4.2.2 Coalgebras, Finality and Bisimilarity

Definition 4.2.4. *Let Δ denote a destructor cosignature over V. A Δ_D-***coalgebra** *is a many-sorted Δ-coalgebra A such that $A_v = D_v$ for each $v \in V$. Also, a Δ_D-***homomorphism** *between Δ_D-coalgebras A and B is a many-sorted Δ-homomorphism $f : A \to B$ such that $f_v = 1_{D_v}$ for each $v \in V$.*

The category of Δ_D-coalgebras and Δ_D-homomorphisms is denoted $\mathsf{Coalg}_D(\Delta)$. Then, destructor cosignature morphisms $\phi : \Delta \to \Delta'$ induce reduct functors $\mathsf{U}_\phi : \mathsf{Coalg}_D(\Delta') \to \mathsf{Coalg}_D(\Delta)$. (The action of U_ϕ on Δ_D-coalgebras and homomorphisms coincides with that of the reduct functor induced by the many-sorted cosignature morphism defining ϕ on the underlying many-sorted Δ-coalgebras and homomorphisms.)

Example 4.2.5. Given the destructor cosignature in Example 4.2.2, fixing the interpretation of the sort `1` to $\{*\}$, and that of the sort `Elt` to $\mathbb{N}$ results in the coalgebras of this cosignature implementing finite and infinite lists of natural numbers.

Remark 4.2.6. Destructor cosignatures and their morphisms are instances of the corresponding abstract concepts. Specifically, if Δ denotes a destructor cosignature over V, and if $\mathsf{G}_\Delta : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ denotes the endofunctor given by:

$$(\mathsf{G}_\Delta X)_s = \begin{cases} D_s & \text{if } \ s \in V \\ \prod\limits_{\delta \in \Delta_{s,s_1 \ldots s_n}} (X_{s_1} + \ldots + X_{s_n}) & \text{if } \ s \in H \end{cases}$$

for $X \in |\mathsf{Set}_D^S|$, then the categories $\mathsf{Coalg}_D(\Delta)$ and $\mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{G}_\Delta)$ are isomorphic. (Δ_D-coalgebras A induce Set_D^S-arrows $\alpha : A \to \mathsf{G}_\Delta A$, with α_h mapping $a \in A_h$ to $(\delta_A(a))_{\delta \in \Delta_{h,s_1 \ldots s_n}}$, for $h \in H$, and conversely, any such Set_D^S-arrow defines a Δ_D-coalgebra structure on its domain; moreover, the two mappings are inverse to each other.) Also, if $\phi : (H, \Delta) \to (H', \Delta')$ denotes a destructor cosignature morphism, and if $\eta_\phi : \mathsf{U}\mathsf{G}_{\Delta'} \Rightarrow \mathsf{G}_\Delta \mathsf{U}$ (with $\mathsf{U} : \mathsf{Set}_D^{S'} \to \mathsf{Set}_D^S$ denoting the

functor induced by the sort component of ϕ) denotes the natural transformation whose components are given by:

$$(\eta_{\phi,X})_v = 1_{D_v},\ v \in V$$
$$(\eta_{\phi,X})_h((x_{\delta'})_{\delta' \in \Delta'_{\phi(h),s'_1 \ldots s'_n}}) = (x_{\phi(\delta)})_{\delta \in \Delta_{h,s_1 \ldots s_m}},\ h \in H$$

for $X \in |\mathsf{Set}^{S'}|$, then U_ϕ agrees with U_{η_ϕ}.

This correspondence immediately results in the existence of final and cofree coalgebras. Such coalgebras can alternatively be obtained as cofree many-sorted coalgebras over suitably-chosen sorted sets.

Proposition 4.2.7. *Let Δ denote a destructor cosignature over V, let $C \in |\mathsf{Set}^S_D|$, and let A denote a cofree many-sorted Δ-coalgebra over C (as given by Remark 4.1.18). Then, A defines, up to isomorphism in* $\mathsf{Coalg}(\Delta)$*, a cofree Δ_D-coalgebra over C.*

Proof. The fact that $\Delta_v = \emptyset$ for $v \in V$ ensures that A defines, up to isomorphism in $\mathsf{Coalg}(\Delta)$, a Δ_D-coalgebra. Cofreeness of A in $\mathsf{Coalg}_D(\Delta)$ then follows from its cofreeness w.r.t. the functor taking many-sorted Δ-coalgebras to their carrier. □

Taking C to be final in Set^S_D yields a final Δ_D-coalgebra.

Corollary 4.2.8. *Let Δ denote a destructor cosignature over V, and let $C \in |\mathsf{Set}^S_D|$ be given by: $C_v = D_v$ for $v \in V$, and $C_h = \{*\}$ for $h \in H$. The carrier of the final Δ_D-coalgebra is given by:*

$$\begin{aligned} F_h = \{\ \varphi : T^1_{\Delta,h} \to \cup\{\, covar(t) \times C \mid t \in T^1_{\Delta,h} \,\} \mid\ & t \in T^1_{\Delta,h} \Rightarrow \pi_1(\varphi(t)) \in covar(t), \\ & t, t' \in T^1_{\Delta,h},\ t' = [t_1/Z_1, \ldots, t_n/Z_n]t,\ \pi_1(\varphi(t)) = Z_k \Rightarrow \pi_1(\varphi(t')) \in covar(t_k) \\ & \text{and moreover, } \pi_2(\varphi(t')) = \pi_2(\varphi(t)) \text{ if } t_k \text{ is a covariable } \},\quad h \in H \\ F_v = D_v,\quad & v \in V \end{aligned}$$

Similarly, cofree coalgebras along destructor cosignature morphisms can alternatively be obtained as cofree many-sorted coalgebras along the underlying many-sorted cosignature morphisms.

Proposition 4.2.9. *Let $\phi : \Delta \to \Delta'$ denote a destructor cosignature morphism, let A denote a Δ_D-coalgebra, and let A' denote a cofree many-sorted Δ'-coalgebra over the many-sorted Δ-coalgebra A w.r.t.* $\mathsf{U}_\phi : \mathsf{Coalg}(\Delta') \to \mathsf{Coalg}(\Delta)$ *(as given by Proposition 4.1.39). Then, A' defines, up to isomorphism in* $\mathsf{Coalg}(\Delta')$*, a cofree Δ'_D-coalgebra over A w.r.t.* $\mathsf{U}_\phi : \mathsf{Coalg}_D(\Delta') \to \mathsf{Coalg}_D(\Delta)$.

Proof. Similar to the proof of Proposition 4.2.7. □

The notion of bisimilarity induced by destructor cosignatures is finer than the one induced by their underlying many-sorted cosignatures – the visible components of bisimilarity relations are equality relations, as opposed to universal relations. As far as the hidden components of bisimilarity relations are concerned, a characterisation similar to the one in Proposition 4.1.19 can be given.

Proposition 4.2.10. *Let Δ denote a destructor cosignature over V, and let A denote a Δ_D-coalgebra. Then, given $h \in H$, two states $a, a' \in A_h$ are bisimilar if and only if for any $t \in T^1_{\Delta,h}$, there exist $s \in S$ and $Z \in covar(t)_s$ such that $t_A(a), t_A(a') \in \iota_Z(A_s)$ and moreover, $t_A(a) = t_A(a')$ if $s \in V$.*

Proof. Similar to the proof of Proposition 4.1.19. □

Corollary 4.2.11. *Let Δ denote a destructor cosignature over V, let F denote a final Δ_D-coalgebra, and let $l, r \in T_\Delta[\{Z_1, \ldots, Z_n\}]_h$ with $Z_1 : s_1, \ldots, Z_n : s_n$ and $h \in H$. Then, for $\varphi \in F_h$, $l_F(\varphi) = r_F(\varphi)$ if and only if for any $t_i \in T^1_{\Delta,s_i}$ for $i = 1, \ldots, n$, $([t_1/Z_1, \ldots, t_n/Z_n]l)_F(\varphi)$ and $([t_1/Z_1, \ldots, t_n/Z_n]r)_F(\varphi)$ are both in $\iota_Z(F_s)$ for some $Z : s$ and $s \in S$ and moreover, $([t_1/Z_1, \ldots, t_n/Z_n]l)_F(\varphi) = ([t_1/Z_1, \ldots, t_n/Z_n]r)_F(\varphi)$ if $s \in V$.*

Proof (sketch). The proof uses Proposition 4.2.10 and is similar to the proof of Corollary 4.1.21. □

Example 4.2.12. The notion of bisimilarity induced by the destructor cosignature given in Example 4.2.2 relates two elements of sort `List` either if both of them denote empty lists (i.e. if the evaluation path for `[Z,N]empty?` corresponds to `Z`), or if both denote non-empty lists (i.e. if the evaluation path for `[Z,N]empty?` corresponds to `N`) and moreover, they have the same number of elements as well as the same elements. (Recall from Example 4.1.20 that the notion of bisimilarity induced by the many-sorted cosignature underlying this destructor cosignature did not distinguish two lists with the same number of elements.)

The algebraic notion of behavioural equivalence captures the indistinguishability of states by observations of visible type (see e.g. Example 3.1.16). A similar characterisation can be given for the notion of bisimilarity associated to destructor cosignatures.

Proposition 4.2.13. *Let Δ denote a destructor cosignature over V, and let A denote a Δ_D-coalgebra. Also, let $V' = V \cup \{1\}$, let D' denote the V'-sorted set given by $D'_v = D_v$ for $v \in V$ and $D'_1 = \{*\}$, and let Δ' denote the destructor cosignature over V' given by $\Delta \cup \{! : h \to 1 \mid h \in H\}$. Then, for $h \in H$ and $a, a' \in A_h$, $a \sim_A a'$ if and only if $t_A(a) = t_A(a')$ for any $t \in T^1_{\Delta',h}$ which contains no hidden-sorted covariables.*

Proof. The conclusion follows from Proposition 4.2.10, after observing that Δ_D-coalgebras are in one-to-one correspondence with $\Delta'_{D'}$-coalgebras, while Δ-coterms t of sort h are in one-to-one correspondence with Δ'-coterms t' of sort h with no hidden-sorted covariables (with t' being obtained by substituting $[Z']!$ for each hidden-sorted covariable Z appearing in t, and with $t_A(a) \in \iota_Z(A_h)$ holding precisely when $t'_A(a) \in \iota_{Z'}(A_1)$). □

That is, two states are bisimilar if and only if they yield, under any experiment which is performed on them, either the same visible result or hidden results of the same type[13].

[13] Note that type information is always observable.

4.2.3 Abstracting Away Bisimilar States

The standard notion of satisfaction of coequations may prove restrictive in cases where one's interest is to only specify system properties up to indistinguishability by observations yielding visible results. In such cases, a notion of satisfaction of coequations up to bisimulation appears to be more appropriate. This section discusses the properties of such a notion of satisfaction.

Definition 4.2.14. *Let Δ denote a destructor cosignature over V. A Δ_D-coalgebra A* **satisfies** *a hidden Δ-coequation e of form $l = r$ if $(t_1, \mathcal{C}'_1), \ldots, (t_n, \mathcal{C}'_n)$* **up to bisimulation** *(written $A \models^{\mathsf{b}}_{\Delta} e$) if and only if, whenever $a \in A_h$ is such that $(t_i)_A(a) \in \iota_{Z_i}(A_{s_i})$ for some $Z_i \in (\mathcal{C}'_i)_{s_i}$, for $i = 1, \ldots, n$, it follows that $l_A(a), r_A(a) \in \iota_Z(A_{s'})$ for some $Z \in covar(l) \cap covar(r)$ with $Z : s'$ and moreover, $l_A(a) \sim_{A,s'} r_A(a)$ (with $\sim_A$ denoting Δ_D-bisimilarity on A).*

Remark 4.2.15. The notion of satisfaction of hidden coequations up to bisimulation by coalgebras of destructor cosignatures is an instance of the corresponding abstract notion (see Definition 3.1.35). For, if (K, l', r') denotes the $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coequation induced by the hidden Δ-coequation e (see Remark 4.2.19), then the requirement that $l_A(a), r_A(a) \in \iota_Z(A_{s'})$ for some $Z : s'$ and that $l_A(a) \sim_{A,s'} r_A(a)$ holds whenever $a \in A_h$ is such that $(t_i)_A(a) \in \iota_{Z_i}(A_{s_i})$ for some $Z_i \in (\mathcal{C}'_i)_{s_i}$, for $i = 1, \ldots, n$ is equivalent to the existence of a Set-arrow c making the following diagram commute:

$$\begin{array}{ccc} A_h & \xrightarrow{\langle l'_{A,h}, r'_{A,h} \rangle} & (1 + \coprod_{s' \in S} \coprod_{Z \in \mathcal{C}_{s'}} A_{s'}) \times (1 + \coprod_{s' \in S} \coprod_{Z \in \mathcal{C}_{s'}} A_{s'}) \\ & \overset{c}{\searrow} & \uparrow \langle 1_1 + (\sum_{s' \in S} \sum_{Z \in \mathcal{C}_{s'}} \pi_{1,s'}), 1_1 + (\sum_{s' \in S} \sum_{Z \in \mathcal{C}_{s'}} \pi_{2,s'}) \rangle \\ & & (1 + \coprod_{s' \in S} \coprod_{Z \in \mathcal{C}_{s'}} \sim_{A,s'}) \end{array}$$

where $l, r \in T_\Delta[\mathcal{C}]_h$, and where $\pi_1, \pi_2 : \sim_A \to A$ denote the projections resulting from $\sim_A \subseteq A \times A$. Also, by Remark 3.1.36, commutativity of the above diagram is equivalent to the satisfaction up to bisimulation of the $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coequation (K, l', r') by the $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coalgebra associated to the Δ_D-coalgebra A.

For coalgebras that are observable (see Definition 3.1.15), the notion of satisfaction of hidden coequations up to bisimulation coincides with the standard notion of satisfaction of hidden coequations. In particular, this results in a final coalgebra of a destructor specification (Δ, E) also defining a final object for the full subcategory of $\mathsf{Coalg}_D(\Delta)$ whose objects satisfy E up to bisimulation.

Versions of the results in Section 4.1.4 (Proposition 4.1.34, Corollary 4.1.35 and Proposition 4.1.36) can also be formulated for the notion of satisfaction of hidden coequations up to bisimulation. Again, in the case of 2 of Proposition 4.1.34, no restriction on the homomorphism f is required.

According to the remarks in Section 3.1.2, proving the satisfaction of hidden coequations up to bisimulation by some class of coalgebras can be reduced to exhibiting a suitable generic bisimulation (see Definition 3.1.44) on that class of coalgebras. This will be illustrated in the following example.

Example 4.2.16. Consider the list specification given in Example 4.1.29. This specification can be further constrained by adding the coequation:

```
[Z,[Z,[Z,E]first]rest]rest = [Z,E]first if ([Z,[Z,L]rest]rest,L)
```

This results in a specification ALT_LIST of alternating lists. For, the coequation:

```
[Z,[Z,L]rest]rest = L if ([Z,[Z,L]rest]rest,L)
```

holds, up to bisimulation, in all coalgebras satisfying the above specification. This can be proved by exhibiting a generic bisimulation $(R_A)_{A\in|\mathsf{Coalg}(\mathtt{ALT\_LIST})|}$ which relates the lhs and rhs of the above coequation whenever the condition of the coequation is satisfied. Specifically, for $A \in |\mathsf{Coalg}(\mathtt{ALT\_LIST})|$, R_A is the least relation on A such that:

$$a\ R_A\ (\mathtt{[Z,[Z,L]rest]rest})_A(a) \text{ whenever } (\mathtt{[Z,[Z,L]rest]rest})_A(a) \in \iota_{\mathtt{L}}(A_{\mathtt{List}})$$

for $a \in A_{\mathtt{List}}$, and such that the visible components of R_A are the equality relations on $\{*\}$ and respectively $\mathbb{N}$. The definition of R_A immediately results in R_A being closed under the application of $\mathtt{rest}_A$. Also, the fact that A satisfies the coequation:

```
[Z,[Z,[Z,E]first]rest]rest = [Z,E]first if ([Z,[Z,L]rest]rest,L)
```

results in R_A also being closed under the application of $\mathtt{first}_A$. Hence, R_A defines a bisimulation on A, whenever $A \in |\mathsf{Coalg}(\mathtt{ALT\_LIST})|$. On the other hand, R_A relates the lhs and rhs of the coequation:

```
[Z,[Z,L]rest]rest = L if ([Z,[Z,L]rest]rest,L)
```

whenever the condition of this coequation is satisfied. It therefore follows that this coequation holds, up to bisimulation, in all coalgebras of the ALT_LIST specification.

We conclude this section with a characterisation of the notion of satisfaction of hidden coequations up to bisimulation in terms of the standard notion of satisfaction of hidden coequations.

Proposition 4.2.17. *Let Δ denote a destructor cosignature over V, and let V' and Δ' be as in Proposition 4.2.13. Also, let A denote a Δ_D-coalgebra, and let e denote a hidden Δ-coequation of form $l = r$ if C. Then, $A \models^{\mathsf{b}}_{\Delta} e$ if and only if $A \models_{\Delta'} [t_1/Z_1, \ldots, t_n/Z_n]l = [t_1/Z_1, \ldots, t_n/Z_n]r$ if C for any suitably-typed coterms $t_1, \ldots, t_n \in T^1_{\Delta'}$ containing no hidden-sorted covariables (where $\{Z_1, \ldots, Z_n\}$ denotes the set of covariables appearing in e).*

Proof. The **if** direction follows from Proposition 4.2.13, while the **only if** direction follows from the soundness of the **substitution** rule for the satisfaction of hidden coequations up to bisimulation (see Theorem 4.2.35), after noting that $A \models^{\mathsf{b}}_{\Delta} e$ if and only if $A \models^{\mathsf{b}}_{\Delta'} e$. (Adding $! : h \to 1$, with $h \in H$, to Δ does not affect Δ_D-bisimilarity.) □

4.2.4 Institutions of D-Coalgebras

The fact that destructor cosignatures and their morphisms are (suitably-restricted) many-sorted cosignatures and respectively cosignature morphisms, and that the reduct functors induced by destructor cosignature morphisms are obtained as restrictions of the reduct functors induced by the

underlying many-sorted cosignature morphisms to coalgebras of destructor cosignatures automatically yields an institution w.r.t. the satisfaction of hidden coequations by coalgebras of destructor cosignatures.

Theorem 4.2.18. *Let* $\mathsf{Coalg}_D : \mathsf{Cosign}_V \to \mathsf{CAT}^{\mathsf{op}}$ *denote the functor taking destructor cosignatures to their categories of coalgebras, and let* $\mathsf{HCoeqn} : \mathsf{Cosign}_V \to \mathsf{SET}$ *denote the functor taking destructor cosignatures to their sets of hidden coequations. Then,* $(\mathsf{Cosign}_V, \mathsf{Coalg}_D, \mathsf{HCoeqn}, \models)$ *is an institution.*

The specifications and specification morphisms of this institution will be referred to as *destructor specifications* and *destructor specification morphisms*.

Remark 4.2.19. Theorem 4.2.18 could have also been obtained by instantiating Proposition 3.1.48. For, if $\mathsf{G}_\Delta : \mathsf{Set}^S_D \to \mathsf{Set}^S_D$ denotes the endofunctor associated to the destructor cosignature (H, Δ) (see Remark 4.2.6), then many-sorted Δ-coequations of form $l = r$ if C, with $l, r \in T_\Delta[\mathcal{C}]_h$ for some $h \in H$, and with C of form $(t_1, \mathcal{C}_1), \ldots, (t_n, \mathcal{C}_n)$ induce $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coequations (K, l', r') having the property that $A \models_\Delta l = r$ if C is equivalent to $\langle A, \alpha\rangle \models_{(\mathsf{Set}^S_D, \mathsf{G}_\Delta)} (\mathsf{K}, l', r')$ for any Δ_D-coalgebra A with $\langle A, \alpha\rangle$ its associated $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coalgebra. Specifically, $\mathsf{K} : \mathsf{Set}^S_D \to \mathsf{Set}^S_D$ is given by:

$$(\mathsf{K}X)_s = \begin{cases} D_s & \text{if } s \in V \\ 1 & \text{if } s \in H \setminus \{h\} \\ 1 + \coprod_{s' \in S} \coprod_{Z \in \mathcal{C}_{s'}} X_{s'} & \text{if } s = h \end{cases}$$

for $X \in |\mathsf{Set}^S_D|$, while $l', r' : \mathsf{U} \Rightarrow \mathsf{KU}$ are defined similarly to l', r' in Remark 4.1.30 (where $\mathsf{U} : \mathsf{Coalg}(\mathsf{Set}^S_D, \mathsf{G}_\Delta) \to \mathsf{Set}^S_D$ takes $(\mathsf{Set}^S_D, \mathsf{G}_\Delta)$-coalgebras to their carrier).

Example 4.2.20. The coalgebraic specification morphism given in Example 4.1.41 also defines a destructor specification morphism. Moreover, this specification morphism is conservative (see Definition 3.1.51), as the coequations added by its target specification provide definitions up to bisimulation (i.e. *coinductive definitions*) for `odd_select` and `even_select`. This results in the existence of unique interpretations of `odd_select` and `even_select` in the final coalgebra of the source specification.

Again, suitable denotations for destructor specifications and destructor specification morphisms are provided by final and respectively cofree coalgebras.

Proposition 4.2.21. *Let* (Δ, E) *denote a destructor specification, let* F *denote a final* Δ_D*-coalgebra (as given by Corollary 4.2.8), and let* F_E *denote the largest many-sorted* Δ*-subcoalgebra of* F *which satisfies the coequations in* E*. Then,* F_E *defines, up to isomorphism in* $\mathsf{Coalg}(\Delta)$*, a final* (Δ_D, E)*-coalgebra.*

Proof. The fact that all the coequations in E are of hidden sort results in F_E defining, up to isomorphism in $\mathsf{Coalg}(\Delta)$, a Δ_D-coalgebra, while the maximality of F_E amongst the many-sorted Δ-subcoalgebras of F which satisfy the coequations in E results in it being final amongst the Δ_D-coalgebras satisfying E. □

Proposition 4.2.22. *Let $\phi : (\Delta, E) \to (\Delta', E')$ denote a destructor specification morphism, let A denote a (Δ_D, E)-coalgebra, let A' denote a cofree Δ'_D-coalgebra over A w.r.t.* $\mathsf{U}_\phi : \mathsf{Coalg}_D(\Delta') \to \mathsf{Coalg}_D(\Delta)$ *(as given by Proposition 4.2.9), and let A'_E denote the largest many-sorted Δ'-subcoalgebra of A' which satisfies the coequations in E'. Then, A'_E defines, up to isomorphism in* $\mathsf{Coalg}(\Delta')$*, a cofree (Δ'_D, E')-coalgebra over A w.r.t.* $\mathsf{U}_\phi{\restriction}_{\mathsf{Coalg}_D(\Delta', E')}$: $\mathsf{Coalg}_D(\Delta', E') \to \mathsf{Coalg}_D(\Delta, E)$.

Proof. Similar to the proof of Proposition 4.2.21. □

The final coalgebra of a destructor specification has the property that it satisfies precisely those hidden coequations in visible-sorted covariables which are semantic consequences of the coequations in the specification.

Proposition 4.2.23. *Let (Δ, E) denote a destructor specification, let F_E denote a final (Δ_D, E)-coalgebra, and let e denote a hidden Δ-coequation in visible-sorted covariables. Then, $E \models_\Delta e$ if and only if $F_E \models_\Delta e$.*

Proof. The **if** direction follows from 2 of Proposition 4.1.34 (the visible-sorted components of any Δ_D-homomorphism, and hence also of the unique Δ_D-homomorphisms into the final (Δ_D, E)-coalgebra, are injective), while the **only if** direction follows from $E \models_\Delta e$ together with $F_E \models_\Delta E$. □

As pointed out in Section 3.1.3, further restrictions need to be imposed to destructor cosignature morphisms in order to obtain an institution w.r.t. the notion of satisfaction of hidden coequations up to bisimulation. Such restrictions amount to the target cosignature not adding new observers for sorts from the source cosignature.

Definition 4.2.24. *A destructor cosignature morphism $\phi : (H, \Delta) \to (H', \Delta')$ is* **horizontal** *if and only if $\delta' \in \Delta'_{\phi(h), w'}$ with $h \in H$ and $w' \in S'^+$ implies $\delta' = \phi(\delta)$ for some $\delta \in \Delta_{h,w}$ with $w \in S^+$.*

The category of destructor cosignatures over V and horizontal destructor cosignature morphisms is denoted $\mathsf{HCosign}_V$.

Remark 4.2.25. If $(\mathsf{U}, \eta_\phi) : (\mathsf{Set}^S_D, \mathsf{G}_\Delta) \to (\mathsf{Set}^{S'}_D, \mathsf{G}_{\Delta'})$ denotes the abstract cosignature morphism induced by a horizontal destructor cosignature morphism $\phi : (H, \Delta) \to (H', \Delta')$ (see Remark 4.2.6), then (U, η_ϕ) is horizontal.

Remark 4.2.25 together with Proposition 3.1.55 now yield the following.

Theorem 4.2.26. $(\mathsf{HCosign}_V, \mathsf{Coalg}_D{\restriction}_{\mathsf{HCosign}_V}, \mathsf{HCoeqn}{\restriction}_{\mathsf{HCosign}_V}, \models^{\mathsf{b}})$ *is an institution.*

A more general version of Proposition 4.2.23 holds for the satisfaction of hidden coequations up to bisimulation.

Proposition 4.2.27. *Let (Δ, E) denote a destructor specification, let F_E denote a final (Δ_D, E)-coalgebra, and let e denote a hidden Δ-coequation. Then, $E \models^{\mathsf{b}}_\Delta e$ if and only if $F_E \models_\Delta e$.*

Proof (sketch). The **if** direction follows from the fact that coalgebra homomorphisms (in particular, homomorphisms into the final (Δ_D, E)-coalgebra) reflect bisimulations, while the **only if** direction follows from $F_E \models^{\mathsf{b}}_{\Delta} E$. □

Remark 4.2.28. Theorem 4.2.3 together with Corollary 2.2.6 result in the categories of specifications associated to the institutions $(\mathsf{Cosign}_V, \mathsf{Coalg}_D, \mathsf{HCoeqn}, \models)$ and $(\mathsf{HCosign}_V, \mathsf{Coalg}_D{\restriction_{\mathsf{HCosign}_V}}, \mathsf{HCoeqn}{\restriction_{\mathsf{HCosign}_V}}, \models^{\mathsf{b}})$ also being finitely cocomplete.

The following compositionality result also holds.

Theorem 4.2.29. *The functor* $\mathsf{Coalg}_D : \mathsf{Cosign}_V \to \mathsf{CAT}^{\mathsf{op}}$ *preserves finite colimits.*

4.2.5 Deduction

The deduction rules of many-sorted coalgebra are sound for the satisfaction of hidden coequations by coalgebras of destructor specifications. However, in order to derive a completeness result, additional deduction rules are required. It turns out that adding the following rule:

$$[\,\textbf{unity}\,]\ \frac{}{E \vdash t = t' \text{ if } (t, Z), (t', Z)} \quad Z : v,\ |D_v| = 1$$

(inspired by a rule in [Cor98]) to the deduction calculus in Section 4.1.6 yields a calculus which is both sound and complete for the satisfaction of hidden coequations by coalgebras of destructor specifications.

Theorem 4.2.30 (Soundness). *The deduction calculus obtained by adding the* **unity** *rule to the deduction calculus of many-sorted coalgebra is sound for the satisfaction of hidden* Δ*-coequations by* (Δ_D, E)*-coalgebras.*

Proof. Soundness of the deduction rules of many-sorted coalgebra follows from Theorem 4.1.50 together with the fact that any (Δ_D, E)-coalgebra is a (many-sorted) (Δ, E)-coalgebra. Also, soundness of the **unity** rule follows from the fact that if $Z : v$ and $|D_v| = 1$, then $t_A(a) = t'_A(a)$ holds whenever $t_A(a), t'_A(a) \in \iota_Z(D_v)$, for any Δ_D-coalgebra A and any $a \in A_s$. □

The completeness proof follows the same line as in the many-sorted case.

Lemma 4.2.31. *Let* (Δ, E) *denote a destructor specification, and let* F_E *denote a final* (Δ_D, E)*-coalgebra. Also, let* $h \in H$*, and let* C *denote some conditions for the sort* h*. If* $E \not\vdash l = r$ *if* C *for any* $l, r \in T_\Delta[\mathcal{C}]_h$ *with* $covar(l) \cap covar(r) = \emptyset$*, then* $F^C_{E,h} \neq \emptyset$*.*

Proof. We begin by noting that F_E is isomorphic to the many-sorted, cofree (Δ, E)-coalgebra over the S-sorted set C given by: $C_v = D_v$ for $v \in V$, and $C_h = \{*\}$ for $h \in H$. (This is a consequence of Proposition 4.2.8 together with Proposition 4.2.21.)

The proof is now similar to the proof of Lemma 4.1.52. We show that $F^C_{E,h}$ has a surjective mapping into the limit object L of the ω^{op}-chain defined in Lemma 4.1.52. Specifically, we show

that $\varphi \in F^C_{E,h} \mapsto (\pi_1(\varphi(t_i)))_{i\in\{1,2,\ldots\}} \in L$ defines a surjective mapping from $F^C_{E,h}$ to L. Then, $F^C_{E,h} = \emptyset$ gives $L = \emptyset$, which, by the proof of Lemma 4.1.52, gives $E \vdash l = r$ if C for some $l, r \in T_\Delta[\mathcal{C}]$ with $covar(l) \cap covar(r) = \emptyset$, thus contradicting the hypothesis.

For the above mapping to be correctly defined, we must show that $(\pi_1(\varphi(t_i)))_{i\in\{1,2,\ldots\}} \in L$ for each $\varphi \in F^C_{E,h}$. If this was not the case for some $\varphi \in F^C_{E,h}$, then $E \vdash l = r$ if $C, (t_1, Z_{t_1}), \ldots, (t_n, Z_{t_n})$ for some $l, r \in T_\Delta[\mathcal{C}]_h$ with $covar(l) \cap covar(r) = \emptyset$ and some $n \in \{1, 2, \ldots\}$ would contradict the soundness of $\vdash$ (as both C and each (t_i, Z_{t_i}) with $i = 1, \ldots, n$ hold in $\varphi \in F^C_{E,h}$, whereas $l = r$ does not). Hence, $(\pi_1(\varphi(t_i)))_{i\in\{1,2,\ldots\}} \in L$ for each $\varphi \in F^C_{E,h}$.

To show that the mapping $\varphi \mapsto (Z_{t_i})_{i\in\{1,2,\ldots\}}$ is surjective, we fix $c_s \in C_s$ for each $s \in S$. Then, given $(Z_{t_i})_{i\in\{1,2,\ldots\}} \in L$, we let $\varphi \in F^C_{E,h}$ be given by $\varphi(t_i) = \langle Z_{t_i}, c_{t_i}\rangle$ for $i = 1, 2, \ldots$, with $c_{t_i} = c_{s_i}$ if $Z_{t_i} : s_i$, for $i = 1, 2, \ldots$.

The proof of $\varphi \in F^C_{E,h}$ is based on the proof of $\varphi \in F^C_{E,s}$ in Lemma 4.1.52.

First, given $t_i, t_j \in T^1_{\Delta,h}$ with $t_j = [t'_1/Z_1, \ldots, t'_n/Z_n]t_i$ and with $Z_{t_i} = Z_k$, the proof of Lemma 4.1.52 gives $Z_{t_j} \in covar(t'_k)$. Moreover, if $t'_k = Z_{t_j}$ then $s_i = s_j$, and hence $c_{t_i} = c_{t_j}$.

Next, given $t \in T^1_\Delta[\{Z_1, \ldots, Z_n\}]_h$, $i \in \{1, \ldots, n\}$ and $(t_i = t'_i \text{ if } C_i) \in E$ such that $t_F(\varphi) \in \iota_{Z_i}(F_{s_i})$, $t_i, t'_i \in T_\Delta[\mathcal{C}_i]_{s_i}$ and C_i holds in $t_F(\varphi)$, and given coterms $u_1, \ldots, u_q$ of suitable sort, the proof of Lemma 4.1.52 gives $l_F(\varphi), r_F(\varphi) \in \iota_Z(F_s)$ for some $Z : s$. Moreover, $l_F(\varphi) = r_F(\varphi)$, as both are equal to c_s. Hence, $\varphi \in F_{E,h}$. The proof of Lemma 4.1.52 also gives $\varphi \in F^C_{E,h}$.

Hence, $F^C_{E,h}$ has a surjective mapping into L. This concludes the proof. □

Lemma 4.2.32. *Let (Δ, E) denote a destructor specification, let $l, r \in T_\Delta[\mathcal{C}]_h$ for some set $\mathcal{C}$ of covariables and some $h \in H$, and let $Z \in \mathcal{C}$. Also, let A_E denote a cofree (Δ_D, E)-coalgebra over the S-sorted set $C \in |\mathsf{Set}^S_D|$ given by: $C_v = D_v$ for $v \in V$, and $C_h = \{*, *'\}$ for $h \in H$. If $E \not\vdash l = r$ if $C, (l, Z), (r, Z)$, then there exists $\varphi \in A^{C,(l,Z),(r,Z)}_{E,h}$ such that $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$.*

Proof. Again, we use the fact that A_E is isomorphic to the (many-sorted) cofree (Δ, E)-coalgebra over C.

Say $Z : s$ with $s \in S$. One can immediately infer that $s \in V$ implies $|D_s| > 1$ (otherwise **unity** together with **weakening** would yield $E \vdash l = r$ if $C, (l, Z), (r, Z)$). Let $c_s, c'_s \in C_s$ be such that $c_s \neq c'_s$.

The proof is now similar to the proof of Lemma 4.1.53. An element of the limit object L of the ω^{op}-chain defined in Lemma 4.1.53 is used to construct $\varphi \in A^{C,(l,Z),(r,Z)}_{E,h}$ with $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$, under the assumption that $E \not\vdash l = r$ if $C, (l, Z), (r, Z)$. Specifically, given $(Z_{t_i})_{i=1,2,\ldots} \in L$, $\varphi \in A^{C,(l,Z),(r,Z)}_{E,h}$ is given by $\varphi(t) = \langle Z_t, c_t\rangle$ for $t \in T^1_{\Delta,h}$, where:

$$c_t = \begin{cases} c_s & \text{if } \ t \notin T, \ Z_t : s \\ c'_s & \text{if } \ t \in T, \ Z_t : s \end{cases}$$

and where T is defined as in Lemma 4.1.53.

The proof of $\varphi \in A^{C,(l,Z),(r,Z)}_{E,h}$ is similar to the proof of $\varphi \in A^{C,(l,Z),(r,Z)}_{E,s}$ in Lemma 4.1.53. Lemma 4.1.53 also gives $l_{A_E}(\varphi) \neq r_{A_E}(\varphi)$. This concludes the proof. □

Theorem 4.2.33 (Completeness). *The deduction calculus obtained by adding the* **unity** *rule to the deduction calculus of many-sorted coalgebra is complete for the satisfaction of hidden coequations by coalgebras of destructor specifications.*

Proof. Let (Δ, E) denote a destructor specification, and let e denote a hidden Δ-coequation such that $E \models_\Delta e$. Also, let F_E denote a final (Δ_D, E)-coalgebra, and let A_E denote a cofree (Δ_D, E)-coalgebra over the S-sorted set C defined in Lemma 4.2.32. The proof of $E \vdash e$ is similar to the corresponding proof in Theorem 4.1.54. If $F^C_{E,h} = \emptyset$, then $E \vdash e$ follows by Lemma 4.2.31. Also, if $F^C_{E,h} \neq \emptyset$, Lemma 4.2.31 and respectively Lemma 4.2.32 are used to show that the assumption that $E \not\vdash e$ yields a contradiction. □

Example 4.2.34. Consider the list specification given in Example 4.1.29, regarded as a destructor specification with visible sorts `1` and `Elt`, and let E consist of the two coequations defining the list invariant. Then, provided that the sort `1` is interpreted by D as a one-element set, one can show that the following holds:

$E \vdash$ `[Z,E]first = [Z,L]rest if ([Z,E]first,Z)`

This follows by **case-analysis** from:

$E \vdash$ `[Z,E]first = [Z,L]rest if ([Z,E]first,Z),([Z,L]rest,Z)`

and:

$E \vdash$ `[Z,E]first = [Z,L]rest if ([Z,E]first,Z),([Z,L]rest,L)`

with the first statement following by **unity** (as `Z:1`), and with the second statement following by **contradiction** from:

$E \vdash$ `[Z,L]rest = [Z',L']rest if ([Z,E]first,Z),([Z,L]rest,L)`

The last statement follows by **transitivity** from:

$E \vdash$ `[Z,L]rest = [Z,L']rest if ([Z,E]first,Z),([Z,L]rest,L)`

(following by **base** and **weakening**), and:

$E \vdash$ `[Z,L]rest = [Z',L]rest if ([Z,E]first,Z),([Z,L]rest,L)`

(following by **cond** and **weakening**).

We conclude this section by noting that the (extended) deduction calculus is also sound for the satisfaction up to bisimulation of hidden coequations, and complete for the satisfaction up to bisimulation of hidden coequations with no hidden-sorted covariables.

Theorem 4.2.35. *Let (Δ, E) denote a destructor specification, and let e denote a hidden Δ-coequation. Then, the following hold:*

1. *$E \vdash e$ implies $E \models^{\mathsf{b}}_\Delta e$*
2. *$E \models^{\mathsf{b}}_\Delta e$ implies $E \vdash e$ if e contains no hidden-sorted covariables.*

Proof. Soundness of $\vdash$ for $\models^{\mathsf{b}}$ follows from the soundness of $\vdash$ for $\models$, after observing that $A \models^{\mathsf{b}}_{\Delta} E$ (respectively $A \models^{\mathsf{b}}_{\Delta} e$) is equivalent to $A/_{\sim_A} \models_{\Delta} E$ ($A/_{\sim_A} \models_{\Delta} e$), for any Δ_D-coalgebra A (see Proposition 3.1.38).

Completeness of $\vdash$ for the satisfaction of hidden coequations with no hidden-sorted covariables follows from the completeness of $\vdash$ for $\models$, together with $E \models^{\mathsf{b}}_{\Delta} e$ being equivalent to $E \models_{\Delta} e$ in the case when all the covariables appearing in e are visible-sorted. (The fact that $A/_{\sim_A} \models_{\Delta} e$ is equivalent to $A \models_{\Delta} e$ whenever e only contains visible-sorted covariables is used here.) □

4.2.6 Expressiveness

This section discusses the expressiveness of the formalism previously introduced, both w.r.t. the structures specifiable within it, and from the point of view of characterising classes of models by means of coequations.

The main result of this section states that, by allowing a change in the underlying category, any (extended) polynomial endofunctor can be transformed into an endofunctor whose components have the form of products of finite coproducts of constant/projection functors, with the categories of coalgebras of the two endofunctors being isomorphic. Endofunctors of the previously-mentioned form are then shown to induce destructor cosignatures (over suitably-chosen data universes), with the categories of coalgebras of the induced cosignatures being isomorphic to the categories of coalgebras of the given endofunctors. That is, destructor cosignatures are at least as expressive as extended polynomial endofunctors[14] from the point of view of the structures specifiable with them.

Theorem 4.2.36. *Let* $\mathsf{T} : \mathrm{Set}^S \to \mathrm{Set}^S$ *denote an extended polynomial endofunctor. Then,* $\mathrm{Coalg}(\mathsf{T}) \simeq \mathrm{Coalg}(\mathsf{G}_\mathsf{T})$ *for some endofunctor* $\mathsf{G}_\mathsf{T} : \mathrm{Set}^{S_\mathsf{T}} \to \mathrm{Set}^{S_\mathsf{T}}$ *whose components have the form of products of finite coproducts of constant/projection functors.*

Proof. We define $S_\mathsf{T} \in |\mathrm{Set}|$ and $\mathsf{G}_\mathsf{T} : \mathrm{Set}^{S_\mathsf{T}} \to \mathrm{Set}^{S_\mathsf{T}}$ by structural induction on the components of T.

For $\mathsf{F} : \mathrm{Set}^S \to \mathrm{Set}$ an extended polynomial functor, we let $S_\mathsf{F} \in |\mathrm{Set}|$, $\mathsf{G}_\mathsf{F} : \mathrm{Set}^{S+S_\mathsf{F}} \to \mathrm{Set}$ and $(\mathsf{F}_{s'})_{s' \in S_\mathsf{F}}$ with $\mathsf{F}_{s'} : \mathrm{Set}^{S+S_\mathsf{F}} \to \mathrm{Set}$ for $s' \in S_\mathsf{F}$ be defined as follows:

1. if $\mathsf{F} = \Pi_s$ for some $s \in S$ or if $\mathsf{F} = A$, then:
   (a) $S_\mathsf{F} = \emptyset$
   (b) $\mathsf{G}_\mathsf{F} = \mathsf{F}$

   (F is already of the required form.)
2. if $\mathsf{F} = \mathsf{F}_1 \times \mathsf{F}_2$, then:
   (a) $S_\mathsf{F} = S_{\mathsf{F}_1} + S_{\mathsf{F}_2}$
   (b) $\mathsf{G}_\mathsf{F} = (\mathsf{G}_{\mathsf{F}_1}\Pi_1) \times (\mathsf{G}_{\mathsf{F}_2}\Pi_2)$

[14] Destructor cosignatures are actually more expressive than extended polynomial endofunctors, as they also cover endofunctors whose components have the form of *infinite* products of finite coproducts of constant/projection functors.

(c) $\mathsf{F}_{s'} = \begin{cases} (\mathsf{F}_1)_{s_1}\Pi_1 & \text{if } s' = \iota_1(s_1) \text{ for some } s_1 \in S_{\mathsf{F}_1} \\ (\mathsf{F}_2)_{s_2}\Pi_2 & \text{if } s' = \iota_2(s_2) \text{ for some } s_2 \in S_{\mathsf{F}_2} \end{cases}$

where for $i \in \{1,2\}$, $\Pi_i : \mathsf{Set}^{S+S_\mathsf{F}} \to \mathsf{Set}^{S+S_{\mathsf{F}_i}}$ denotes the projection functor induced by the injection $1_S + \iota_i : S + S_{\mathsf{F}_i} \to S + S_\mathsf{F}$. (If F_1 and F_2 have already been transformed into the required form, then the representation of F as a functor of the required form is obtained by tupling the representations of F_1 and F_2.)

3. if $\mathsf{F} = \mathsf{F}_1 + \mathsf{F}_2$, then:

   (a) $S_\mathsf{F} = S_{\mathsf{F}_1} + S_{\mathsf{F}_2} + \{s_1'\} + \{s_2'\}$

   (b) $\mathsf{G}_\mathsf{F} = \Pi_1' + \Pi_2'$

   (c) $\mathsf{F}_{s'} = \begin{cases} (\mathsf{F}_1)_{s_1}\Pi_1 & \text{if } s' = \iota_1(s_1) \text{ for some } s_1 \in S_{\mathsf{F}_1} \\ (\mathsf{F}_2)_{s_2}\Pi_2 & \text{if } s' = \iota_2(s_2) \text{ for some } s_2 \in S_{\mathsf{F}_2} \\ \mathsf{G}_{\mathsf{F}_1}\Pi_1 & \text{if } s' = \iota_3(s_1') \\ \mathsf{G}_{\mathsf{F}_2}\Pi_2 & \text{if } s' = \iota_4(s_2') \end{cases}$

   where, for $i \in \{1,2\}$, $\Pi_i : \mathsf{Set}^{S+S_\mathsf{F}} \to \mathsf{Set}^{S+S_{\mathsf{F}_i}}$ denotes the projection functor induced by the injection $1_S + \iota_i : S + S_{\mathsf{F}_i} \to S + S_\mathsf{F}$, while $\Pi_i' : \mathsf{Set}^{S+S_\mathsf{F}} \to \mathsf{Set}$ denotes the projection functor induced by the injection $\iota_{i+3} : \{s_i'\} \to S + S_{\mathsf{F}_1} + S_{\mathsf{F}_2} + \{s_1'\} + \{s_2'\}$. (F is transformed into a functor of the required form by transferring the structures specified by F_1 and F_2 to two new sorts s_1' and s_2', and then capturing the structure specified by F by means of an operation symbol with result type $s_1' + s_2'$.)

4. if $\mathsf{F} = (\mathsf{F}_1)^A$, then:

   (a) $S_\mathsf{F} = S_{\mathsf{F}_1}$

   (b) $\mathsf{G}_\mathsf{F} = \prod_{a \in A} \mathsf{G}_{\mathsf{F}_1}$

   (c) $\mathsf{F}_{s'} = (\mathsf{F}_1)_{s'}$ for each $s' \in S_{\mathsf{F}_1}$

   (The existence of a Set-isomorphism between B^A and $\prod_{a \in A} B$, with $A, B \in |\mathsf{Set}|$ is used here to transform F into a functor of the required form, provided that F_1 has already been transformed into the required form.)

It then follows by structural induction on F that both G_F and each of the $\mathsf{F}_{s'}$ with $s' \in S_\mathsf{F}$ are in the form of products of finite coproducts of constant/projection functors.

Now given $\mathsf{T} : \mathsf{Set}^S \to \mathsf{Set}^S$, $\mathsf{T} = (\mathsf{T}_s)_{s \in S}$, let $S_\mathsf{T} = S + \coprod_{s \in S} S_{(\mathsf{T}_s)}$ and $\mathsf{G}_\mathsf{T} : \mathsf{Set}^{S_\mathsf{T}} \to \mathsf{Set}^{S_\mathsf{T}}$ be given by:

$$(\mathsf{G}_\mathsf{T})_{s'} = \begin{cases} \mathsf{G}_{(\mathsf{T}_s)}\Pi_s & \text{if } s' = \iota_1(s) \text{ for some } s \in S \\ (\mathsf{T}_s)_{s''}\Pi_s & \text{if } s' = \iota_2(\iota_s(s'')) \text{ for some } s \in S \text{ and some } s'' \in S_{(\mathsf{T}_s)} \end{cases}$$

where, for $s \in S$, $\Pi_s : \mathsf{Set}^{S_\mathsf{T}} \to \mathsf{Set}^{S+S_{(\mathsf{T}s)}}$ denotes the projection functor induced by the injection $1_S + \iota_S : S + S_{(\mathsf{T}_s)} \to S_\mathsf{T}$.

Then, the fact that $\mathsf{Coalg}(\mathsf{T}) \simeq \mathsf{Coalg}(\mathsf{G}_\mathsf{T})$ follows by structural induction on (the components of) T. □

Example 4.2.37. We exemplify the construction of G_T by taking T to be the endofunctor typically used to specify binary trees, that is, $\mathsf{T} : \mathsf{Set}^{\{\mathtt{Tree}\}} \to \mathsf{Set}^{\{\mathtt{Tree}\}}$, $\mathsf{T} = 1 + (\mathsf{Id} \times \mathsf{Id})$.

In this case, the construction gives:

1. $S_1 = \emptyset$
   $\mathsf{G}_1 = 1$
2. $S_{\mathsf{Id}} = \emptyset$
   $\mathsf{G}_{\mathsf{Id}} = \mathsf{Id}$
3. $S_{\mathsf{Id}\times\mathsf{Id}} = \emptyset$
   $\mathsf{G}_{\mathsf{Id}\times\mathsf{Id}} = \mathsf{G}_{\mathsf{Id}} \times \mathsf{G}_{\mathsf{Id}} = \mathsf{Id} \times \mathsf{Id}$
4. $S_{1+(\mathsf{Id}\times\mathsf{Id})} = \{\mathtt{Leaf}, \mathtt{Node}\}$
   $\mathsf{G}_{1+(\mathsf{Id}\times\mathsf{Id})} : \mathsf{Set}^{\{\mathtt{Tree},\mathtt{Leaf},\mathtt{Node}\}} \to \mathsf{Set}$, $\mathsf{G}_{1+(\mathsf{Id}\times\mathsf{Id})} = \Pi_{\mathtt{Leaf}} + \Pi_{\mathtt{Node}}$
   $(1 + (\mathsf{Id} \times \mathsf{Id}))_{\mathtt{Leaf}} = \mathsf{G}_1\Pi_{\mathtt{Tree}} = 1$
   $(1 + (\mathsf{Id} \times \mathsf{Id}))_{\mathtt{Node}} = \mathsf{G}_{\mathsf{Id}\times\mathsf{Id}}\Pi_{\mathtt{Tree}} = \Pi_{\mathtt{Tree}} \times \Pi_{\mathtt{Tree}}$
   (where $\Pi_{\mathtt{Tree}}, \Pi_{\mathtt{Leaf}}, \Pi_{\mathtt{Node}} : \mathsf{Set}^{\{\mathtt{Tree},\mathtt{Leaf},\mathtt{Node}\}} \to \mathsf{Set}$ denote the corresponding projection functors)

Hence, $\mathsf{G}_\mathsf{T} : \mathsf{Set}^{\{\mathtt{Tree},\mathtt{Leaf},\mathtt{Node}\}} \to \mathsf{Set}^{\{\mathtt{Tree},\mathtt{Leaf},\mathtt{Node}\}}$ is given by:

$$\mathsf{G}_\mathsf{T}(X, Y, Z) = (Y + Z, 1, X \times X)$$

for $X, Y, Z \in |\mathsf{Set}|$.

Corollary 4.2.38. *Let* $\mathsf{T} : \mathsf{Set}^S \to \mathsf{Set}^S$ *denote an extended polynomial endofunctor, such that all the sets appearing as exponents in* T *are enumerable*[15]*. Then,* $\mathsf{Coalg}(\mathsf{T}) \simeq \mathsf{Coalg}_D(\Delta)$ *for some data universe* (V, D) *and some destructor cosignature* (H, Δ) *over* V*.*

Proof. Let $\mathsf{G}_\mathsf{T} : \mathsf{Set}^{S_\mathsf{T}} \to \mathsf{Set}^{S_\mathsf{T}}$ denote the endofunctor yielded by Theorem 4.2.36. Since all the sets appearing as exponents in T are enumerable, it follows by the proof of Theorem 4.2.36 that all the components of G_T have the form of enumerable products of finite coproducts of constant/projection functors.

Now let V contain a sort v for each constant functor A appearing in the definition of G_T, and let D denote the V-sorted set whose v-component is given by the corresponding A, for each $v \in V$. Also, let $H = S_\mathsf{T}$, and let Δ denote the destructor cosignature induced by the endofunctor $\mathsf{G}' : \mathsf{Set}_D^{V \cup S_\mathsf{T}} \to \mathsf{Set}_D^{V \cup S_\mathsf{T}}$ whose S_T-components are obtained from the corresponding components of G_T by replacing each occurrence of a constant functor A with the corresponding projection functor $\Pi_v : \mathsf{Set}_D^{V \cup S_\mathsf{T}} \to \mathsf{Set}$.

[15] This condition ensures that the set of operations of the induced destructor cosignature is enumerable.

The construction of G' immediately yields $\mathsf{Coalg}(\mathsf{G}') \simeq \mathsf{Coalg}(\mathsf{G}_\mathsf{T})$. Also, Theorem 4.2.36 gives $\mathsf{Coalg}(\mathsf{T}) \simeq \mathsf{Coalg}(\mathsf{G}_\mathsf{T})$, while Remark 4.2.6 gives $\mathsf{Coalg}(\mathsf{G}') \simeq \mathsf{Coalg}_D(\Delta)$. Hence, $\mathsf{Coalg}(\mathsf{T}) \simeq \mathsf{Coalg}_D(\Delta)$. This concludes the proof. □

In particular, if T is a polynomial endofunctor, then T-coalgebras are the same as Δ-coalgebras over D, for suitably-chosen D and Δ. That is, destructor cosignatures are at least as expressive for coalgebra as many-sorted signatures are for algebra. However, unlike in the algebraic case, moving from one-sorted to many-sorted coalgebras, and from many-sorted cosignatures to destructor cosignatures is actually necessary (as well as sufficient) in order to model via cosignatures arbitrary (extended) polynomial endofunctors[16].

We conclude this section with a few remarks on the expressive power of equational sentences in coalgebraic specification. It has been shown in [Cor98] that such sentences are not sufficiently expressive to yield a Birkhoff-style characterisability result. The notion of coequation used here differs slightly from the one used in [Cor98], but, as the next two examples will illustrate, has an expressive power similar to the one in [Cor98].

Example 4.2.39. Consider the many-sorted cosignature consisting of sorts $\mathtt{1}$ and $\mathtt{Stream}$, and of an operation symbol $\mathtt{next} : \mathtt{Stream} \to \mathtt{1}\ \mathtt{Stream}$. Also, define a finite stream to be an element a of a coalgebra A of this cosignature, additionally satisfying: $\mathtt{next}_A^n(a) \in \iota_n(A_\mathtt{1})$ for some $n \in \{1, 2, \ldots\}$, where $\mathtt{next}_A^n : A_\mathtt{1} + (\ldots + (A_\mathtt{1} + A_\mathtt{Stream}) \ldots)$ with $n \in \{1, 2, \ldots\}$ is defined inductively by: $\mathtt{next}_A^1 = \mathtt{next}_A$ and $\mathtt{next}_A^{n+1} = (1_{A_\mathtt{1}} + \mathtt{next}_A^n) \circ \mathtt{next}_A$ for $n = 1, 2, \ldots$. Then, it is easy to check that the coalgebras whose elements denote finite streams constitute a covariety which is not specifiable by coequations. Furthermore, this also holds when regarding the above many-sorted cosignature as a destructor cosignature over $\{\mathtt{1}\}$.

Example 4.2.40. Consider the list specification given in Example 4.1.29, regarded as a destructor specification with visible sorts $\mathtt{1}$ and $\mathtt{Elt}$. Then, the coalgebras of this specification which, in addition, have the property that their elements denote lists with any two adjacent elements being different from each other constitute a covariety[17]. However, this covariety is not coequationally specifiable[18].

This lack of expressiveness is caused by the fact that the coequations used to characterise arbitrary covarieties (see Section 3.1.6) do not have finitary syntactic presentations. This is different from the algebraic case, where the equations used to characterise arbitrary varieties (see Section 3.2.6) induce congruence relations on algebras of terms.

In spite of not being able to provide a characterisability result similar to Birkhoff's, coequations succeed in capturing, in a concise manner, observational properties quantified over the entire state spaces of the systems being specified.

[16] The introduction of new sorts is necessary because coproducts do not distribute over products in Set, while the use of destructor cosignatures is necessary to capture constant functors.

[17] The fact that the interpretation of the sort $\mathtt{Elt}$ is fixed, and that the $\mathtt{Elt}$-component of any coalgebra homomorphism is the identity is used here.

[18] A coequational specification of lists having the above-mentioned property can, however, be given in a setting where the underlying data is allowed to carry algebraic structure. This issue will be dealt with in Chapter 5.

4.2.7 Related Work

This section briefly compares the approach presented in this chapter with other equational approaches to coalgebraic specification [Jac96c, Jac96a, Jac97, Cor98, Roș98, Gol], as well as with modal logic approaches to coalgebraic specification [Mos99, Kur98, Kur00, Röß98, Röß00, Jac00].

To a certain extent, our approach can be regarded as a generalisation of both [Jac96c, Jac96a, Jac97] and [Cor98]. For, destructor cosignatures are able to specify arbitrary extended polynomial endofunctors (see Corollary 4.2.38), including those considered in [Jac96c, Jac96a, Jac97] (see Example 3.1.7) and [Cor98] (see Example 3.1.8). For instance, the destructor cosignature induced by the endofunctor considered in Example 3.1.46 consists of visible sorts 1 and A, a hidden sort `Stack`, and operation symbols $\mathtt{push}_a : \mathtt{Stack} \to \mathtt{Stack}$ for $a \in A$, $\mathtt{pop} : \mathtt{Stack} \to \mathtt{Stack}$ and $\mathtt{top} : \mathtt{Stack} \to 1 + A$. Furthermore, any assertion (see Example 3.1.46) or equation (see Example 3.1.27) in which constants are only used as arguments to observers, and whose conditions[19] (if any) are type constraints, induces a coequation over the corresponding destructor cosignature. For instance, the coequations induced by the the second and third assertions in Example 3.1.46 are as follows:

$$[[S]\mathtt{pop}]\mathtt{push}_a = S$$
$$[S]\mathtt{pop} = S \text{ if } ([Z,S]\mathtt{top},Z)$$

(and, in order to agree with the approach in [Jac96c, Jac96a, Jac97], should be required to hold up to bisimulation).

Still, a major difference between the approaches in [Jac96c, Jac96a, Jac97, Cor98] and the one here stands in the use, in [Jac96c, Jac96a, Jac97, Cor98], of data values as *constant observations*. An instance of this is provided by the first assertion in Example 3.1.46, which has no correspondent in the form of a coequation. However, the view taken here is that a distinction should be made between the structure of systems and their functionality, with coalgebra being used to specify structural properties, and with algebra being used to specify functionality (see Chapter 3). Moreover, in our opinion, constraints involving constant observations should only be imposed to particular states (such as those yielded by certain constructors), and hence should not be considered at this level. A similar observation can be made about operations with structured domains (such as the push operation in Example 3.1.46), which are typically related to the functionality of systems, and therefore should not be considered when coalgebraically specifying state spaces[20].

Closure properties of equationally-definable classes of coalgebras are also investigated in [Roș98, Gol], where it is shown that characterisability results similar to Birkhoff's can be formulated by considering a different notion of covariety. The setting considered in [Roș98] is that of hidden algebra (see Examples 3.1.6 and 3.1.37), with no generalised constants being allowed in hidden signatures, and with equations containing at most one variable of a hidden sort. [Roș98] then shows that the equationally-definable classes of hidden algebras are precisely those which are closed under domains and images of homomorphisms, coproducts and *representative inclusions* (defined as inclusions whose codomains satisfy exactly the same equations as their domains[21]). The results in [Roș98] are taken

[19] This only applies to [Jac96c, Jac96a, Jac97].

[20] In [Cor98], viewing *parameterised methods* as observers resulted in complications when attempting to define the associated operations by equations (as arbitrary algebraic terms were not allowed in equations).

[21] Closure of $\mathcal{C}$ under representative inclusions requires that, whenever $\iota : A \hookrightarrow B$ is a representative inclusion and

one step further by [Gol], where the (not fully structural) requirement involving the closure under representative inclusions is replaced by a requirement involving closure under *filter enlargements* (with the filter enlargement of a coalgebra A being defined solely in terms of the structure of A). Also, the sentences considered in [Gol] are more general than those considered in [Roș98], being given by boolean combinations of equations (with the associated notion of satisfaction being, as in [Roș98], behavioural). The main result in [Gol] characterises the classes of models definable by *observable formulae* (i.e. formulae only involving equations of visible sort) as classes which are closed under domains and images of homomorphisms, coproducts and *ultrafilter enlargements* (with the elements of the ultrafilter enlargement of a coalgebra A being given by suitably-closed collections of subsets of the carrier of A, and with the structure of the ultrafilter enlargement of A being determined by the structure of A). Another result in [Gol] states that the classes of models definable by conditional equations with observable premises are precisely those which are closed under domains and images of homomorphisms, coproducts and filter enlargements. In formulating these results, extensive use is made of equations of form $t = d$, with t a term and d a data constant. As previously noted, the approach presented here does not accommodate such equations. Consequently, similar results are unlikely to hold in our setting.

As far as modal logic approaches to coalgebraic specification are concerned, one of their main assets stands in the existence of results concerning the expressiveness of modal formulae both w.r.t. single states and w.r.t. covarieties. Specifically, in such approaches, *logical equivalence of states* (defined as the satisfaction of the same modal formulae) coincides with bisimilarity (see [Mos99, Theorem 6.6], [Röß00, Proposition 4.8]). This results in the existence of characterising formulae for single states (see [Mos99, Theorem 7.1]), as well as in the existence of characterisability results for covarieties (see [Kur00, Theorem 2.8.2]). However, infinite conjunctions and disjunctions are typically needed in such approaches to characterise single states and respectively covarieties by means of modal formulae (see [Mos99, Kur00]), while the formulation of completeness results requires a restriction to finitary sentences, as well as the satisfaction of some rather restrictive finiteness conditions by the endofunctors in question (see [Röß00, Section 5], [Kur00, Theorem 3.4.2]). While not sufficiently expressive to yield similar characterisability results, equational approaches only employ finitary sentences, and do not require any additional restrictions in order to derive completeness results (see [Cor98, Theorem 15], Theorem 4.2.33).

Some further remarks can be made about the relationship between the equational approach presented here and the modal logic approaches described in [Röß00] and respectively [Jac00]. The endofunctors considered there are endofunctors on Set constructed from constant functors, the identity functor, binary products and coproducts, exponentiation with fixed exponent and powerset (in [Röß00]), and respectively extended polynomial endofunctors on Set (in [Jac00]). In both cases, the modal operators used are determined by the form of the endofunctor considered. However, *strong next-time operators* are considered in [Röß00], whereas *weak next-time operators* are used in [Jac00]. To illustrate the differences between [Röß00] and [Jac00] on the one hand, and between [Röß00, Jac00] and the approach presented here on the other, we consider as an example the endofunctor G : Set $\to$ Set given by:

$$\mathsf{G}X = (1 + X) \times (1 + \mathbb{N})$$

A is in $\mathcal{C}$, then B is also in $\mathcal{C}$.

for $X \in |\mathsf{Set}|$. The destructor cosignature that corresponds to this endofunctor is the one considered in Example 4.1.29. Also, the modal language associated to G by the approach in [Röß00] is defined by:

$$\varphi ::= \bot \mid \varphi \to \psi \mid \texttt{<tailF>*} \mid \texttt{<tailS>}\varphi \mid \texttt{<headF>*} \mid \texttt{<headS>}n$$

with $n \in \mathbb{N}$, whereas the modal language associated to G by the approach in [Jac00] is defined by:

$$\begin{aligned}\varphi ::= &\bot \mid \varphi \to \psi \mid \texttt{[tailF]*} \mid \texttt{[tailF]}\bot \mid \texttt{[tailS]}\varphi \mid \\ &\texttt{[headF]*} \mid \texttt{[headF]}\bot \mid \texttt{[headS]}n \mid \texttt{[headS]}\bot\end{aligned}$$

with $n \in \mathbb{N}$. In both cases, `tailF`, `headF` and respectively `tailS`, `headS` correspond to the situations when `tail` and `head` fail (i.e. yield a result of type `1`) and respectively succeed (i.e. yield a result of type `List` and respectively `Nat`). The satisfaction of such formulae by states a of G-coalgebras $\langle A, \alpha\rangle$ is then defined by:

$$\begin{aligned}
\langle A,\alpha\rangle, a \models \texttt{<tailF>*} &\Leftrightarrow \pi_1(\alpha(a)) = \iota_1(*)\\
\langle A,\alpha\rangle, a \models \texttt{<tailS>}\varphi &\Leftrightarrow \pi_1(\alpha(a)) = \iota_2(a') \text{ and } \langle A,\alpha\rangle, a' \models \varphi\\
\langle A,\alpha\rangle, a \models \texttt{<headF>*} &\Leftrightarrow \pi_2(\alpha(a)) = \iota_1(*)\\
\langle A,\alpha\rangle, a \models \texttt{<headS>}n &\Leftrightarrow \pi_2(\alpha(a)) = \iota_2(n)
\end{aligned}$$

and respectively by:

$$\begin{aligned}
\langle A,\alpha\rangle, a \models \texttt{[tailF]*} &\Leftrightarrow \pi_1(\alpha(a)) = \iota_1(e) \text{ implies } e = *\\
\langle A,\alpha\rangle, a \models \texttt{[tailF]}\bot &\Leftrightarrow \pi_1(\alpha(a)) \notin \iota_1(1)\\
\langle A,\alpha\rangle, a \models \texttt{[tailS]}\varphi &\Leftrightarrow \pi_1(\alpha(a)) = \iota_2(a') \text{ implies } \langle A,\alpha\rangle, a' \models \varphi\\
\langle A,\alpha\rangle, a \models \texttt{[headF]*} &\Leftrightarrow \pi_2(\alpha(a)) = \iota_1(e) \text{ implies } e = *\\
\langle A,\alpha\rangle, a \models \texttt{[headF]}\bot &\Leftrightarrow \pi_2(\alpha(a)) \notin \iota_1(1)\\
\langle A,\alpha\rangle, a \models \texttt{[headS]}n &\Leftrightarrow \pi_2(\alpha(a)) = \iota_2(m) \text{ implies } m = n\\
\langle A,\alpha\rangle, a \models \texttt{[headS]}\bot &\Leftrightarrow \pi_2(\alpha(a)) \notin \iota_2(\mathbb{N})
\end{aligned}$$

while the satisfaction of formulae by G-coalgebras is defined as satisfaction in all the states. In addition, in both approaches, the modal operators satisfy a number of axioms formalising their behaviour. For instance, the fact that `tail` yields a result which is either in $\iota_1(1)$ or in $\iota_2(X)$ is captured by the axiom:

$$\texttt{<tailF>*} \dot{\vee} \texttt{<tailS>}\top$$

in [Röß00], and respectively by the axiom:

$$\texttt{[tailS]}\bot \dot{\vee} \texttt{<tailF>}\bot$$

in [Jac00].

The list invariant (see Example 4.1.29) is now captured by the formula:

$$\texttt{<tailF>*} \leftrightarrow \texttt{<headF>*}$$

using the approach in [Röß00], and respectively by the formula:

$$\texttt{[tailS]}\bot \leftrightarrow \texttt{[headS]}\bot$$

using the approach in [Jac00].

Also, infinite lists are specified by the coequation:

`[Z,L]tail = [Z',L]tail`

using the approach presented here, and by the formulae:

$\neg$`<tailF>*`

and respectively:

`[tailF]`$\bot$

using the approaches in [Röß00] and [Jac00].

On the other hand, finite lists can not be specified by means of coequations, but can be specified by the (infinitary) formula:

(`<tailF>*`) $\vee$ (`<tailS><tailF>*`) $\vee$ (`<tailS><tailS><tailF>*`) $\vee \ldots$

using the approach in [Röß00], and respectively by the (infinitary) formula:

(`[tailS]`$\bot$) $\vee$ (`[tailS][tailS]`$\bot$) $\vee$ (`[tailS][tailS][tailS]`$\bot$) $\vee \ldots$

using the approach in [Jac00].

Finally, alternating lists have finitary presentations by means of coequations (see Example 4.2.16), whereas infinitary formulae are needed to specify alternating lists both in [Röß00] and in [Jac00], with the corresponding formulae being given by:

$$\bigvee_{n \in \mathbb{N}} (\texttt{<tailS><tailS><headS>}n \rightarrow \texttt{<headS>}n)$$

and respectively by:

$$\bigvee_{n \in \mathbb{N}} (\texttt{[headS]}n \rightarrow \texttt{[tailS][tailS][headS]}n)$$

(where the satisfaction of the list invariant is also assumed in the second case).

This example strengthens the affirmation that, while less expressive than modal approaches, equational approaches are able to capture more concisely certain observational properties quantified over state spaces (whereas infinitary sentences are often needed in modal logic approaches to specify such properties).

5 Specification of Objects

The abstract equational specification framework described in Chapter 3 is here instantiated to a formalism for the specification of objects.

Objects are regarded as entities characterised by a state together with an interface providing limited access to this state, with simple objects being used as components for more complex objects. The interface of an object can be used to request certain computations to be performed by the object (with such computations typically resulting in a change of state), or certain information to be supplied by the object with regard to its current state.

The object-oriented programming paradigm translates these ideas into a programming methodology. The notion of *class* is used to define the structure and functionality of a collection of similar objects (its *instances*). The structure of the class instances is determined by a number of *instance variables*, while their functionality is determined by a number of *operations*, usually classified into *constructors* (used to create new instances) and *instance methods* (used to modify existing instances). Moreover, classes are organised into *inheritance hierarchies*, with each class inheriting the structure and functionality provided by its ancestors, and being able to add new structure/functionality, as well as to *override* existing structure/functionality. Each class instance is characterised by its own values for the instance variables, and shares the class functionality with all the other class instances. In addition, each class instance has an *identity*, which distinguishes it from other instances with similar values for the instance variables.

The formalism about to be described concerns the specification of classes, and therefore abstracts away the notion of object identity. This allows an instantiation of the abstract framework described in Section 3.3 to be used for specifying object behaviour and for reasoning about object correctness. The particular instantiation is determined by the form of object interfaces, and involves the use of many-sorted coalgebraic operations in specifying structural properties of classes[1], and respectively of many-sorted algebraic operations in specifying the functionality of classes. Relating the two categories of features then amounts to specifying the changes in structure associated to a particular choice of functionality. In addition, the instantiation reflects the special treatment received by data types in object-oriented languages.

As already noted in Chapter 1, the decision to use coalgebra in specifying the structure of objects and respectively algebra in specifying their functionality is justified by the fact that the attribute values of an object after a method evaluation depend solely on its attribute values prior to the method evaluation. This allows the notion of observational indistinguishability of object states to be defined solely in terms of the object attributes. In this respect, our approach is similar to the one in [HK99, Kur00], where methods are typically regarded as part of the algebraic component and are required to preserve the notion of observational indistinguishability induced by the coalgebraic

[1] The use of many-sorted coalgebra for the observational aspect is further justified by the need to account for the possibility of a choice in what can be observed of an object, e.g. in the presence of inheritance.

component, and differs from the ones in [Jac96c, Jac96a, Jac97, Kur98, Rö ß00, Jac00], where methods also play a part in defining the notion of observational indistinguishability.

By being an instance of the abstract framework presented in Section 3.3, the formalism about to be introduced benefits from the availability of coinductive techniques for proving the equivalence of object-oriented programs, as well as of inductive techniques for proving the satisfaction of state invariants in reachable states. Specifically, proving that two object-oriented programs are observationally indistinguishable can be reduced to exhibiting *some* bisimulation (the largest one is often not the most convenient one) which relates them, whereas proving that a state invariant holds in reachable states can be reduced to exhibiting a subspace of the state space which contains the reachable states (again, the subspace consisting *only* of reachable states is not always the most convenient one) and whose states satisfy the given invariant.

The chapter is structured as follows. Section 5.1 introduces a syntax for the specification of structures whose computational and observational features can be captured within (variants of) many-sorted algebra and respectively many-sorted coalgebra. Sections 5.2 and 5.3 then use this syntax for specifying data and respectively object types, and for reasoning about their correctness. In addition, Section 5.3 illustrates the use of the resulting formalism in specifying inheritance relationships between classes.

5.1 Constraints

If F and G denote the endofunctors associated to a many-sorted signature Γ and respectively a many-sorted cosignature Δ over the same set S of sorts, with T and D as the induced monad and respectively comonad, and if U : Alg(F) $\to$ Set^S and U : Coalg(G) $\to$ Set^S denote the functors taking F-algebras and respectively G-coalgebras to their carrier, then defining natural transformations of types TU $\Rightarrow$ GTU and FDU $\Rightarrow$ DU (inducing liftings of T to G-coalgebras and respectively of D to F-algebras) involves, in each case, defining the results of G-observations on F-programs in terms of particular observations of the program arguments. However, specific observations of the program arguments can only be used in defining the result of observations on the program itself once their actual result type has been determined. The notion of *constraint* allows for this type of case analysis to be performed when defining the above-mentioned liftings. Further intuitions for this notion are provided by the following example.

Example 5.1.1. An object specification of stacks of natural numbers involves visible sorts `1` and `Nat`, a hidden sort `Stack`, *constructors* `empty` :$\to$ `Stack`, `push` : `Stack Nat` $\to$ `Stack` and `pop` : `Stack` $\to$ `Stack`, and *observers* `top` : `Stack` $\to$ `1 Nat` and `rest` : `Stack` $\to$ `1 Stack`[2]. Then, when defining the effect of the constructor `pop` on the observer `top`, a distinction should be made between stacks containing at most one element and stacks containing at least two elements. For, in the first case, the top of the resulting stack is undefined, whereas in the second case the top of the resulting stack is given by the second element of the original stack. The following constraints are used to specify this:

```
[Z,N]top.pop(s) = Z.* if [Z,[Z,N]top]rest.s = Z.z
[Z,N]top.pop(s) = N.n if [Z,[Z,N]top]rest.s = N.n
```

[2]Notions of *signature of constructors* and *cosignature of observers* will be introduced in Section 5.3.

They correspond to the case when the stack contains at most one element (i.e. observing it using `[Z,[Z,N]top]rest` yields a result of type `1`), and respectively to the case when the stack contains at least two elements (i.e. observing it using `[Z,[Z,N]top]rest` yields a result of type `Nat`). In the first case, the resulting stack is empty, and therefore observing it using `[Z,N]top` yields a result of type `1`, whereas in the second case the resulting stack contains at least one element, and hence observing it using `[Z,N]top` yields the same result (of type `Nat`) as when observing the original stack using `[Z,[Z,N]top]rest`. This is captured by the use of the variable `n` both in the condition of the second constraint, where a value is provided for `n`, and in the rhs of this constraint, where the value of `n` is used to define the value of the lhs of the constraint.

Constraints are now formally defined as follows.

Definition 5.1.2. *Let Γ and Δ denote a many-sorted signature and respectively cosignature over the same set S of sorts. A (Γ,Δ)-***constraint** *is a tuple (c,t,Z,t',C), where:*

- *$c \in T_\Delta[\mathcal{C}]_s$ and $t \in T_\Gamma(\mathcal{V})_s$ for some $s \in S$*
- *$Z \in covar(c)_{s'}$ and $t' \in T_\Gamma(\{Y_1,\ldots,Y_n\})_{s'}$ for some $s' \in S$*
- *$C = \{(c_1, X_{j_1}, Z_1, Y_1), \ldots, (c_n, X_{j_n}, Z_n, Y_n)\}$ with $c_i \in T_\Delta[\mathcal{C}_i]_{s_i}$, $X_{j_i} \in var(t)_{s_i}$ for some $s_i \in S$ and $Z_i \in covar(c_i)_{s'_i}$, $Y_i : s'_i$ for some $s'_i \in S$, for $i = 1,\ldots n$.*

(c,t,Z,t',C) is alternatively denoted $c.t = Z.t'$ if $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_n.X_{j_n} = Z_n.Y_n$.

The conditional part C of a constraint is used to extract the values Y_i of the observations c_i on the variables X_{j_i} appearing in t, to be used in t' for defining the value of the observation c on the result yielded by the t.

Instantiating the definition of (F, G)-bialgebras (see Remark 3.3.3) with the endofunctors induced by a many-sorted signature Γ and a many-sorted cosignature Δ over the same set of sorts yields a notion of (Γ,Δ)-*bialgebra*. Next, a notion of satisfaction of (Γ,Δ)-constraints by (Γ,Δ)-*bialgebras* is defined.

For a (Γ,Δ)-bialgebra A, a set $\mathcal{V}$ of variables, a set C of conditions for the variables in $\mathcal{V}$ and an assignment $\theta : \mathcal{V} \to A$, the conditions $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_n.X_{j_n} = Z_n.Y_n$ in C are said to *hold* for θ if and only if $(c_i)_A(\theta(X_{j_i})) = \iota_{Z_i}(a_i)$ for some $a_i \in A_{s_i}$, for $i = 1,\ldots,n$. In this case, one writes $\theta_C : \{Y_1,\ldots,Y_n\} \to A$ for the assignment given by $\theta_C(Y_i) = a_i$ for $i = 1,\ldots,n$.

Definition 5.1.3. *Let Γ and Δ denote a many-sorted signature and respectively cosignature over the same set S of sorts. A (Γ,Δ)-bialgebra A* **satisfies** *a (Γ,Δ)-constraint e of form $c.t = Z.t'$ if $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_n.X_{j_n} = Z_n.Y_n$ (written $A \models_{\Gamma,\Delta} e$) if and only if $c_A(\theta^\#(t)) = \iota_Z(\theta^\#_C(t'))$ for any assignment $\theta : var(t) \to A$ such that C holds for θ (with $\theta^\# : T_\Gamma(var(t)) \to A\restriction_\Gamma$ and $\theta^\#_C : T_\Gamma(\{Y_1,\ldots,Y_n\}) \to A\restriction_\Gamma$ denoting the unique extensions of θ and θ_C to Γ-homomorphisms).*

Suitably-restricted sets of (Γ,Δ)-constraints can be used to specify lifted signatures and cosignatures, with the models of the resulting lifted signatures or cosignatures being in one-to-one correspondence with the (Γ,Δ)-bialgebras satisfying the given constraints. In the following, attention is

restricted to lifted signatures and cosignatures of the form of the ones considered in Sections 3.3.6 and 3.3.7. Such lifted signatures and cosignatures, defined using natural transformations of type $\mathsf{FU} \Rightarrow \mathsf{G}(\mathsf{U} + \mathsf{FU})$ and respectively $\mathsf{F}(\mathsf{U} \times \mathsf{GU}) \Rightarrow \mathsf{GU}$, will be shown to arise from sets of (Γ, Δ)-constraints of form:

$$[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t \text{ if } c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$$

subject to the additional requirement that $t \in T_\Gamma(\{Y_1, \ldots, Y_k\})$ contains at most one Γ-symbol, and respectively that each of $c_1, \ldots, c_n$ contains at most one Δ-symbol. (In each case, the sets of constraints defining the natural transformations are subject to further restrictions.)

In writing constraints, it is often the case that some of the variables appearing in the lhs need to be indirectly passed as arguments to the term in the rhs, namely through conditions of form $Z.X = Z.Y$. For simplicity, in forthcoming examples such conditions will be omitted from the conditional part of constraints, and the original variables will also be used in the rhs of constraints.

Example 5.1.4. The constraints used to specify the behaviour of stacks (see Example 5.1.1) are as follows:

```
[Z,N]top.empty = Z.*
[Z,S]rest.empty = Z.*

[Z,N]top.push(s,n) = N.n
[Z,S]rest.push(s,n) = S.s

[Z,N]top.pop(s) = Z.* if [Z,[Z,N]top]rest.s = Z.z
[Z,N]top.pop(s) = N.n if [Z,[Z,N]top]rest.s = N.n
[Z,S]rest.pop(s) = Z.* if [Z,[Z,S]rest]rest.s = Z.z
[Z,S]rest.pop(s) = S.s' if [Z,[Z,S]rest]rest.s = S.s'
```

Note that, in the above, the two constraints defining `push` are shorthands for:

```
[Z,N]top.push(s,n) = N.n' if N.n = N.n'
[Z,S]rest.push(s,n) = S.s' if S.s = S.s'
```

The deduction calculi for equations and coequations (see Sections 2.3.1 and 4.1.6) can be suitably combined into a deduction calculus for constraints. This calculus is sound for the satisfaction of constraints by bialgebras. However, it is not complete. Nevertheless, provided that *sufficient constraints are given in the premises*, and that *sufficient assumptions are made about the variables involved*[3], its rules are sufficient for rewriting particular observations of given computations into computations yielding the same result.

[3] The precise meaning of these requirements will become clear later in the chapter (see Propositions 5.2.11 and 5.3.17).

The deduction calculus for constraints consists of the following rules:

$$[\,\textbf{base}\,]\quad \frac{}{E \vdash e}\ \ e \in E$$

$$[\,\textbf{base}_1\,]\quad \frac{}{E \vdash Z.t = Z.t(\overline{Y}/\overline{X}) \text{ if } Z_1.X_1 = Z_1.Y_1, \ldots, Z_m.X_m = Z_m.Y_m}$$

$$[\,\textbf{base}_2\,]\quad \frac{}{E \vdash c.X = Z.Y \text{ if } c.X = Z.Y}$$

$$[\,\textbf{substitution}_1\,]\quad \frac{E \vdash c.t = Z.t' \text{ if } c_1.X_{j_1} = Z_1.Y_1, \ldots, c_n.X_{j_n} = Z_n.Y_n \quad E \vdash c_1.t_{j_1} = Z_1.t'_1 \text{ if } C_1, \ldots, E \vdash c_n.t_{j_n} = Z_n.t'_n \text{ if } C_n}{E \vdash c.t(\overline{t}/\overline{X}) = Z.t'(\overline{t'}/\overline{Y}) \text{ if } C_1, \ldots, C_n}$$

$$[\,\textbf{substitution}_2\,]\quad \frac{E \vdash c.t = Z'_i.t' \text{ if } c_1.X_{j_1} = Z_1.Y_1, \ldots, c_m.X_{j_m} = Z_m.Y_m \quad E \vdash c'_i.t' = Z.t'' \text{ if } c''_1.Y_{k_1} = Z''_1.Y''_1, \ldots, c''_n.Y_{k_n} = Z''_n.Y''_n}{\begin{array}{r} E \vdash [\overline{c'}/\overline{Z'}]c.t = Z.t'' \text{ if } (c_i.X_{j_i} = Z_i.Y_i)_{i=1,\ldots,m}, \\ ([\ldots c''_l/Z_{k_l}\ldots]c_{k_l}.X_{j_{k_l}} = Z''_l.Y''_l)_{l=1,\ldots,n} \end{array}}$$

Theorem 5.1.5 (Soundness). *Deduction using the above rules is sound for the satisfaction of* (Γ, Δ)*-constraints by* (Γ, Δ)*-bialgebras.*

Proof (sketch). The proof uses standard properties of equality. □

Remark 5.1.6. Equations and coequations can sometimes be used as alternatives for (sets of) constraints. For, a Γ-equation $(\forall \mathcal{V})\ l = r$ with $var(r) \subseteq var(l) = \mathcal{V}$ holds in the underlying Γ-algebra of a (Γ, Δ)-bialgebra if and only if the (Γ, Δ)-constraint:

$$Z.l = Z.r$$

holds in the given (Γ, Δ)-bialgebra. Also, a Δ-coequation $c = c'$ if $(c_1, \mathcal{C}_1), \ldots, (c_n, \mathcal{C}_n)$ with $covar(c') \subseteq covar(c)$ holds in the underlying Δ-coalgebra of a (Γ, Δ)-bialgebra if and only if each of the (Γ, Δ)-constraints:

$$c.X = Z.Y \text{ if } c_1.X = Z_1.Y_1, \ldots, c_n.X = Z_n.Y_n, c'.X = Z.Y$$

with $Z_i \in \mathcal{C}_i$ for $i = 1, \ldots, n$ and $Z \in covar(c')$ holds in the given (Γ, Δ)-bialgebra. This observation will be used later in this chapter to provide alternative definitions for *derived constructors* and *observers* (with derived constructors and observers defining instances of the abstract notions of constructor and respectively observer).

5.2 Specifying Data

Instantiating the approach outlined in Section 3.3.7 to endofunctors of the form of the ones induced by many-sorted signatures and cosignatures yields a formalism for the specification of data types. This formalism constitutes an extension of the standard algebraic approach to the specification of data types, rather than an alternative to it. The extension involves the specification of a number of data type observers, and is based on a distinction between *generators* and *defined functions* amongst the algebraic operations associated with a data type.

5.2.1 Syntax and Semantics

The next two examples give an outline of the approach, and at the same time provide some intuitions for the notions introduced later in this section.

Example 5.2.1. Boolean values are specified algebraically using a sort `Bool` and operation symbols `true` : $\rightarrow$ `Bool` and `false` : $\rightarrow$ `Bool`. These operation symbols define the *generators* for the type `Bool`. Any boolean value can be denoted by a ground term over these operation symbols. Additional algebraic operations on booleans can then be defined by induction over these ground terms. In particular, the standard boolean operators, i.e. `not` : `Bool` $\rightarrow$ `Bool`, `and` : `Bool Bool` $\rightarrow$ `Bool` and `or` : `Bool Bool` $\rightarrow$ `Bool` can be defined using equations:

```
not(true) = false
not(false) = true
and(false,false) = false
and(false,true) = false
...
```

It is important to note that such definitions refer to the initial algebra of the signature of generators, rather than to an arbitrary algebra of this signature.

One can also give a coalgebraic specification of boolean values, namely by using sorts `1` and `Bool` together with an operation symbol `?` : `Bool` $\rightarrow 1 + 1$. The sort `1` denotes a one-element type, and consequently has no operation symbols associated to it. The operation denoted by `?` is used to observe the truth value of a boolean value (true or false, depending on whether the result of `?` is in $\iota_1(1)$ or in $\iota_2(1)$).

The algebraic specification of the type of booleans defines the values as well as the functionality associated to this type (by means of generators and respectively defined functions), while the coalgebraic specification of booleans defines ways of observing boolean values. The two specifications can be combined using the approach in Section 3.3.7. However, due to the nature of defined functions (which are only defined on the initial algebra of the signature of generators), it is only the generators that should to be considered when defining the lifting required by this approach. The additional structure provided by the defined functions on the initial coalgebra of the resulting lifted cosignature can only be inherited by the coalgebra used as denotation for this lifted cosignature (given by the quotient of the initial coalgebra of the lifted cosignature by bisimilarity), and only provided that the defined functions preserve bisimilarity. Nevertheless, in most cases (including the one here), preservation of bisimilarity by the defined functions is a consequence of the notion of bisimilarity on the initial coalgebra of the lifted cosignature considered being given by the equality relation. The availability of the additional structure provided by the defined functions on the coalgebras used as denotations for lifted cosignatures will prove crucial in certain situations (see e.g. Example 5.3.23).

Now let $* : \rightarrow$ `1` denote a generator for the type `1`, and let $\mathsf{F}, \mathsf{G} : \mathsf{Set}^{\{1,\mathtt{Bool}\}} \rightarrow \mathsf{Set}^{\{1,\mathtt{Bool}\}}$ denote the endofunctors induced by the many-sorted signature consisting of $*$, `true` and `false`, and respectively

by the many-sorted cosignature consisting of `?`. The components of F and G are therefore given by:

$$(\mathsf{F}X)_{\mathtt{1}} = 1$$
$$(\mathsf{F}X)_{\mathtt{Bool}} = 1 + 1$$
$$(\mathsf{G}X)_{\mathtt{1}} = 1$$
$$(\mathsf{G}X)_{\mathtt{Bool}} = X_{\mathtt{1}} + X_{\mathtt{1}}$$

(where `1` is used to denote both a sort, if it appears as a subscript, and the empty product in Set, otherwise). Finally, let $\mathsf{U} : \mathsf{Alg}(\mathsf{Set}^{\{\mathtt{1},\mathtt{Bool}\}}, \mathsf{F}) \to \mathsf{Set}^{\{\mathtt{1},\mathtt{Bool}\}}$ denote the functor taking $(\mathsf{Set}^{\{\mathtt{1},\mathtt{Bool}\}}, \mathsf{F})$-algebras to their carrier. Then, the following constraints induce a natural transformation $\rho : \mathsf{F}(\mathsf{U} \times \mathsf{GU}) \Rightarrow \mathsf{GU}$:

```
[T,F]?.true = T.*
[T,F]?.false = F.*
```

Specifically, the components of ρ are given by:

$$(\rho_A)_{\mathtt{1}}(*) = *$$
$$(\rho_A)_{\mathtt{Bool}}(\iota_1(*)) = \iota_1(*_A)$$
$$(\rho_A)_{\mathtt{Bool}}(\iota_2(*)) = \iota_2(*_A)$$

for each $(\mathsf{Set}^{\{\mathtt{1},\mathtt{Bool}\}}, \mathsf{F})$-algebra A (where $*$ denotes the unique element of 1, while $*_A$ denotes the interpretation provided by A to the operation symbol $\mathtt{*} : \to \mathtt{1}$).

Example 5.2.2. Similarly, natural numbers are specified using sorts `1` and `Nat`, algebraic operation symbols $\mathtt{*} : \to \mathtt{1}$, $\mathtt{0} : \to \mathtt{Nat}$ and $\mathtt{s} : \mathtt{Nat} \to \mathtt{Nat}$, a coalgebraic operation symbol $\mathtt{p} : \mathtt{Nat} \to \mathtt{1} + \mathtt{Nat}$ (with `s` and `p` denoting the successor and respectively predecessor functions), and constraints:

```
[Z,N]p.0 = Z.*
[Z,N]p.s(x) = N.x
```

Now let $\mathsf{F}, \mathsf{G} : \mathsf{Set}^{\{\mathtt{1},\mathtt{Nat}\}} \to \mathsf{Set}^{\{\mathtt{1},\mathtt{Nat}\}}$ denote the endofunctors induced by the many-sorted signature consisting of `*`, `0` and `s`, and respectively by the many-sorted cosignature consisting of `p`. The components of F and G are given by:

$$(\mathsf{F}X)_{\mathtt{1}} = 1$$
$$(\mathsf{F}X)_{\mathtt{Nat}} = 1 + X_{\mathtt{Nat}}$$
$$(\mathsf{G}X)_{\mathtt{1}} = 1$$
$$(\mathsf{G}X)_{\mathtt{Nat}} = X_{\mathtt{1}} + X_{\mathtt{Nat}}$$

Also, let $\mathsf{U} : \mathsf{Alg}(\mathsf{Set}^{\{\mathtt{1},\mathtt{Nat}\}}, \mathsf{F}) \to \mathsf{Set}^{\{\mathtt{1},\mathtt{Nat}\}}$ denote the functor taking $(\mathsf{Set}^{\{\mathtt{1},\mathtt{Nat}\}}, \mathsf{F})$-algebras to their carrier. Then, the two constraints above induce a natural transformation $\rho : \mathsf{F}(\mathsf{U} \times \mathsf{GU}) \Rightarrow \mathsf{GU}$ whose components are given by:

$$(\rho_A)_{\mathtt{1}}(*) = *$$
$$(\rho_A)_{\mathtt{Nat}}(\iota_1(*)) = \iota_1(*_A)$$
$$(\rho_A)_{\mathtt{Nat}}(\iota_2(\langle a, a' \rangle)) = \iota_2(a)$$

for each (Set$^{\{1,\mathtt{Nat}\}}$, F)-algebra A.

In addition, one can use induction to equationally define other algebraic operations on natural numbers. For instance, natural number addition can be defined using an algebraic operation symbol + : Nat Nat → Nat, subject to the following equations:

```
0+y = y
s(x)+y = s(x+y)
```

According to the remarks in Example 5.2.1, the correctness of the above definition amounts to the preservation of bisimilarity by the resulting interpretation of + in the initial coalgebra of the lifted cosignature of natural numbers. As in Example 5.2.1, this is a consequence of the fact that, in this coalgebra (whose carrier is isomorphic to $(\{*\}, \mathbb{N})$), bisimilarity coincides with equality.

We conclude this example by noting that, unlike in Example 5.2.1, where the carriers of the initial algebra of the underlying signature and respectively of the final coalgebra of the underlying cosignature were both isomorphic to $(\{*\}, \{\mathtt{t}, \mathtt{f}\})$, here these carriers are not isomorphic to each other. (The former is isomorphic to $(\{*\}, \mathbb{N})$, whereas the latter is isomorphic to $(\{*\}, \mathbb{N} \cup \{\infty\})$.)

We now introduce some terminology.

Definition 5.2.3. *Let Ξ denotes a many-sorted cosignature with sort set S. A* **one-step Ξ-observation** *of sort $s \in S$ is an s-sorted Ξ-coterm containing at most one Ξ-symbol, while a* **one-step Ξ-behaviour** *for the sort s is a tuple $(n_\xi)_{\xi \in \Xi_s}$, with $n_\xi \in \{1, \ldots, n\}$ for $\xi \in \Xi_{s,s_1 \ldots s_n}$. A condition of form $c.X = Z.Y$ with $X : s$, c a one-step Ξ-observation of sort s, $Y : s'$ and $Z \in covar(c)_{s'}$ is said to* **hold** *for a one-step Ξ-behaviour $(n_\xi)_{\xi \in \Xi_s}$ for the sort s if and only if, whenever c is of form $[Z_1, \ldots, Z_n]\xi$ and $n_\xi = l$, it follows that $Z = Z_l$. Also, a Ξ-***behaviour** *for the sort s is an element $f \in F_s$ of the final Ξ-coalgebra. A condition of form $c.X = Z.Y$ with $X : s$, $c \in T_\Delta[\mathcal{C}]_s$, $Y : s'$ and $Z \in covar(c)_{s'}$ is said to* **hold** *for a Ξ-behaviour f for the sort s if and only if $c_F(f) \in \iota_Z(F_{s'})$.*

One-step Ξ-behaviours for the sort s specify, for each operation symbol ξ on s, an evaluation path for the Ξ-coterm $[Z_1, \ldots, Z_n]\xi$ (where $\xi : s \to s_1 \ldots s_n$).

The sets of constraints used in Examples 5.2.1 and 5.2.2 have the property that, for each pair consisting of an algebraic operation symbol and a coalgebraic operation symbol of the same sort, there exists precisely one constraint defining the effect of evaluating the coalgebraic operation on the result yielded by the algebraic operation. It is this property that guarantees the existence and uniqueness of the natural transformations ρ. The next definition exploits this observation.

Definition 5.2.4. *A* **data cosignature** *is a tuple (S, Ψ, Ξ, E), with Ψ and Ξ denoting a many-sorted signature and respectively cosignature with sort set S, and with E denoting a set of (Ψ, Ξ)-constraints additionally satisfying:*

1. *All the constraints in E are of form:*

$$[Z_1, \ldots, Z_n]\xi.\psi(X_1, \ldots, X_m) = Z.t \text{ if } c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$$

with $c_1, \ldots, c_n$ one-step Ξ-observations.

2. *For each $s \in S$, $\psi \in \Psi_{s_1 \ldots s_m, s}$, $\xi \in \Xi_{s, s'_1 \ldots s'_n}$ and each choice of one-step Ξ-behaviours for the sorts $s_1, \ldots, s_m$, there exists $[Z_1, \ldots, Z_n]\xi.\psi(X_1, \ldots, X_m) = Z.t$ if C in E such that C holds for the chosen one-step Ξ-behaviours for $X_1, \ldots, X_m$.*
3. *If $[Z_1, \ldots, Z_n]\xi.\psi(X_1, \ldots, X_m) = Z_{k_i}.t_i$ if C_i in E for $i = 1, 2$ are such that C_1 and C_2 hold for some choice of one-step Ξ-behaviours for $X_1, \ldots, X_m$, then $Z_{k_1} = Z_{k_2}$ and $\bar{t}_1 = \bar{t}_2$ (where, if $c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$ are the conditions appearing in $C_1 \cup C_2$, then $\bar{t}_1$ and $\bar{t}_2$ denote the Ψ-terms obtained from t_1 and t_2 by identifying any two variables Y_i and Y_l with $X_{j_i} = X_{j_l}$ and $c_i = c_l$[4]).*

If $\mathsf{F} : \mathsf{Set}^S \to \mathsf{Set}^S$ and $\mathsf{G} : \mathsf{Set}^S \to \mathsf{Set}^S$ denote the endofunctors associated to Ψ and Ξ, if D denotes the comonad induced by G, and if $\mathsf{U} : \mathsf{Alg}(\mathsf{Set}^S, \mathsf{F}) \to \mathsf{Set}^S$ denotes the functor taking $(\mathsf{Set}^S, \mathsf{F})$-algebras to their carrier, then data cosignatures (S, Ψ, Ξ, E) induce lifted cosignatures $(\mathsf{Set}^S, \mathsf{F}, \mathsf{G}, \sigma)$, with $\sigma : \mathsf{FDU} \Rightarrow \mathsf{DU}$ being the natural transformation induced by the natural transformation $\rho : \mathsf{F}(\mathsf{U} \times \mathsf{GU}) \Rightarrow \mathsf{GU}$ (see Sections 3.3.7 and 3.3.6) whose components are given by:

$$\pi_\xi(\rho_{A,s}(\iota_\psi(\langle a_i, (a^i_\xi)_{\xi \in \Xi_{s_i}} \rangle_{i=1,\ldots,m}))) = \iota_Z(t_A(y_1, \ldots, y_k))$$

for each Ψ-algebra A, $s \in S$, $\psi \in \Psi_{s_1 \ldots s_m, s}$, $\xi \in \Xi_{s, s'_1 \ldots s'_n}$ and $\langle a_i, (a^i_\xi)_{\xi \in \Xi_{s_i}} \rangle \in A_{s_i} \times (\mathsf{G}A)_{s_i}$ with $i = 1, \ldots, m$, where the constraint $[Z_1, \ldots, Z_n]\xi.\psi(X_1, \ldots, X_m) = Z.t$ if $c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$ in E is such that its conditions hold for the one-step Ξ-behaviours induced by $(a^i_\xi)_{\xi \in \Xi_{s_i}}$ with $i = 1, \ldots, m$[5], and where $y_1, \ldots, y_k$ are given by:

$$y_i = \begin{cases} a_{j_i} & \text{if } \; c_i = Z_i \\ a^{j_i}_\xi & \text{if } \; c_i = [\overline{Z}]\xi \end{cases}, \quad i \in \{1, \ldots, k\}$$

Moreover, the coalgebras of the induced lifted cosignature are in one-to-one correspondence with the (Ψ, Ξ)-bialgebras satisfying E. This is a consequence of the definition of ρ, and of the existence of a characterisation of $(\mathsf{Set}^S, \mathsf{F}, \mathsf{G}, \sigma)$-coalgebras as suitably-restricted (F, G)-bialgebras dual to the one outlined in Remark 3.3.26.

Finally, Ψ-equations and Ξ-coequations define F-equations and respectively G-coequations (see Example 3.2.13 and Remark 4.1.30).

According to the remarks in Section 3.3.7, suitable denotations for data cosignatures are provided by the codomains of the quotients of the unique homomorphisms from the initial to the final coalgebras of such cosignatures.

Example 5.2.5. The many-sorted signature, many-sorted cosignature and set of constraints in Examples 5.2.1 and respectively 5.2.2 define data cosignatures. The carriers of the coalgebras used as denotations for these cosignatures are isomorphic to $(\{*\}, \{\mathtt{t}, \mathtt{f}\})$ and respectively $(\{*\}, \mathbb{N})$. (In both cases, bisimilarity on the initial coalgebra is given by the equality relation.)

A notion of *data cosignature morphism* (inducing a lifted cosignature morphism) can also be defined.

[4] In this case, Y_i and Y_l refer to the same value.

[5] Conditions 2 and 3 guarantee the existence of such a constraint and respectively the independence of the definition of ρ on a particular choice of constraint.

Definition 5.2.6. *Let (S, Ψ, Ξ, E) and (S', Ψ', Ξ', E') denote data cosignatures. A* **data cosignature morphism** *from (S, Ψ, Ξ, E) to (S', Ψ', Ξ', E') is a pair (ϕ, θ), with $\phi : (S, \Psi) \to (S', \Psi')$ a many-sorted signature morphism and $\theta : (S, \Xi) \to (S', \Xi')$ a many-sorted cosignature morphism, additionally satisfying:*

1. *$\phi{\restriction_S} = \theta{\restriction_S}$*
2. *the functions $\theta{\restriction_{\Xi_s}} : \Xi_s \to \Xi'_{\theta(s)}$ with $s \in S$ are bijections*
3. *$(\phi, \theta)(e) \in E'$ for each $e \in E$ (where, for $e \in E$, $(\phi, \theta)(e)$ denotes the pointwise translation of e along (ϕ, θ)).*

If, in addition:

4. *the functions $\phi{\restriction_{\Psi_s}} : \Psi_s \to \Psi'_{\phi(s)}$ with $s \in S$ are surjections*

then (ϕ, θ) is called **horizontal**.

If $(\mathsf{Set}^S, \mathsf{F}, \mathsf{G}, \sigma)$ and $(\mathsf{Set}^{S'}, \mathsf{F}', \mathsf{G}', \sigma')$ denote the lifted cosignatures induced by the data cosignatures (S, Ψ, Ξ, E) and respectively (S', Ψ', Ξ', E'), then (horizontal) data cosignature morphisms $(\phi, \theta) : (S, \Psi, \Xi, E) \to (S', \Psi', \Xi', E')$ induce (horizontal) lifted cosignature morphisms $(\mathsf{U}, \xi, \eta) : (\mathsf{Set}^S, \mathsf{F}, \mathsf{G}, \sigma) \to (\mathsf{Set}^{S'}, \mathsf{F}', \mathsf{G}', \sigma')$, with $(\mathsf{U}, \xi) : (\mathsf{Set}^S, \mathsf{F}) \to (\mathsf{Set}^{S'}, \mathsf{F}')$ denoting the abstract signature morphism induced by the many-sorted signature morphism ϕ (see Example 3.2.20), and with $(\mathsf{U}, \eta) : (\mathsf{Set}^S, \mathsf{G}) \to (\mathsf{Set}^{S'}, \mathsf{G}')$ denoting the abstract cosignature morphism induced by the many-sorted cosignature morphism θ (see Remark 4.1.38). The second requirement in Definition 5.2.6 ensures that η is a natural isomorphism, the third requirement guarantees that ρ, ρ', ξ and η satisfy (the dual of) the compatibility condition required in Section 3.3.6, while the fourth requirement ensures that ξ is a natural epimorphism.

Example 5.2.7. The inclusion of the data cosignature of booleans (see Example 5.2.1) into the joint data cosignature of booleans and natural numbers (consisting, in addition to the sorts, operation symbols and constraints given in Example 5.2.1, of the sorts, operation symbols and constraints given in Example 5.2.2) defines a horizontal data cosignature morphism.

5.2.2 Correctness Proofs

Suitably instantiating the abstract notions of satisfaction given by the dual of Definition 3.3.12 yields the following notions of satisfaction of equations up to bisimulation and respectively of coequations up to reachability by coalgebras of data cosignatures.

Definition 5.2.8. *Let (S, Ψ, Ξ, E) denote a data cosignature. An (S, Ψ, Ξ, E)-coalgebra A* **satisfies** *a Ψ-equation of form $(\forall\, \mathcal{V})\ l = r$* **up to bisimulation** *if and only if $\theta^\#(l) \sim_A \theta^\#(r)$ holds for any assignment $\theta : \mathcal{V} \to A$, with $\sim_A$ denoting Ξ-bisimilarity (see Proposition 4.1.19) on $A{\restriction_\Xi}$. Also, A* **satisfies** *a Ξ-coequation of form $c = c'$ if $(c_1, \mathcal{C}_1), \ldots, (c_n, \mathcal{C}_n)$* **up to reachability** *if and only if, for any $t \in T_\Psi$, $c_A(t_A) = c'_A(t_A)$ holds whenever $(c_i)_A(t_A) \in \iota_{Z_i}(A_{s_i})$ for some $s_i \in S$ and some $Z_i \in \mathcal{C}_{i,s_i}$, for $i = 1, \ldots, n$.*

According to the dual of Theorem 3.3.17 and respectively of Theorem 3.3.19, the notion of data cosignature morphism yields an institution w.r.t. the satisfaction of equations up to bisimulation, while the notion of horizontal data cosignature morphism yields an institution w.r.t. the satisfaction of coequations up to reachability. Moreover, since Set^S is regular (see Example 2.1.47), it follows by the remarks in Section 3.3.7 that the coalgebras used as denotations for data cosignatures satisfy precisely those equations (respectively coequations) which are satisfied up to bisimulation (up to reachability) by the initial (final) coalgebras.

It should be noted that, in order for the above notion of satisfaction of equations up to bisimulation not to result in the satisfaction of unwanted equations, the notion of bisimulation induced by the underlying cosignature should be sufficiently fine. For instance, the absence of a coalgebraic operation for boolean values would result in the coalgebra used as denotation for the resulting data cosignature identifying `true` and `false` (as, in this case, the notion of bisimulation induced by the underlying cosignature would not distinguish any two boolean values).

The remaining of this section concerns the formulation and proof of correctness properties for the behaviours specified by data cosignatures.

To facilitate the formulation of correctness properties, *derived* Ξ*-observers* $\xi : s \to s_1 \dots s_n$ can be specified using $(\emptyset, \Xi \cup \{\xi\})$-constraints E_ξ of form:

$$[Z_1, \dots, Z_n]\xi.X = Z.Y \text{ if } C$$

with C only containing Ξ-symbols, additionally satisfying:

1. For any choice of a Ξ-behaviour for X, E_ξ contains a constraint whose condition holds for that Ξ-behaviour.
2. Whenever $[Z_1, \dots, Z_n]\xi.X = Z_{k_i}.Y_{l_i}$ if C_i with $i = 1, 2$ in E_ξ are such that C_1 and C_2 hold for some choice of a Ξ-behaviour for X, it follows that $Z_{k_1} = Z_{k_2}$, and that the conditions defining Y_{l_1} and Y_{l_2} are the same up to a renaming of the covariables.

Remark 5.2.9. Since the satisfaction of Ξ-coequations of form $c = c'$ if C with $covar(c) \subseteq covar(c')$ is equivalent to the satisfaction of sets of constraints of the above form (see Remark 5.1.6), derived Ξ-observers ξ can alternatively be specified using sets of $\Xi \cup \{\xi\}$-coequations of form:

$$[Z_1, \dots, Z_n]\xi = c \text{ if } C$$

with $covar(c) \subseteq \{Z_1, \dots, Z_n\}$, subject to constraints similar to 1 and 2 above[6]. Furthermore, previously-defined derived Ξ-observers can be used in coequations defining new derived Ξ-observers.

The conditions in the definition of a derived observer ξ ensure that any Ξ-coalgebra structure on $A \in |\mathsf{Set}^S|$ can be uniquely extended to a $\Xi \cup \{\xi\}$-coalgebra structure on A which satisfies the constraints (coequations) defining ξ. Moreover, Ξ-coalgebra homomorphisms also preserve the

[6] The reason for defining derived observers using constraints as opposed to coequations in the first place will become clear when similar definitions will be given for observers on object states. (In the case of objects, the presence of a fixed data universe carrying both algebraic and coalgebraic structure will result in constraints being more expressive than coequations in defining derived observers.)

structure given by the derived observers. As a result, derived observers induce (Set^S, G)-observers, and therefore can be used in formalising correctness properties by means of coequations (see also Remark 3.1.24).

Remark 5.2.10. If Ξ^{d} denotes a set of derived Ξ-observers, and if E_Ξ^{d} denotes the set of $(\emptyset, \Xi \cup \Xi^{\mathsf{d}})$-constraints defining them, then arbitrary $\Xi \cup \Xi^{\mathsf{d}}$-observations can be expressed in terms of Ξ-observers only using the constraints in E_Ξ^{d}. That is, for any $c \in T_{\Xi \cup \Xi^{\mathsf{d}}}[\mathcal{C}]_s$ and any choice of a Ξ-behaviour for the sort s, $E_\Xi^{\mathsf{d}} \vdash c.X = Z.Y$ if C, for some conditions C which only contain Ξ-symbols and which hold for the chosen Ξ-behaviour for s. This follows by structural induction on c. First, if c is a covariable, the conclusion follows immediately by **base$_2$**. Also, if $c = [c_1, \ldots, c_n]\xi$, depending on whether $\xi \in \Xi$ or $\xi \in \Xi^{\mathsf{d}}$, it follows by **base$_2$** and respectively by the definition of ξ that $E_\Xi^{\mathsf{d}} \vdash [Z_1, \ldots, Z_n]\xi.X = Z_i.Y$ if C, with C holding for the chosen Ξ-behaviour for s. In both cases, the induction hypothesis yields $E_\Xi^{\mathsf{d}} \vdash c_i.Y = Z.Y'$ if C', with C' holding for the Ξ-behaviour for (the sort of) Y resulting from the chosen Ξ-behaviour for s together with the condition defining Y. The conclusion then follows by **substitution$_2$**.

According to the remarks in Section 3.3.1 (see also Section 3.3.7), proofs of satisfaction of equations up to bisimulation by coalgebras of data cosignatures can be reduced to defining bisimulation relations which relate the lhs and rhs of the given equations, while proofs of satisfaction of coequations up to reachability by coalgebras of data cosignatures can be reduced to defining subalgebras whose states satisfy the given coequations. In both cases, this involves determining the values of certain observations on the results yielded by certain computations. The deduction calculus for constraints can be successfully used in this sense, due to the following result.

Proposition 5.2.11. *Let (S, Ψ, Ξ, E) denotes a data cosignature. Then, the following hold:*

1. *For each $t \in T_\Psi(\{X_1, \ldots, X_m\})_s$, each $\xi \in \Xi_{s,s_1 \ldots s_n}$ and each choice of one-step Ξ-behaviours for the sorts of $X_1, \ldots, X_m$, there exist $t' \in T_\Psi(\{Y_1, \ldots, Y_k\})$, $Z \in \{Z_1, \ldots, Z_n\}$ and conditions C of form $c_1.X_{j_1} = Z_1'.Y_1, \ldots, c_k.X_{j_k} = Z_k'.Y_k$ with c_i one-step Ξ-observations, such that C holds for the chosen one-step Ξ-behaviours for $X_1, \ldots, X_m$, and such that $E \vdash [Z_1, \ldots, Z_n]\xi.t = Z.t'$ if C.*
2. *For each $t \in T_\Psi(\{X_1, \ldots, X_m\})_s$, each $c \in T_\Xi[\mathcal{C}]_s$ and each choice of Ξ-behaviours for the sorts of $X_1, \ldots, X_m$, there exist $t' \in T_\Psi(\{Y_1, \ldots, Y_k\})$, $Z \in covar(c)$ and conditions C of form $c_1.X_{j_1} = Z_1'.Y_1, \ldots, c_k.X_{j_k} = Z_k'.Y_k$, such that C holds for the chosen Ξ-behaviours for $X_1, \ldots, X_m$, and such that $E \vdash c.t = Z.t'$ if C.*

That is, any (one-step) Ξ-observation of any Ψ-computation rewrites to a Ψ-computation, provided that sufficiently-many assumptions are made about the (one-step) Ξ-behaviours of the variables appearing in the term used to denote the initial Ψ-computation.

Proof. The first statement follows by structural induction on t. First, if t is a variable X, then **base$_2$** yields $E \vdash [Z_1, \ldots, Z_n]\xi.X = Z.Y$ if $[Z_1, \ldots, Z_n]\xi.X = Z.Y$, for each $\xi \in \Xi$ appropriate for X. Next, if t is of form $\psi(t_1, \ldots, t_l)$ with $t_1, \ldots, t_l \in T_\Psi(\{X_1, \ldots, X_m\})$, then the induction hypothesis yields, for each $i \in \{1, \ldots, l\}$ and each $\xi \in \Xi$ appropriate for t_i, a (Ψ, Ξ)-constraint e of

form $[\overline{Z}]\xi.t_i = Z.t'$ if C such that C holds for the chosen one-step Ξ-behaviours for $X_1, \ldots, X_m$, and such that $E \vdash e$. These constraints uniquely determine some one-step Ξ-behaviours for (the sorts of) $t_1, \ldots, t_l$. The hypothesis then yields, for each $\xi \in \Xi$ appropriate for t, a (Ψ, Ξ)-constraint of form $[Z_1, \ldots, Z_n]\xi.\psi(X_1, \ldots, X_l) = Z.t'$ if $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_k.X_{j_k} = Z_k.Y_k$ in E, whose conditions hold for the previously-obtained one-step Ξ-behaviours for (the sorts of) $X_1, \ldots, X_l$. Then, **substitution$_1$** yields $E \vdash [Z_1, \ldots, Z_n]\xi.\psi(t_1, \ldots, t_l) = Z.t'(t'_1/Y_1, \ldots, t'_k/Y_k)$ if $C_1, \ldots, C_k$, with t'_i and C_i being given by $E \vdash c_i.t_{j_i} = Z_i.t'_i$ if C_i, for $i = 1, \ldots, k$[7], and with $C_1, \ldots, C_k$ holding for the chosen one-step Ξ-behaviours for $X_1, \ldots, X_m$.

The second statement follows by structural induction on c. First, if c is a covariable Z, then **base$_1$** yields $E \vdash Z.t = Z.t(\overline{Y}/\overline{X})$ if $(Z_i.X_i = Z_i.Y_i)_{i=1,\ldots,m}$. Next, if c is of form $[c_1, \ldots, c_n]\xi$, then 1 yields, for each $t \in T_\Psi(\{X_1, \ldots, X_m\})$ and each choice of Ξ-behaviours for $X_1, \ldots, X_m$, a (Ψ, Ξ)-constraint e of form $[Z_1, \ldots, Z_n]\xi.t = Z_i.t'$ if C, with C of form $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_k.X_{j_k} = Z_k.Y_k$ holding for (the one-step Ξ-behaviours induced by) the chosen Ξ-behaviours for $X_1, \ldots, X_m$, such that $E \vdash e$. The choice of Ξ-behaviours for $X_1, \ldots, X_m$ together with the conditions defining $Y_1, \ldots, Y_k$ uniquely determine a choice of Ξ-behaviours for $Y_1, \ldots, Y_k$. Applying the induction hypothesis to t', c_i and the previously-obtained Ξ-behaviours for $Y_1, \ldots, Y_k$ yields a (Ψ, Ξ)-constraint e' of form $c_i.t' = Z.t''$ if $c'_1.Y_{k_1} = Z'_1.Y'_1, \ldots, c'_l.Y_{k_l} = Z'_l.Y'_l$ whose conditions hold for these Ξ-behaviours for $Y_1, \ldots, Y_k$, and which is such that $E \vdash e'$. Then, **substitution$_2$** yields $E \vdash [c_1, \ldots, c_n]\xi.t = Z.t''$ if $C, ([\ldots c'_i/Z_{k_i} \ldots]c_{k_i}.X_{j_{k_i}} = Z'_i.Y'_i)_{i=1,\ldots,l})$. □

Remark 5.2.12. In the presence of derived Ξ-observers defined using constraints E^{d}_Ξ, the second statement generalises to coterms c also containing derived observers. (In this case, deduction from E is replaced by deduction from $E \cup E^{\mathsf{d}}_\Xi$.) This follows by **substitution$_1$** together with the observation in Remark 5.2.10.

Example 5.2.13. Consider the specification of natural numbers given in Example 5.2.2, enriched with a coalgebraic operation symbol `!` : `Nat` $\to$ `1` defined using the following constraints:

```
[Z]!.0 = Z.*
[Z]!.s(x) = Z.*
```

The absence of any coalgebraic operations for the sort `1` immediately results in the equation $(\forall s, t)\ s = t$ with s, t : `1` holding, up to bisimulation, in all coalgebras of the data cosignature of natural numbers. On the other hand, the use of the coalgebraic operation `p` prevents a similar situation for equations of type `Nat`.

The coalgebraic operation `!` does not affect the notion of bisimulation induced by the resulting cosignature. However, this operation is essential for defining derived observers. Specifically, derived observers `<m` : `Nat` $\to$ `1` + `1` with `m` $\in \mathbb{N}$ can be specified using coequations:

```
[T,F]<0 = [F]!
[T,F]<(m+1) = [T,[T,F]<m]p
```

with `m` $\in \mathbb{N}$. According to Remark 5.2.9, these coequations translate to constraints:

```
[T,F]<0.n = F.f if [F]!.n = F.f
```

[7] Note that, by the induction hypothesis, each of $c_1, \ldots, c_k$ are one-step Ξ-observations.

```
[T,F]<(m+1).n = F.f if [T,[T,F]<m]p.n = F.f
[T,F]<(m+1).n = T.t if [T,[T,F]<m]p.n = T.t
```

A consequence of the coequational definition of <m with m ∈ ℕ is that the coequation:

```
[T,F]<(m+1) = [T,F]<m if ([T,F]<m,T)
```

with m ∈ ℕ (formalising the fact that a natural number which is smaller than m is also smaller than m+1) holds in all coalgebras of the cosignature of natural numbers. This follows by induction on ℕ using the deduction calculus of many-sorted coalgebra. Alternatively, one can use induction both on ℕ and on the sort Nat to show that the above coequation holds, up to reachability, in all coalgebras of the cosignature of natural numbers.

In the presence of equationally-defined algebraic operations, standard algebraic techniques can be used to prove the satisfaction of equations involving such operations by the initial coalgebras of data cosignatures, as well as by the quotients by bisimilarity of these coalgebras.

Example 5.2.14. Given the joint data cosignature of booleans and natural numbers, additional algebraic operations on natural numbers can be defined similarly to natural number addition (see Example 5.2.2). In particular, an equality test == : Nat Nat → Bool can be defined inductively using the following equations:

```
0==0 = true
0==s(y) = false
s(x)==0 = false
s(x)==s(y) = x==y
```

The correctness of the above definition follows, again, from bisimilarity on the initial coalgebra of the data cosignature of booleans and natural numbers being given by the equality relation. Furthermore, one can use induction to prove that the equation:

```
not(s(x)==x) = true
```

holds in the initial algebra of the algebraic specification of booleans and natural numbers. Consequently, the equation also holds in the initial coalgebra of the data cosignature of booleans and natural numbers, as well as in its quotient by bisimilarity.

5.3 Specifying Objects

Instantiating the approach described in Sections 3.3.1–3.3.6 to endofunctors of the form of the ones induced by many-sorted signatures and respectively cosignatures whose sorts and operation symbols have been classified into visible and hidden ones (with the interpretation of the visible sorts and operation symbols being fixed) yields a formalism for the specification and verification of objects. The resulting formalism and its associated proof techniques are described in the following.

5.3.1 Syntax and Semantics

As in the case of data types, attention is restricted to lifted signatures of the form of the ones considered in Section 3.3.6. This restriction simplifies the presentation of the results as well as the associated proof techniques, and at the same time appears to be sufficiently general to cover the situations which arise typically in practice.

The next example provides some intuitions for the forthcoming definitions.

Example 5.3.1. Recall from Examples 5.1.1 and 5.1.4 that a specification of stacks involves visible sorts `1` and `Nat`, a hidden sort `Stack`, observers `top` : `Stack` $\to$ `1 Nat`, `rest` : `Stack` $\to$ `1 Stack`, constructors `empty` : $\to$ `Stack`, `push` : `Stack Nat` $\to$ `Stack`, `pop` : `Stack` $\to$ `Stack`, and constraints:

```
[Z,N]top.empty = Z.*
[Z,S]rest.empty = Z.*

[Z,N]top.push(s,n) = N.n
[Z,S]rest.push(s,n) = S.s

[Z,N]top.pop(s) = Z.* if [Z,[Z,N]top]rest.s = Z.z
[Z,N]top.pop(s) = N.n if [Z,[Z,N]top]rest.s = N.n
[Z,S]rest.pop(s) = Z.* if [Z,[Z,S]rest]rest.s = Z.z
[Z,S]rest.pop(s) = S.s' if [Z,[Z,S]rest]rest.s = S.s'
```

The above specification should be regarded as relative to a fixed data universe, given by a data cosignature containing the sorts `1` and `Nat` together with a coalgebra of this data cosignature. The data cosignature is here taken to consist of sorts `1`, `Bool` and `Nat`, algebraic operations `*` : $\to$ `1`, `true` : $\to$ `Bool`, `false` : $\to$ `Bool`, `0` : $\to$ `Nat`, `s` : `Nat` $\to$ `Nat` and coalgebraic operations `?` : `Bool` $\to 1+1$, `p` : `Nat` $\to 1+$ `Nat`, subject to the constraints given in Examples 5.2.1 and 5.2.2, while the coalgebra is taken to be the quotient by bisimilarity of the initial coalgebra of this data cosignature.

Now let $S = \{1, \mathtt{Bool}, \mathtt{Nat}, \mathtt{Stack}\}$, let D denote the above-mentioned coalgebra, let $\mathsf{F}, \mathsf{G} : \mathsf{Set}_D^S \to \mathsf{Set}_D^S$ denote the endofunctors whose `1`-, `Bool`- and `Nat`-sorted components are identities, and whose `Stack`-sorted components are given by:

$$(\mathsf{F}X)_{\mathtt{Stack}} = 1 + (X_{\mathtt{Stack}} \times X_{\mathtt{Nat}}) + X_{\mathtt{Stack}}$$
$$(\mathsf{G}X)_{\mathtt{Stack}} = (X_1 + X_{\mathtt{Nat}}) \times (X_1 + X_{\mathtt{Stack}})$$

and let $\mathsf{U} : \mathsf{Coalg}(\mathsf{Set}_D^S, \mathsf{G}) \to \mathsf{Set}_D^S$ denote the functor taking $(\mathsf{Set}_D^S, \mathsf{G})$-coalgebras to their carrier. Then, the above constraints induce a natural transformation $\rho : \mathsf{FU} \Rightarrow \mathsf{G}(\mathsf{U} + \mathsf{FU})$, whose `1`-, `Bool`- and `Nat`-sorted components are identities, and whose `Stack`-sorted components are given by:

$$\pi_1((\rho_C)_{\mathtt{Stack}}(\iota_1(*))) = \iota_1(*_D)$$
$$\pi_2((\rho_C)_{\mathtt{Stack}}(\iota_1(*))) = \iota_1(*_D)$$

$$\pi_1((\rho_C)_{\mathtt{Stack}}(\iota_2(\langle c, d\rangle))) = \iota_2(d)$$
$$\pi_2((\rho_C)_{\mathtt{Stack}}(\iota_2(\langle c, d\rangle))) = \iota_2(c)$$

$$\pi_1((\rho_C)_{\mathtt{Stack}}(\iota_3(c))) = \begin{cases} \iota_1(*_D) & \text{if } [\iota_1, \mathtt{top}_C](\mathtt{rest}_C(c)) \in \iota_1(C_1) \\ \iota_2(d) & \text{if } [\iota_1, \mathtt{top}_C](\mathtt{rest}_C(c)) = \iota_2(d) \in \iota_2(C_{\mathtt{Nat}}) \end{cases}$$

$$\pi_2((\rho_C)_{\mathtt{Stack}}(\iota_3(c))) = \begin{cases} \iota_1(*_D) & \text{if } [\iota_1, \mathtt{rest}_C](\mathtt{rest}_C(c)) \in \iota_1(C_1) \\ \iota_2(c') & \text{if } [\iota_1, \mathtt{rest}_C](\mathtt{rest}_C(c)) = \iota_2(c') \in \iota_2(C_{\mathtt{Stack}}) \end{cases}$$

for each $(\mathsf{Set}^S_D, \mathsf{G})$-coalgebra C and each $c \in C_{\mathtt{Stack}}$.

We now fix a data cosignature $(V, \Psi, \Xi, E_{\Psi,\Xi})$, and let D denote the quotient of the initial $(V, \Psi, \Xi, E_{\Psi,\Xi})$-coalgebra by Ξ-bisimilarity. (For convenience, we assume that any derived Ξ-observers have been incorporated into Ξ, and that the corresponding defining constraints have been incorporated into $E_{\Psi,\Xi}$.) It therefore follows that D satisfies (in the standard sense) all the coequations satisfied up to reachability by the final $(V, \Psi, \Xi, E_{\Psi,\Xi})$-coalgebra, as well as all the equations satisfied up to bisimulation by the initial $(V, \Psi, \Xi, E_{\Psi,\Xi})$-coalgebra. In particular, D satisfies (in the standard sense) any coequation or equation satisfied up to reachability, respectively up to bisimulation by all $(V, \Psi, \Xi, E_{\Psi,\Xi})$-coalgebras.

As far as equationally-defined algebraic operations on the data are concerned, they are not to be implicitly regarded as part of the signature Ψ. Such operations can be used in object specifications; moreover, in certain cases their availability will prove crucial (see e.g. Example 5.3.23). However, the formulation of some of the forthcoming results relies upon the absence of such operations. For this reason, in stating these results it is assumed that the many-sorted signature Ψ does not contain any equationally-defined algebraic operations, and a remark regarding the existence of a generalisation of the result in question to the case when equationally-defined algebraic operations on the data are present is added whenever this is appropriate.

We continue with a few definitions.

Definition 5.3.2. *A* **cosignature of observers over** Ξ *is given by a pair* (H, Δ)*, with* H *a set of* **hidden sorts** *and* Δ *a* $V \cup H$*-sorted cosignature satisfying: (i)* $\Xi \subseteq \Delta$*, and (ii)* $(\Delta \setminus \Xi)_v = \emptyset$ *for* $v \in V$*. Also, a* **cosignature morphism** *between cosignatures of observers* (H, Δ) *and* (H', Δ') *is given by a many-sorted cosignature morphism* $\phi : (V \cup H, \Delta) \to (V \cup H', \Delta')$ *satisfying: (i)* $\phi{\restriction_\Xi} = \iota_\Xi : \Xi \to \Delta'$*, and (ii)* $\phi(H) \subseteq H'$*.*

Definition 5.3.3. *Let* Δ *denote a cosignature of observers over* Ξ*. A* Δ_D**-coalgebra** *(respectively* Δ_D**-homomorphism***) is a many-sorted* $(V \cup H, \Delta)$*-coalgebra* A *(many-sorted* $(V \cup H, \Delta)$*-homomorphism* f*) such that* $A{\restriction_\Xi} = D{\restriction_\Xi}$ *(*$f{\restriction_V} = 1_D$*).*

Example 5.3.4. The observers used in Example 5.1.1 to specify stack objects define a cosignature of observers over the many-sorted cosignature consisting of sorts `1` and `Nat`, and an operation symbol $\mathtt{p} : \mathtt{Nat} \to \mathtt{1} + \mathtt{Nat}$.

Cosignatures of observers and their morphisms are instances of the abstract notions of cosignature and cosignature morphism[8], while many-sorted Δ-coequations of hidden sort (with Ξ-operation symbols being allowed in coequations) are an instance of the abstract notion of coequation. Cosignatures

[8] In particular, this results in the existence of final coalgebras.

of observers generalise destructor cosignatures (see Definition 4.2.1) by allowing the visible types to carry coalgebraic structure. Moreover, final Δ_D-coalgebras are obtained as cofree many-sorted coalgebras over $D\restriction_\Xi$ along the inclusion cosignature morphism of (V,Ξ) into $(V \cup H, \Delta)$. (The constraints defining cosignatures of observers ensure that these many-sorted $(V \cup H, \Delta)$-coalgebras define Δ_D-coalgebras, while cofreeness over $D\restriction_\Xi$ results in them being final.) It is also worth noting that the $(\Delta \setminus \Xi)_D$-reducts of final Δ_D-coalgebras are final $(\Delta \setminus \Xi)_D$-coalgebras. That is, the Ξ-structure of D does not influence the construction of final Δ_D-coalgebras. This, in turn, results in Δ_D-bisimilarity on Δ_D-coalgebras being the same as $(\Delta\setminus\Xi)_D$-bisimilarity on their $(\Delta\setminus\Xi)_D$-reducts.

Definition 5.3.5. *Let Δ denotes a cosignature of observers. A Δ_D-***behaviour** *for a sort $s \in V \cup H$ is an element $f \in F_s$ of the final Δ_D-coalgebra*[9]*. A condition of form $c.X = Z.Y$ with $X : s$, $c \in T_\Delta[\mathcal{C}]_s$, $Y : s'$ and $Z \in covar(c)_{s'}$ is said to* **hold** *for a Δ_D-behaviour $f \in F_s$ for the sort s if and only if $c_F(f) \in \iota_Z(F_{s'})$.*

Signatures of constructors over Ψ and their *algebras*, and respectively *morphisms* between signatures of constructors are defined similarly to cosignatures of observers and their coalgebras (with initial algebras for signatures of constructors being obtained as free many-sorted algebras over $D\restriction_\Psi$), and respectively to morphisms between cosignatures of observers. Signatures of constructors and their morphisms are instances of the abstract notions of signature and signature morphism, while many-sorted equations of hidden sort are an instance of the abstract notion of equation.

Example 5.3.6. The constructors used in Example 5.1.1 to specify stack objects define a signature of constructors over the many-sorted signature consisting of sorts `1` and `Nat`, and operation symbols $* : \to$ `1`, $0 : \to$ `Nat` and `s` : `Nat` $\to$ `Nat`.

Definition 5.3.7. *Let Γ denote a signature of constructors over Ψ. A* **one-step Γ-computation** *of sort $s \in V \cup H$ is an s-sorted Γ-term containing at most one $\Gamma \setminus \Psi$-symbol.*

In the following, signatures of constructors and cosignatures of observers are used to specify the functionality of objects and respectively their structural properties, while suitably-restricted sets of constraints are used to specify the relationship between the two.

Definition 5.3.8. *An* **object signature** *is a tuple (H, Δ, Γ, E) with (H, Δ) a cosignature of observers, (H, Γ) a signature of constructors, and E a set of (Γ, Δ)-constraints additionally satisfying:*

1. *All the constraints in E are of form:*

$$[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t \text{ if } c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$$

with $\gamma \in \Gamma_{s_1 \ldots s_m, h}$, $\delta \in \Delta_{h, s'_1 \ldots s'_n}$, $h \in H$, $t \in T_\Gamma(\{Y_1, \ldots, Y_k\})$ a one-step Γ-computation, and with $c_i \in T_{\Delta \cup \Delta^d}[\mathcal{C}_i]$ and $Z'_i \in \mathcal{C}_i$ for $i = 1, \ldots, k$.

[9] Consequently, a Δ_D-behaviour for a sort $v \in V$ is given by an element $d \in D_v$, which, in turn, induces a Ξ-behaviour for the sort v (given by the image of d under the unique Ξ-coalgebra homomorphism into the final Ξ-coalgebra). This observation will be used later in the chapter.

2. *For each $h \in H$, $\gamma \in \Gamma_{s_1 \ldots s_m, h}$, $\delta \in \Delta_{h, s'_1 \ldots s'_n}$ and each choice of Δ_D-behaviours for the sorts $s_1, \ldots, s_m$, there exists $[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t$ if C in E such that C holds for the chosen Δ_D-behaviours for $X_1, \ldots, X_m$.*
3. *If $[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z_{k_i}.t_i$ if C_i in E for $i = 1, 2$ are such that both C_1 and C_2 hold for some choice of Δ_D-behaviours for $X_1, \ldots, X_m$, then $Z_{k_1} = Z_{k_2}$ and moreover, if $Z_{k_1}, Z_{k_2} : h$ with $h \in H$ (respectively $Z_{k_1}, Z_{k_2} : v$ with $v \in V$), then $\bar{t}_1 = \bar{t}_2$ ($D \models_\Psi (\forall \overline{Y})\ \bar{t}_1 = \bar{t}_2$)[10], with $\bar{t}_1$ and $\bar{t}_2$ having the same denotations as in Definition 5.2.4, and with $\overline{Y}$ consisting of the variables appearing in $\bar{t}_1, \bar{t}_2$.*

Now let $\mathsf{F}, \mathsf{G} : \mathrm{Set}_D^{V \cup H} \to \mathrm{Set}_D^{V \cup H}$ denote the endofunctors induced by (H, Γ) and (H, Δ). (Their components are given by:

$$(\mathsf{F}X)_s = \begin{cases} \coprod_{\gamma \in \Gamma_{s_1 \ldots s_n, s}} (X_{s_1} \times \ldots \times X_{s_n}) & \text{if } s \in H \\ D_s & \text{if } s \in V \end{cases}$$

and respectively by:

$$(\mathsf{G}X)_s = \begin{cases} \prod_{\delta \in \Delta_{s, s_1 \ldots s_n}} (X_{s_1} + \ldots + X_{s_n}) & \text{if } s \in H \\ D_s & \text{if } s \in V \end{cases}$$

for $X \in |\mathrm{Set}_D^{V \cup H}|$ and $s \in V \cup H$.) Also, let T denote the monad induced by F, and let $\mathsf{U} : \mathrm{Coalg}(\mathrm{Set}_D^{V \cup H}, \mathsf{G}) \to \mathrm{Set}_D^{V \cup H}$ denote the functor taking $(\mathrm{Set}_D^{V \cup H}, \mathsf{G})$-coalgebras to their carrier. Then, object signatures (H, Δ, Γ, E) induce lifted signatures $(\mathrm{Set}_D^{V \cup H}, \mathsf{G}, \mathsf{F}, \sigma)$, with $\sigma : \mathsf{TU} \Rightarrow \mathsf{GTU}$ being the natural transformation induced by the natural transformation $\rho : \mathsf{FU} \Rightarrow \mathsf{G}(\mathsf{U} + \mathsf{FU})$ (see Section 3.3.6) whose hidden-sorted components are given by:

$$\pi_\delta(\rho_{A,h}(\iota_\gamma(\langle a_1, \ldots, a_m \rangle))) = \begin{cases} \iota_1(t_D(y_1, \ldots, y_k)) \\ \qquad \text{if } t \in T_{(V \cup H, \Psi)}(\{Y_1, \ldots, Y_k\}) \\ \iota_2(\iota_{\gamma'}(\langle t_{1,D}(y_1, \ldots, y_k), \ldots, t_{l,D}(y_1, \ldots, y_k) \rangle)) \\ \qquad \text{if } t = \gamma'(t_1, \ldots, t_l) \\ \qquad \text{with } t_1, \ldots, t_l \in T_{(V \cup H, \Psi)}(\{Y_1, \ldots, Y_k\}) \end{cases}$$

(where, by convention, $t_D(y_1, \ldots, y_k) = y_i$ if $t = Y_i : h$, for $t \in T_{(V \cup H, \Psi)}(\{Y_1, \ldots, Y_k\})$), for each Δ_D-coalgebra A, $h \in H$, $\gamma \in \Gamma_{s_1 \ldots s_m, h}$, $\delta \in \Delta_{h, s'_1 \ldots s'_n}$ and $a_i \in A_{s_i}$ for $i = 1, \ldots, m$, where the constraint $[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t$ if $c_1.X_{j_1} = Z'_1.Y_1, \ldots, c_k.X_{j_k} = Z'_k.Y_k$ in E is such that its conditions hold for $a_1, \ldots, a_m$, and where $y_i = (c_i)_{A^\mathsf{d}}(a_{j_i})$ for $i = 1, \ldots, k$.

Moreover, the algebras of the induced lifted signature are in one-to-one correspondence with the (Γ_D, Δ_D)-*bialgebras* (that is, (Γ, Δ)-bialgebras whose underlying Ψ- and Ξ-structures are given by

[10] In the presence of derived observers in C_1, C_2, the requirements that $\bar{t}_1 = \bar{t}_2$ and respectively $D \models_\Psi (\forall \overline{Y})\ \bar{t}_1 = \bar{t}_2$ can be weakened by replacing all the derived observers with their definitions before identifying the Ys with similar definitions in C_1 and C_2. (Some further case analysis may be required to expand the definitions of the derived observers.)

$D\restriction_\Psi$ and respectively $D\restriction_\Xi$) satisfying E. Again, this is a consequence of the definition of ρ and of Remark 3.3.26.

Suitable denotations for object signatures are provided by the codomains of the quotients of the unique homomorphisms from the initial to the final coalgebras of such signatures (see Section 3.3.1).

Example 5.3.9. The cosignature of observers, signature of constructors and set of constraints in Example 5.3.1 define an object signature. The `Stack`-component of the carrier of the algebra used as denotation for this object signature is isomorphic to $(\mathbb{N})^*$ – its elements are in one-to-one correspondence with finite lists of natural numbers. (The notion of bisimilarity on the initial algebra of this object signature relates any stack denoted by a term over $\{\mathtt{empty}, \mathtt{push}, \mathtt{pop}\}$ to a stack denoted by a term over $\{\mathtt{empty}, \mathtt{push}\}$[11].)

Example 5.3.10. An object signature for bounded stacks of maximum size `m`, with $\mathtt{m} \in \mathbb{N}$ can be obtained by extending the object signature of stacks with a hidden sort `BStack`, observers `stack` : `BStack` $\rightarrow$ `Stack`, `depth` : `BStack` $\rightarrow$ `Nat`, constructors `empty` : $\rightarrow$ `BStack`, `push` : `BStack Nat` $\rightarrow$ `BStack`, `pop` : `BStack` $\rightarrow$ `BStack`, and constraints:

```
[S]stack.empty = S.empty
[N]depth.empty = N.0

[S]stack.push(b,n) = S.push(s,n)
     if [[F,T]<m]depth.b = F.f and [S]stack.b = S.s
[S]stack.push(b,n) = S.s
     if [[F,T]<m]depth.b = T.t and [S]stack.b = S.s
[N]depth.push(b,n) = N.s(d)
     if [[F,T]<m]depth.b = F.f and [D]depth.b = D.d
[N]depth.push(b,n) = N.d
     if [[F,T]<m]depth.b = T.t and [D]depth.b = D.d

[S]stack.pop(b) = S.pop(s) if [S]stack.b = S.s
[N]depth.pop(b) = N.0 if [[F,D]p]depth.b = F.f
[N]depth.pop(b) = N.d if [[F,D]p]depth.b = D.d
```

The above constraints together with the constraints given in Example 5.3.1 satisfy the conditions in Definition 5.3.8, and therefore define an object signature. The hidden components of the carrier of the algebra used as denotation for this object signature are isomorphic to $(\mathbb{N})^*$ and respectively the subset of $(\mathbb{N})^*$ whose elements are lists of length not exceeding `m`. (The notion of bisimilarity on the initial algebra of this object signature relates any bounded stack denoted by a term over $\{\mathtt{empty}, \mathtt{push}, \mathtt{pop}\}$ to a bounded stack denoted by a term over $\{\mathtt{empty}, \mathtt{push}\}$ containing at most `m` occurrences of `push`.)

Definition 5.3.11. *Let (H, Δ, Γ, E) and $(H', \Delta', \Gamma', E')$ denote object signatures. An* **object signature morphism** *from (H, Δ, Γ, E) to $(H', \Delta', \Gamma', E')$ is a pair (θ, ϕ), with $\theta : (H, \Delta) \rightarrow (H', \Delta')$ a cosignature morphism and $\phi : (H, \Gamma) \rightarrow (H', \Gamma')$ a signature morphism, additionally satisfying:*

1. $\phi\restriction_H = \theta\restriction_H$

[11] See Example 5.3.21 for a proof of this statement.

2. *the functions* $\phi{\restriction_{\Gamma_h}} : \Gamma_h \to \Gamma'_{\phi(h)}$ *with* $h \in H$ *are bijections*
3. $(\phi, \theta)(e) \in E'$ *for each* $e \in E$.

If, in addition:

4. *the functions* $\theta{\restriction_{\Delta_h}} : \Delta_h \to \Delta'_{\theta(h)}$ *with* $h \in H$ *are surjections*

then (θ, ϕ) *is called* **horizontal**.

If $(\mathsf{Set}_D^{V \cup H}, \mathsf{G}, \mathsf{F}, \sigma)$ and $(\mathsf{Set}_D^{V \cup H'}, \mathsf{G}', \mathsf{F}', \sigma')$ denote the lifted signatures induced by the object signatures (H, Δ, Γ, E) and respectively $(H', \Delta', \Gamma', E')$ (with σ and σ' being defined via ρ and ρ'), then (horizontal) object signature morphisms $(\theta, \phi) : (H, \Delta, \Gamma, E) \to (H', \Delta', \Gamma', E')$ induce (horizontal) lifted signature morphisms $(\mathsf{U}, \eta, \xi) : (\mathsf{Set}_D^{V \cup H}, \mathsf{G}, \mathsf{F}, \sigma) \to (\mathsf{Set}_D^{V \cup H'}, \mathsf{G}', \mathsf{F}', \sigma')$, with $(\mathsf{U}, \eta) : (\mathsf{Set}_D^{V \cup H}, \mathsf{G}) \to (\mathsf{Set}_D^{V \cup H'}, \mathsf{G}')$ and $(\mathsf{U}, \xi) : (\mathsf{Set}_D^{V \cup H}, \mathsf{F}) \to (\mathsf{Set}_D^{V \cup H'}, \mathsf{F}')$ denoting the abstract cosignature and signature morphism induced by θ and respectively ϕ. The fact that ξ is a natural isomorphism follows from the second requirement in Definition 5.3.11. Also, the fact that $\tau_{A' + \mathsf{F}'A'} \circ \mathsf{U}\rho'_{A'} \circ \xi_{A'} = \mathsf{G}[\mathsf{U}\iota'_1, \mathsf{U}\iota'_2] \circ \mathsf{G}(1_{\mathsf{U}A'} + \xi_{A'}) \circ \rho_{A'{\restriction_\Delta}}$ (with $\tau : \mathsf{TU} \Rightarrow \mathsf{UT}'$ denoting the natural transformation induced by the natural transformation $\xi : \mathsf{FU} \Rightarrow \mathsf{UF}'$, see Section 3.3.6) for any Δ'_D-coalgebra A' follows from the definitions of ρ and ρ' using the third requirement in Definition 5.3.11. Finally, the fourth requirement in Definition 5.3.11 ensures that η is a natural monomorphism.

We conclude this section with a few remarks concerning *derived* Δ*-observers*. (The use of such observers facilitates both the definition of object signatures and the formulation of correctness properties for the specified behaviours.)

Derived Δ-observers are defined similarly to derived Ξ-observers (see Section 5.2.2), except that here the algebraic and coalgebraic structure of D can be used in definitions. Specifically, *derived* Δ*-observers* $\delta : h \to s_1 \dots s_n$ are defined using $(\Psi, \Delta \cup \{\delta\})$-constraints E_δ of form:

$$[Z_1, \dots, Z_n]\delta.X = Z.t \text{ if } C$$

with C only containing Δ-symbols, additionally satisfying:

1. For any choice of a Δ_D-behaviour for X, E_δ contains a constraint whose condition holds for that Δ_D-behaviour.
2. Whenever $[Z_1, \dots, Z_n]\delta.X = Z_{k_i}.t_i$ if C_i with $i = 1, 2$ in E_δ are such that C_1 and C_2 hold for some choice of a Δ_D-behaviour for X, it follows that $Z_{k_1} = Z_{k_2}$, while $D \models_\Psi (\forall \overline{Y})\ \bar{t}_1 = \bar{t}_2$ (with $\bar{t}_1$ and $\bar{t}_2$ having the same denotations as in Section 5.2, and with $\overline{Y}$ consisting of the variables appearing in $\bar{t}_1, \bar{t}_2$).

In particular, *constant observers*[12] can be defined using constraints of form $[Z]c.X = Z.t$ with $t \in T_\Psi$.

Remark 5.3.12. A consequence of Remark 5.1.6 is that sets of $\Delta \cup \{\delta\}$-coequations of form:

$$[Z_1, \dots, Z_n]\delta = c \text{ if } C'$$

[12] Constant observers are similar to the *constant observations* used in [Cor98].

with $covar(c) \subseteq \{Z_1, \ldots, Z_n\}$ and with C' only containing Δ-symbols, subject to constraints similar to 1 and 2 above can alternatively be used to specify derived Δ-observers whose definition does not involve Ψ-symbols.

Example 5.3.13. Given the object signature in Example 5.3.10, one can define a derived observer `full?` : `BStack` $\to$ `1 1` using the coequation:

```
[T,F]full? = [[F,T]<m]depth
```

Since the definitions of derived Δ-observers determine unique extensions of Δ_D-coalgebra structures to $(\Delta \cup \{\delta\})_D$-coalgebra structures, and since such definitions only involve Δ- and Ψ-symbols (and hence commute with Δ_D-coalgebra homomorphisms), it follows that derived observers induce $(\mathsf{Set}_D^{V \cup H}, \mathsf{G})$-observers. As a result, derived Δ-observers can be used both in defining object signatures (namely in the conditional parts of constraints), and in formalising correctness properties by means of coequations.

In order to use derived observers in the conditions of constraints, one has to define what it means for a condition containing a derived observer to hold in a Δ_D-coalgebra. For a Δ_D-coalgebra A and a cosignature of derived Δ-observers Δ^{d}, the unique $(\Delta \cup \Delta^{\mathsf{d}})_D$-coalgebra obtained by interpreting the Δ^{d}-observers in A according to their definition is denoted A^{d}. Then, a condition of form $c.X = Z.Y$ with $X : s$, $c \in T_{\Delta \cup \Delta^{\mathsf{d}}}[\mathcal{C}]_s$, $Y : s'$ and $Z \in covar(c)_{s'}$ is said to *hold* for a Δ_D-behaviour $f \in F_s$ for the sort s if and only if $c_{F^{\mathsf{d}}}(f) \in \iota_Z(F_{s'})$.

Remark 5.3.14. If E_Δ^{d} denotes the set of $(\Psi, \Delta \cup \Delta^{\mathsf{d}})$-constraints defining some derived Δ-observers Δ^{d}, then E_Δ^{d} can be used to express arbitrary $\Delta \cup \Delta^{\mathsf{d}}$-observations in terms of Δ-observers and Ψ-operations only. Specifically, for any $c \in T_{\Delta \cup \Delta^{\mathsf{d}}}[\mathcal{C}]_s$, any $t \in T_\Psi(\{X_1, \ldots, X_m\})_s$ and any choice of a Δ_D-behaviour for the sorts of $X_1, \ldots, X_m$, there exists a $(\Psi, \Delta \cup \Delta^{\mathsf{d}})$-constraint e of form $c.t = Z.t'$ if C, with C only containing Δ-observers and holding for the chosen Δ_D-behaviours for $X_1, \ldots, X_n$, such that $E_{\Psi,\Xi} \cup E_\Delta^{\mathsf{d}} \vdash e$. This follows by structural induction on c. First, if c is a covariable, then the conclusion follows immediately by **base$_2$**. Also, if c is of form $[c_1, \ldots, c_n]\delta$ with $\delta \in (\Delta \cup \Delta^{\mathsf{d}})_v = \Xi_v$ for some $v \in V$ (and hence $t \in T_\Psi(\{X_1, \ldots, X_m\})_v$ and $c \in T_\Xi[\mathcal{C}]_v$), then the conclusion follows from the fact that $E_{\Psi,\Xi} \vdash c.t = Z.t'$ if C for some conditions C which hold for (the Ξ-behaviours resulting from) the chosen Δ_D-behaviours for $X_1, \ldots, X_m$. Finally, if c is of form $[c_1, \ldots, c_n]\delta$ with $\delta \in (\Delta \cup \Delta^{\mathsf{d}})_h$ with $h \in H$ (and hence $m = 1$ and $t = X_1$), then depending on whether $\delta \in \Delta_h$ or $\delta \in \Delta_h^{\mathsf{d}}$, **base$_2$** and respectively the definition of δ yields $E_{\Psi,\Xi} \cup E_\Delta^{\mathsf{d}} \vdash [Z_1, \ldots, Z_n]\delta.X_1 = Z_i.t'$ if C, with C of form $c_1.X_1 = Z_1'.Y_1, \ldots, c_k.X_1 = Z_k'.Y_k$ holding for the chosen Δ_D-behaviour for X_1. Then, **substitution$_2$** yields $E_{\Psi,\Xi} \cup E_\Delta^{\mathsf{d}} \vdash [c_1, \ldots, c_n]\delta.X_1 = Z.t''$ if $C, ([\ldots, c_i'/Z_{k_i}', \ldots]c_{k_i}.X_1 = Z_i''.Y_i')_{i=1,\ldots,l}$, where, by the induction hypothesis, $E_{\Psi,\Xi} \cup E_\Delta^{\mathsf{d}} \vdash c_i.t' = Z.t''$ if C' with C' of form $c_1'.Y_{k_1} = Z_1''.Y_1', \ldots, c_l'.Y_{k_l} = Z_l''.Y_l'$ holding for the Δ_D-behaviours for $Y_1, \ldots, Y_k$ resulting from the chosen Δ_D-behaviour for X together with the conditions defining $Y_1, \ldots, Y_k$.

The statement in Remark 5.3.14 does not generalise to the case when equationally-defined algebraic operations are allowed in Ψ. For, in this case, a (Ψ, Ξ)-constraint as required by the proof of this statement can not, in general, be exhibited.

Finally, it is worth noting that cosignature morphisms $\theta : (H, \Delta) \to (H', \Delta')$ induce translations of $(\Psi, \Delta \cup \{\delta\})$-constraints into $(\Psi, \Delta' \cup \{\delta\})$-constraints, and hence of derived Δ-observers into derived Δ'-observers. (Such translations are needed whenever derived observers are present in the constraints used to define object signatures or in the coequations used to formalise correctness properties, in order to translate those constraints or coequations along object signature morphisms.)

5.3.2 Correctness Proofs

The notions of satisfaction (of hidden coequations up to reachability and of hidden equations up to bisimulation) obtained by instantiating the abstract notions of satisfaction given by Definition 3.3.12 (see also Remark 3.3.14) are as follows.

Definition 5.3.15. *Let (H, Δ, Γ, E) denote an object signature. An (H, Δ, Γ, E)-algebra A* **satisfies** *a hidden Δ-coequation of form $c = c'$ if $(c_1, \mathcal{C}_1), \ldots, (c_n, \mathcal{C}_n)$* **up to reachability** *if and only if, for any $t \in T_\Gamma$[13], $c_A(t_A) = c'_A(t_A)$ holds whenever $(c_i)_A(t_A) \in \iota_{Z_i}(A_{s_i})$ for some $s_i \in S$ and some $Z_i \in \mathcal{C}_{i,s_i}$, for $i = 1, \ldots, n$. Also, A* **satisfies** *a hidden Γ-equation of form $(\forall \mathcal{V})\ l = r$* **up to bisimulation** *if and only if $\theta^\#(l) \sim_A \theta^\#(r)$ holds for any assignment $\theta : \mathcal{V} \to A$, with $\sim_A$ denoting Δ_D-bisimilarity (see Proposition 4.2.10) on $A{\restriction_\Delta}$.*

According to Theorems 3.3.17 and 3.3.19, the notion of object signature morphism yields an institution w.r.t. the satisfaction of hidden coequations up to reachability, while the notion of horizontal object signature morphism yields an institution w.r.t. the satisfaction of hidden equations up to bisimulation. Moreover, since $\mathsf{Set}_D^{V \cup H}$ is regular (see Example 2.1.47), it follows by Proposition 3.3.23 that the algebras used as denotations for object signatures satisfy precisely those hidden coequations (respectively equations) which are satisfied up to reachability (up to bisimulation) by the final (initial) algebras.

To facilitate the formulation of correctness properties of object behaviour, one can also define *derived Γ-constructors* $\gamma : s_1 \ldots s_m \to h$, namely using $(\Gamma \cup \{\gamma\}, \Xi)$-constraints E_γ of form:

$$Z.\gamma(X_1, \ldots, X_m) = Z.t \text{ if } C$$

with t only containing Γ-symbols, subject to restrictions similar to those in the definition of derived Δ-observers. Moreover, the use of Ξ-observers in such constraints allows for definitions by case analysis. (Derived Γ-constructors γ whose definitions do not require case analysis can alternatively be specified using $\Gamma \cup \{\gamma\}$-equations of form: $(\forall \overline{X})\ \gamma(X_1, \ldots, X_m) = t$ with $t \in T_\Gamma(\{X_1, \ldots, X_m\})$.) Derived Γ-constructors induce $(\mathsf{Set}_D^{V \cup H}, \mathsf{F})$-constructors, and therefore can be used in formalising correctness properties by means of equations.

Remark 5.3.16. If E_Γ^{d} denotes the set of $(\Gamma \cup \Gamma^{\mathsf{d}}, \Xi)$-constraints defining some derived Γ-constructors Γ^{d}, then E_Γ^{d} can be used to express arbitrary $\Gamma \cup \Gamma^{\mathsf{d}}$-computations in terms of Γ-constructors and Ξ-operations only. Specifically, for any $t \in T_{\Gamma \cup \Gamma^{\mathsf{d}}}(\mathcal{V})_s$, any $c \in T_\Xi[\mathcal{C}]_s$ and any choice of Ξ_D-behaviours for the variables appearing in t, there exists a $(\Gamma \cup \Gamma^{\mathsf{d}}, \Xi)$-constraint e of form $c.t = Z.t'$ if C, with

[13] The fact that D is given by a quotient of the initial Ξ-algebra, and therefore is Ξ-reachable, is used here.

t' only containing Γ-symbols and with C holding for the chosen Ξ_D-behaviours for the variables appearing in t, such $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Gamma} \vdash e$. This follows by structural induction on t. First, if t is a variable, then the conclusion follows immediately by **base$_2$**. Also, if $t = \gamma(t_1, \ldots, t_m)$ with $\gamma \in (\Gamma \cup \Gamma^{\mathsf{d}})_v = \Psi_v$ for some $v \in V$ (and hence $t \in T_{\Psi}(\mathcal{V})_v$ and $c \in T_{\Xi}[\mathcal{C}]_v$), then the conclusion follows from the fact that $E_{\Psi,\Xi} \vdash c.t = Z.t'$ if C, for some conditions C which hold for the Ξ-behaviours resulting from the chosen Ξ_D-behaviours for the variables appearing in t. Finally, if $t = \gamma(t_1, \ldots, t_m)$ with $\gamma \in (\Gamma \cup \Gamma^{\mathsf{d}})_h$ for some $h \in H$ (and hence $c = Z$ for some $Z \in \mathcal{C}$), then the induction hypothesis yields $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Gamma} \vdash c'.t_i = Z'.t'$ if C' with C' holding for the chosen Ξ_D-behaviours for the variables appearing in t_i, for each $i \in \{1, \ldots, m\}$ and each $c' \in T_{\Xi}[\mathcal{C}]$ appropriate for t_i. Also, the chosen Ξ_D-behaviours for the variables appearing in $t_1, \ldots, t_m$ yield a choice of Ξ_D-behaviours for (the sorts of) $t_1, \ldots, t_m$. (If t_i is of visible sort, the choice of a Ξ_D-behaviour for t_i is obtained by applying t_i to the chosen Ξ_D-behaviours for the variables appearing in it, whereas if t_i is of hidden sort, there is only one choice of a Ξ_D-behaviour for t_i.) Then, depending on whether $\gamma \in \Gamma_h$ or $\gamma \in \Gamma^{\mathsf{d}}_h$, **base$_1$** and respectively the definition of γ yield $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Gamma} \vdash Z.\gamma(X_1, \ldots, X_m) = Z.t$ if C, with C of form $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_k.X_{j_k} = Z_k.Y_k$ holding for the resulting Ξ_D-behaviours for $t_1, \ldots, t_m$. Now, **substitution$_1$** yields $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Gamma} \vdash Z.\gamma(t_1, \ldots, t_m) = Z.t(t'_1/Y_1, \ldots, t'_k/Y_k)$ if $C_1, \ldots, C_k$, with t'_i and C_i being given by $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Gamma} \vdash c_i.t_{j_i} = Z_i.t'_i$ if C_i for $i = 1, \ldots, k$.

Again, the statement in Remark 5.3.16 does not generalise to the case when equationally-defined algebraic operations are allowed in Ψ (for the same reason for which Remark 5.3.14 did not generalise).

As in the case of data cosignatures, correctness proofs employ inductive and respectively coinductive techniques. These techniques build on the remarks in Section 3.3.1, according to which proving the satisfaction of equations up to bisimulation can be reduced to exhibiting generic bisimulations which relate the lhs and rhs of the given equations, while proving the satisfaction of coequations up to reachability can be reduced to exhibiting subalgebras whose states satisfy the given coequations. At the extremes, this amounts to proving that bisimilarity on the underlying coalgebras relates the lhs and rhs of the given equations, and respectively that reachable states satisfy the given coequations. Furthermore, in proving the satisfaction of coequations up to reachability, it is usually sufficient to consider the reachable subalgebras as candidates for the subalgebras whose states satisfy the coequations.

A result similar to Proposition 5.2.11, justifying the use of the deduction calculus for constraints in correctness proofs (namely for rewriting particular observations of given computations to computations yielding similar results) can be formulated for object signatures.

Proposition 5.3.17. *Let (H, Δ, Γ, E) denote an object signature, and let E^{d}_{Δ} denote the set of $(\Psi, \Delta \cup \Delta^{\mathsf{d}})$-constraints used to define some derived Δ-observers Δ^{d}. Also, let $E' = E_{\Psi,\Xi} \cup E \cup E^{\mathsf{d}}_{\Delta}$. Then, the following hold:*

1. *For each $\gamma \in \Gamma_{s_1 \ldots s_m, s}$, each $c \in T_{\Delta \cup \Delta^{\mathsf{d}}}[\mathcal{C}]_s$ and each choice of Δ_D-behaviours for the sorts $s_1, \ldots, s_m$, there exist a one-step Γ-computation $t \in T_{\Gamma}(\{Y_1, \ldots, Y_k\})$, $Z \in covar(c)$, and conditions C for $X_1 : s_1, \ldots, X_m : s_m$, such that C holds for the chosen Δ_D-behaviours for $s_1, \ldots, s_m$, and such that $E' \vdash c.\gamma(X_1, \ldots, X_m) = Z.t$ if C.*

2. *For each $t \in T_\Gamma(\{X_1, \ldots, X_m\})_s$, each $c \in T_{\Delta \cup \Delta^d}[\mathcal{C}]_s$ and each choice of Δ_D-behaviours for the sorts of $X_1, \ldots, X_m$, there exist $t' \in T_\Gamma(\{Y_1, \ldots, Y_k\})$, $Z \in covar(c)$, and conditions C for $X_1, \ldots, X_m$, such that C holds for the chosen Δ_D-behaviours for $X_1, \ldots, X_m$, and such that $E' \vdash c.t = Z.t'$ if C.*

That is, any Δ-observation on any (one-step) Γ-computation rewrites to a (one-step) Γ-computation, provided that sufficiently-many assumptions are made about the Δ_D-behaviours of the variables appearing in the term used to denote the given Γ-computation.

Proof. In proving the first statement, we first consider the case when c does not contain any derived observers. In this case, the proof is by structural induction on c. First, if c is a covariable, then **base$_1$** yields $E' \vdash Z.\gamma(X_1, \ldots, X_m) = Z.\gamma(Y_1, \ldots, Y_m)$ if $Z_1.X_1 = Z_1.Y_1, \ldots, Z_m.X_m = Z_m.Y_m$ for any $\gamma \in \Gamma$ appropriate for Z. Next, if c is of form $[c_1, \ldots, c_n]\delta$ for some $\delta \in \Delta_{s,s'_1 \ldots s'_n}$, two cases can be distinguished.

1. $\delta \in \Delta_v$ with $v \in V$. It therefore follows that $c \in T_\Xi[\mathcal{C}]$. The fact that $(V, \Psi, \Xi, E_{\Psi,\Xi})$ is a data cosignature then yields, for each $\gamma \in \Psi_{s_1 \ldots s_m, v} = \Gamma_{s_1 \ldots s_m, v}$ and each choice of Ξ-behaviours (and therefore for each choice of Δ_D-behaviours) for the sorts $s_1, \ldots, s_m$, a (Ψ, Ξ)-constraint e of form $c.\gamma(X_1, \ldots, X_m) = Z.t$ if C such that C holds for the chosen Δ_D-behaviours for $X_1, \ldots, X_m$, and such that $E_{\Psi,\Xi} \vdash e$. Hence, $E' \vdash e$.
2. $\delta \in \Delta_h$ with $h \in H$. The fact that (H, Δ, Γ, E) is an object signature then yields, for each $\gamma \in \Gamma_{s_1 \ldots s_m, h}$ and each choice of Δ_D-behaviours for the sorts $s_1, \ldots, s_m$, a (Γ, Δ)-constraint of form $[Z_1, \ldots, Z_n]\delta.\gamma(X_1, \ldots, X_m) = Z_i.t$ if C in E, with C of form $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_k.X_{j_k} = Z_k.Y_k$ and with $t \in T_\Gamma(\{Y_1, \ldots, Y_k\})$ a one-step Γ-computation, such that C holds for the chosen Δ_D-behaviours for $s_1, \ldots, s_m$. The conditions defining $Y_1, \ldots, Y_k$ together with the chosen Δ_D-behaviours for $X_1, \ldots, X_m$ yield a choice of Δ_D-behaviours for $Y_1, \ldots, Y_k$, and hence for the variables appearing in t. We now distinguish two subcases.
   (a) $Z_i : v$ with $v \in V$. Hence, $c_i \in T_\Xi[\mathcal{C}]_v$ and $t \in T_\Psi(\{Y_1, \ldots, Y_k\})$. The fact that $(V, \Psi, \Xi, E_{\Psi,\Xi})$ is a data cosignature then yields a (Ψ, Ξ)-constraint e of form $c_i.t = Z.t'$ if $c'_1.Y_{k_1} = Z'_1.Y'_1, \ldots, c'_l.Y_{k_l} = Z'_l.Y'_l$ whose conditions hold for (the Ξ-behaviours induced by) the resulting Δ_D-behaviours for $Y_1, \ldots, Y_k$, and which is such that $E_{\Psi,\Xi} \vdash e$ (and consequently $E' \vdash e$). Now, **substitution$_2$** yields $E' \vdash [c_1, \ldots, c_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t'$ if $C, ([\ldots c'_i/Z_{k_i} \ldots]c_{k_i}.X_{j_{k_i}} = Z'_i.Y'_i)_{i=1,\ldots,l})$.
   (b) $Z_i : h$ with $h \in H$. First, if $t = Y_j$ with $j \in \{1, \ldots, k\}$, then **base$_2$** yields $E' \vdash c_i.Y_j = Z.Y$ if $c_i.Y_j = Z.Y$, with $c_i.Y_j = Z.Y$ holding for the resulting Δ_D-behaviour for Y_j, while **substitution$_2$** yields $E' \vdash [c_1, \ldots, c_n]\delta.\gamma(X_1, \ldots, X_m) = Z.Y$ if $C, [\ldots c_i/Z_j \ldots]c_j.X_{i_j} = Z.Y$. Also, if $t = \gamma'(t_1, \ldots, t_p)$ with $\gamma' \in \Gamma_h$ and $t_1, \ldots, t_p \in T_\Psi(\{Y_1, \ldots, Y_k\})$, then the constraints in $E_{\Psi,\Xi}$ together with the resulting Δ_D-behaviours for $Y_1, \ldots, Y_k$ yield a choice of Δ_D-behaviours for (the sorts of) $t_1, \ldots, t_p$[14]. The induction hypothesis then yields a (Γ, Δ)-constraint e of form $c_i.\gamma'(X'_1, \ldots, X'_p) = Z.t'$ if $c'_1.X_{r'_1} = Z'_1.Y'_1, \ldots, c'_q.X_{r'_q} = Z'_q.Y'_q$ whose conditions hold

[14] Note that all the variables appearing in $t_1, \ldots, t_p$ are visible-sorted, and therefore their Δ_D-behaviours are given by elements of D.

for the previously-obtained Δ_D-behaviours for $t_1, \ldots, t_p$, and which is such that $E' \vdash e$. Then, **substitution$_1$** yields $E' \vdash c_i.\gamma'(t_1, \ldots, t_p) = Z.t'(t'_1/Y'_1, \ldots, t'_q/Y'_q)$ if $C_1, \ldots, C_q$, with t'_i and C_i being such that $E' \vdash c'_i.t_{r_i} = Z'_i.t'_i$ if C_i, for $i = 1, \ldots, q$. If $C_1 \cup \ldots \cup C_q$ is of form $c''_1.Y_{k_1} = Z''_1.Y''_1, \ldots, c''_l.Y_{k_l} = Z''_l.Y''_l$, **substitution$_2$** yields $E' \vdash [c_1, \ldots, c_n]\delta.\gamma(X_1, \ldots, X_m) = Z.t'(t'_1/Y'_1, \ldots, t'_q/Y'_q)$ if $C, C'_1, \ldots, C'_l$, with C'_i of form $[\ldots c''_i/Z_{k_i} \ldots]c_{k_i}.X_{j_{k_i}} = Z''_i.Y''_i$ for $i = 1, \ldots, l$.

Hence, the first statement holds whenever c does not contain any derived observers. This yields, for each $\gamma \in \Gamma_{s_1 \ldots s_m, s}$ and each choice of Δ_D-behaviours for the sorts $s_1, \ldots, s_m$, a Δ_D-behaviour for the sort s. The fact that the first statement also holds for an arbitrary $c \in T_{\Delta \cup \Delta^{\mathsf{d}}}[\mathcal{C}]$ now follows by Remark 5.3.14, which yields, for each $c \in T_{\Delta \cup \Delta^{\mathsf{d}}}[\mathcal{C}]_s$, a $(\Psi, \Delta \cup \Delta_D)$-constraint e of form $c.X = Z.t$ if $c_1.X = Z_1.Y_1, \ldots, c_k.X = Z_k.Y_k$ whose conditions hold for the previously-obtained Δ_D-behaviour for the sort s, and which is such that $E_{\Psi,\Xi} \cup E^{\mathsf{d}}_{\Delta} \vdash e$. Then, if $E' \vdash c_i.\gamma(X_1, \ldots, X_m) = Z_i.t_i$ if C_i with C_i holding for the given Δ_D-behaviours for $s_1, \ldots, s_m$, for $i = 1, \ldots, k$, **substitution$_1$** yields $E' \vdash c.\gamma(X_1, \ldots, X_m) = Z.t(t_1/Y_1, \ldots, t_k/Y_k)$ if $C_1, \ldots, C_k$.

The proof of the second statement is by structural induction on t. First, if t is a variable, **base$_2$** yields $E' \vdash c.X = Z.Y$ if $c.X = Z.Y$, with $c.X = Z.Y$ holding for the chosen Δ_D-behaviour for (the sort of) X. Next, if t is of form $\gamma(t_1, \ldots, t_k)$ with $\gamma \in \Gamma_{s_1 \ldots s_k, s}$, two cases can be distinguished.

1. $s \in V$. It therefore follows that $\gamma \in \Psi_s$, $t_1, \ldots, t_k \in T_\Psi(\{X_1, \ldots, X_m\})$ and $c \in T_\Xi[\mathcal{C}]$. The conclusion then follows by $(V, \Psi, \Xi, E_{\Psi,\Xi})$ being a data cosignature.
2. $s \in H$. In this case, the induction hypothesis yields, for each $i \in \{1, \ldots, k\}$ and each $c' \in T_\Delta[\mathcal{C}]$ appropriate for t_i, a (Γ, Δ)-constraint e of form $c'.t_i = Z.t'$ if C, with C holding for the chosen Δ_D-behaviours for $X_1, \ldots, X_m$, and with $E' \vdash e$. These constraints uniquely determine a choice of Δ_D-behaviours for the sorts $s_1, \ldots, s_k$. It then follows by (H, Δ, Γ, E) being an object signature that E contains a (Γ, Δ)-constraint of form $c.\gamma(X_1, \ldots, X_k) = Z.t'$ if $c_1.X_{j_1} = Z_1.Y_1, \ldots, c_l.X_{j_l} = Z_l.Y_l$ whose conditions hold for the resulting Δ_D-behaviours for $s_1, \ldots, s_k$. Then, **substitution$_1$** yields $E' \vdash c.t = Z.t'(t'_1/Y_1, \ldots, t'_k/Y_k)$ if $C_1, \ldots, C_k$, with t'_i and C_i being given by $E' \vdash c_i.t_{j_i} = Z_i.t'_i$ if C_i for $i = 1, \ldots, k$.

□

Remark 5.3.18. In the presence of derived Γ-constructors Γ^{d} defined using some $(\Gamma \cup \Gamma^{\mathsf{d}}, \Xi)$-constraints E^{d}_Γ, the second statement generalises to terms t also containing Γ^{d}-symbols. (In this case, deduction from E' is replaced by deduction from $E' \cup E^{\mathsf{d}}_\Gamma$.) This follows by **substitution$_2$** together with the observation in Remark 5.3.16.

Remark 5.3.19. Proposition 5.3.17 does not generalise to the case when equationally-defined algebraic operations are present in Ψ. The reason for this is that the results of Ξ-observations on the results yielded by such operations can not, in general, be expressed using a (Ψ, Ξ)-constraint. Nevertheless, the availability of such operations substantially increases the expressiveness of the formalism, while the proof techniques are, in most cases, as straightforward as in the case when such

operations do not appear in constraints[15].

In addition to inductive and coinductive techniques, coequational and respectively equational deduction can also be used in proving the satisfaction of coequations up to reachability and respectively of equations up to bisimulation by algebras of object signatures.

Theorem 5.3.20 (Soundness).

1. *The deduction calculus in Section 4.2.5 is sound for the satisfaction of coequations up to reachability by algebras of object signatures.*
2. *The deduction calculus in Section 2.3.1 is sound for the satisfaction of equations up to bisimulation by algebras of object signatures.*

Proof (sketch). Soundness of the two **closure** rules follows from the preservation of reachability by observers and respectively of bisimilarity by constructors in algebras of object signatures. Soundness of the remaining rules follows by standard properties of equality. □

The remaining of this section gives some examples of correctness proofs.

Example 5.3.21. Consider the object signature of stacks given in Example 5.3.1 (denoted `STACK` in what follows). The coinductive technique for proving the satisfaction of equations up to bisimulation described in Section 3.3.1 can be used to show that the equations:

```
pop(empty) = empty
pop(push(s,n)) = s
```

with `s` of type `Stack` hold, up to bisimulation, in all `STACK`-algebras. Specifically, the satisfaction of the above equations can be inferred by exhibiting a generic bisimulation on the coalgebras underlying `STACK`-algebras, which, in addition, relates the lhs and rhs of each of these equations.

For a `STACK`-algebra A, let R_A denote the binary relation on the carrier of A whose visible components are given by the equality relations, and whose `Stack` component is the least binary relation on $A_{\texttt{Stack}}$ satisfying:

$c\ R_{A,\texttt{Stack}}\ c$, for each $c \in A_{\texttt{Stack}}$

$\texttt{pop}_A(\texttt{empty}_A)\ R_{A,\texttt{Stack}}\ \texttt{empty}_A$

$\texttt{pop}_A(\texttt{push}_A(c,d))\ R_{A,\texttt{Stack}}\ c$, for each $c \in A_{\texttt{Stack}}$ and each $d \in A_{\texttt{Nat}}$

Then, the relations R_A with $A \in$ Alg(`STACK`) define a generic bisimulation on the coalgebras underlying `STACK`-algebras. For, the following can be inferred from the `STACK`-constraints using the deduction calculus given in Section 5.1:

```
[Z,N]top.pop(empty) = Z.*
```

[15] The only exception appears to be caused by the use of equationally-defined operations in specifying derived observers. In this case, the formalism introduced here is not sufficiently expressive to support the proofs of certain correctness properties (see the concluding remarks in Example 5.3.22).

```
[Z,N]top.empty = Z.*
[Z,S]rest.pop(empty) = Z.*
[Z,S]rest.empty = Z.*

[Z,N]top.pop(push(s,n)) = Z.z if [Z,N]top.s = Z.z
[Z,N]top.s = Z.z if [Z,N]top.s = Z.z
[Z,N]top.pop(push(s,n)) = N.n' if [Z,N]top.s = N.n'
[Z,N]top.s = N.n' if [Z,N]top.s = N.n'
[Z,S]rest.pop(push(s,n)) = Z.z if [Z,S]rest.s = Z.z
[Z,S]rest.s = Z.z if [Z,S]rest.s = Z.z
[Z,S]rest.pop(push(s,n)) = S.s' if [Z,S]rest.s = S.s'
[Z,S]rest.s = S.s' if [Z,S]rest.s = S.s'
```

Moreover, for $A \in$ Alg(STACK), R_A relates the interpretations in A of the lhs and rhs of the two equations. It therefore follows by the remarks in Section 3.3.1 that $A \models^{\mathsf{b}}$ `pop(empty) = empty` and $A \models^{\mathsf{b}}$ `pop(push(s,n)) = s` hold for any STACK-algebra A.

A state invariant for stacks is captured by the coequations:

```
[Z,S]rest = [Z,S']rest if ([Z,N]top,Z)
[Z,N]top = [Z,N']top if ([Z,S]rest,Z)
```

Proving that the above coequations hold, up to reachability, in all STACK-algebras requires further insights into the notion of reachability under `empty`, `push` and `pop`. For, the fact that these coequations hold in a state `s` does not guarantee that they hold in `pop(s)`, and therefore straightforward induction can not be used to show that the coequations hold in all reachable states. However, the observation that the equations:

```
pop(empty) = empty
pop(push(s,n)) = s
```

hold, up to bisimulation, in STACK-algebras allows one to reduce proving that the stack invariant holds up to reachability under `empty`, `push` and `pop` to proving that the stack invariant holds up to reachability under `empty` and `push` only. For, from the satisfaction of the above equations, one can infer that any Stack-state reachable under `empty`, `push` and `pop` is bisimilar to a Stack-state reachable under `empty` and `push` only. Then, the satisfaction of the stack invariant follows from this invariant holding in `empty` and being preserved by `push`, together with the observation that the coequations defining the invariant hold in a state `s` whenever they hold in a state bisimilar to `s`.

An alternative way to prove that the stack invariant holds, up to reachability, in STACK-algebras is to use the technique described in Section 3.3.1, according to which it suffices to show that the invariant holds in the states of certain subalgebras. Here, the subalgebras can be taken to consist of those states which are bisimilar to states reachable under `empty` and `push`. (The closure of such sets of states under the application of `empty` and `push` is a consequence of the definition of these sets, while their closure under the application of `pop` is a consequence of the fact that the previously-mentioned equations hold, up to bisimulation, in all STACK-algebras.) Then, the coequations defining the stack invariant hold in the states of the subalgebras thus defined. (On the one hand, the coequations hold in states reachable under `empty` and `push`, and on the other hand, their satisfaction in a state

s implies their satisfaction in any state which is bisimilar to s.) As a result, STACK-algebras satisfy, up to reachability, the coequations defining the stack invariant.

Example 5.3.22. Consider the object signature of bounded stacks given in Example 5.3.10 (denoted BSTACK in what follows). Since no constructors or observers of type Stack are added to the object signature of stacks by the object signature of bounded stacks, it follows that the inclusion of the object signature of stacks into the object signature of bounded stacks defines a horizontal object signature morphism.

The depth of a bounded stack of size m should not exceed m. This state invariant for bounded stacks can be formalised using the coequation:

```
[[T,F]<(m+1)]depth = [T]!
```

with ! : BStack $\rightarrow$ 1 denoting the constant observer defined by:

```
[T]!.b = T.*
```

One can then use induction to prove that the above coequation holds, up to reachability, in all BSTACK-algebras. First, the coequation holds in empty, since the following hold:

```
[[T,F]<(m+1)]depth.empty = T.*
[T]!.empty = T.*
```

Now assume that the coequation holds in b. This immediately results in the condition [[T,F]<(m+1)]depth.X = T.t holding for b. The fact that the coequation also holds in push(b,n) then follows by case analysis from:

```
[[T,F]<(m+1)]depth.push(b,n) = T.t
     if [[T,F]<m]depth.b = T.t and [D]depth.b = D.d
```

and:

```
[[T,F]<(m+1)]depth.push(b,n) = T.t
     if [[T,F]<m]depth.b = F.f and [D]depth.b = D.d
     and [[T,F]<(m+1)]depth.b = T.t
```

together with:

```
[T]!.push(b,n) = T.*
```

For, the first constraint defining [N]depth.push(b,n) together with:

```
[T,F]<(m+1).s(d) = T.t if [T,F]<m.d = T.t
```

yield, by **substitution$_2$**:

```
[[T,F]<(m+1)]depth.push(b,n) = T.t
     if [[T,F]<m]depth.b = T.t and [D]depth.b = D.d
```

while the second constraint defining `[N]depth.push(b,n)` together with:

```
[T,F]<(m+1).d = T.t if [T,F]<m+1.d = T.t
```

yield, again by **substitution$_2$**:

```
[[T,F]<(m+1)]depth.push(b,n) = T.t
     if [[T,F]<m]depth.b = F.f and [D]depth.b = D.d
     and [[T,F]<m+1]depth.b = T.t
```

Also, the definition of `!` yields, by **substitution$_1$**:

```
[T]!.push(b,n) = T.*
```

The fact that the coequation holds in `push(b,n)` now follows by noting that the equation `t = *` with `t:1` holds in the underlying data coalgebra, and that the conditions:

```
[[T,F]<m]depth.b = T.t and [D]depth.b = D.d
```

and:

```
[[T,F]<m]depth.b = F.f and [D]depth.b = D.d
 and [[T,F]<(m+1)]depth.b = T.t
```

cover all possible behaviours for `b`, provided that the condition `[[T,F]<(m+1)]depth.X = T.t` holds in `b`.

The fact that the coequation holds in `pop(b)` whenever it holds in `b` follows, again, by case analysis. On the one hand, the first constraint defining `[N]depth.pop(b)` together with:

```
[T,F]<(m+1).0 = T.*
```

yield, by **substitution$_2$**:

```
[[T,F]<(m+1)]depth.pop(b) = T.* if [[F,D]p]depth.b = F.f
```

On the other hand, the second constraint defining `[N]depth.pop(b)` together with:

```
[T,F]<(m+1).d = T.t if [T,F]<(m+1).d = T.t
```

yield, again by **substitution$_2$**:

```
[[T,F]<(m+1)]depth.pop(b) = T.t
     if [[T,D]p]depth.b = D.d and [[T,[T,F]<(m+1)]p]depth.b = T.t
```

That is:

```
[[T,F]<(m+1)]depth.pop(b) = T.t
     if [[T,D]p]depth.b = D.d and [[T,F]<(m+2)]depth.b = T.t
```

This, together with:

```
[[T,F]<(m+2)]depth.b = T.t if [[T,F]<(m+1)]depth.b = T.t
```

(following from the fact that the coequation:

```
[T,F]<(m+2) = [T,F]<(m+1) if ([T,F]<(m+1),T)
```

holds in the underlying data coalgebra, see Example 5.2.13, and therefore by **closure**, see Proposition 5.3.20, the coequation:

```
[[T,F]<(m+2)]depth = [[T,F]<(m+1)]depth if ([[T,F]<(m+1)]depth,T)
```

holds in any BSTACK-algebra) yield:

```
[[T,F]<(m+1)]depth.pop(b) = T.t
     if [[T,D]p]depth.b = D.d and [[T,F]<(m+1)]depth.b = T.t
```

Finally, the definition of ! yields, by **substitution**$_1$:

```
[T]!.pop(b) = T.*
```

The fact that the coequation holds in pop(b) whenever it holds in b now follows by noting that the equation t = * with t:1 holds in the underlying data coalgebra, and that the conditions:

```
[[F,D]p]depth.b = F.f
```

and:

```
[[T,D]p]depth.b = D.d and [[T,F]<(m+1)]depth.b = T.t
```

cover all possible behaviours for b, provided that the condition [[T,F]<(m+1)]depth.X = T.t holds in b.

We conclude this example with a remark on the definition of the derived observer full?. An alternative approach to specifying bounded stacks would have been to define a derived observer full : BStack → Bool using the following constraint:

```
[B]full.b = B.not(d<m) if [D]depth.b = D.d
```

and use conditions of form [[T,F]?]full.b = T.t and respectively [[T,F]?]full.b = F.f in the definition of push. In this case, the invariant for bounded stacks could have been formulated using a derived observer inv : BStack → Bool defined by the constraint:

```
[B]inv.b = B.d<=m if [D]depth.b = D.d
```

with the algebraic operations < : Nat Nat → Bool and <= : Nat Nat → Bool being defined similarly to the operation == : Nat Nat → Bool of Example 5.2.14. However, in this case the equational definitions of not and < would have prevented a constraint of form:

```
[B]inv.push(b,n) = B.true
     if [[T,F]?]full.b = F.f and [D]depth.b = D.d
```

from being derived from:

```
[B]inv.push(b,n) = B.s(d)<=m
     if [[T,F]?]full.b = F.f and [D]depth.b = D.d
```

(see also Remark 5.3.19). A possible explanation of this fact is that the above specification of bounded stacks attempts to use *algebraic* operations in order to perform case analysis (namely in the definition of `push`).

Example 5.3.23. Recall from Example 4.2.40 that lists having the property that any two adjacent elements are different from each other could not be specified in a purely coalgebraic setting. This property of lists can, however, be formalised provided that the underlying data is allowed to carry algebraic structure. This is illustrated here using an object signature which consists of a hidden sort `List`, observers `first` : `List` $\to$ `1` `Nat`, `rest` : `List` $\to$ `1` `List`, constructors `empty` : $\to$ `List`, `next` : `List` $\to$ `List` (with `1` and `Nat` denoting visible sorts), and constraints:

```
[Z,N]first.empty = Z.*
[Z,L]rest.empty = Z.*

[Z,N]first.next(l) = N.0 if [Z,N]first.l = Z.z
[Z,N]first.next(l) = N.s(n) if [Z,N]first.l = N.n
[Z,L]rest.next(l) = L.l
```

That is, the constructor `next` appends a value in front of its argument, with the value appended being given either by the successor of the first element of the list representing the argument (if this list is not empty) or by `0` (otherwise).

In order to specify the above-mentioned property of lists, we introduce a derived observer `inv` : `List` $\to$ `Bool`, defined using the constraints:

```
[B]inv.l = B.true
     if [Z,[Z,N]first]rest.l = Z.z
[B]inv.l = B.not(f==s)
     if [Z,N]first.l = N.f and [Z,[Z,N]first]rest.l = N.s
[B]inv.l = B.false
     if [Z,N]first.l = Z.z and [Z,[Z,N]first]rest.l = N.s
```

(The presence of the last of the above constraints is required by the definition of object signatures, since the list invariant, as given e.g. in Example 4.1.29, can not be assumed to hold in an arbitrary `List`-state[16].)

Now, if `true` : `List` $\to$ `Bool` denotes the constant observer defined by:

```
[B]true.l = B.true
```

then the above-mentioned property of lists is captured by the coequation:

```
[B]inv = [B]true
```

(Since the coequation is required to hold in *all* reachable `List`-states, and since observers preserve reachability, it suffices to require that the first and second element of *any* list are different from each other.)

The proof of the above invariant is by induction on `List`. First, the invariant holds in `empty`, since:

```
[Z,[Z,N]first]rest.empty = Z.*
```

[16] One can, however, show that the invariant holds in all reachable states.

can be inferred, and therefore:

```
[B]inv.empty = B.true
```

holds. Now assume that the invariant holds in `l`. The fact that it also holds in `next(l)` follows from:

```
[B]inv.next(l) = B.true if [Z,N]first.l = Z.z
```

and:

```
[B]inv.next(l) = B.not(s(n)==n) if [Z,N]first.l = N.n
```

(with each of the above two constraints following by **substitution$_1$** and **substitution$_2$** from the definitions of `inv` and `next`), together with the equation:

```
not(s(x)==x) = true
```

holding in the underlying data coalgebra (see Example 5.2.14).

5.3.3 Specifying Inheritance

The rôle of inheritance in object-oriented programming is twofold. On the one hand, inheritance is a mechanism for classifying objects according to their structure and functionality, with each class inheriting the structure and functionality of its ancestors, and possibly adding new structure and/or functionality[17]. On the other hand, inheritance is a mechanism for the reuse of class implementations, with the implementation of a class being made available to its descendants. A realistic approach to specifying inheritance should therefore account both for its classification aspect and for its reusability aspect.

To a certain extent, the reusability aspect of inheritance can be captured within an algebraic framework (see e.g. [Cîr98]). However, capturing the classification aspect of inheritance within such a framework turns out to be more difficult, as algebraic operations are not sufficiently general to formalise observational features such as the classification of instances of a class according to their least type (with the ordering on types being determined by the inheritance relationship). In this respect, a combined algebraic-coalgebraic framework appears to be more promising. This section briefly illustrates how the previously-introduced formalism can be used to specify classes in the presence of inheritance.

In the following, it is assumed that class declarations only contain declarations of *constructors*, *direct attributes* (corresponding to instance variables), *indirect attributes* (corresponding to instance methods with no arguments which return a value but do not result in a change of state) and *methods* (corresponding to instance methods which do not return any value but which result in a change of state). The reason for not allowing side-effects (i.e. instance methods which return a value and at the same time result in a change of state) or parameterised attributes (i.e. instance methods with arguments which at the same time return a value) is the need for a complete separation between the computational aspect and the observational aspect of object states. (Under this

[17] This results in inheritance hierarchies.

restriction, side-effects can still be encoded by using both a method and an indirect attribute, and possibly some additional direct attributes, while parameterised attributes can be captured by several unparameterised attributes, namely one for each possible combination of values for the parameters.)

We now consider an arbitrary collection of classes related to each other by inheritance relationships, and illustrate how such a collection can be specified in our formalism. Only *single inheritance* is allowed, that is, each class can only have one direct ancestor (or *superclass*).

Each class c is captured by two hidden sorts, c and $\bar{c}$, whose intended denotations are the collection of instances of c, *excluding* and respectively *including* instances of its subclasses. The use of two different sorts will later result in the notion of a *least class* (i.e. most specific class) of an object being captured in the formalism.

We begin by specifying the behaviour associated to hidden sorts c. First, the observers associated to such a sort are as follows:

1. an observer $\delta : c \to \bar{c}_1 \dots \bar{c}_n$ for each class c and each direct attribute δ, having result type $c_1 \dots c_n$, declared by c
2. an observer $\uparrow_c$: $c \to c'$ (capturing the inheritance of the structure of c' by c), for each class c with direct superclass c'
3. a derived observer $\delta : c \to \bar{c}_1 \dots \bar{c}_n$ defined by the coequation:

$$[\overline{Z}]\delta = [[\overline{Z}]\delta] \uparrow_c$$

   for each class c with direct superclass c' and each direct attribute δ, having result type $c_1 \dots c_n$, declared by c' or by one of its superclasses and not overridden by c (If δ is last declared by a superclass of c', then a derived observer $\delta : c' \to \bar{c}_1 \dots \bar{c}_n$ is also being defined.)
4. a derived observer $\delta : c \to \bar{c}_1 \dots \bar{c}_n$ for each class c and each indirect attribute δ, having result type $c_1 \dots c_n$, either declared by c or inherited from a superclass and not overridden by c (Since the values of indirect attributes depend solely on the values of direct attributes, it follows that indirect attributes can be captured by derived observers.)

Next, the constructors associated to a sort c are as follows:

1. a (derived) constructor $\gamma_c : \bar{c}_1 \dots \bar{c}_m \to c$ for each class c and each constructor, having argument types $c_1, \dots, c_m$, declared by c
2. a (derived) constructor $\sigma_c : c\, \bar{c}_1 \dots \bar{c}_m \to c$ for each class c and each method, having argument types $c_1, \dots, c_m$, either declared by c or inherited from a superclass and not overridden by c.

The resulting constructor is derived or not, depending on whether the implementation of the constructor/method in question uses calls to other constructors/methods of the same object or defines the effect of the constructor/method on the instance variables.

The derived observers induced by indirect attributes and the (derived) constructors induced by constructors and methods are subject to constraints determined by the desired behaviour for the class c. In writing these constraints, a useful observation is that the specifications of the superclass

versions of inherited operations are available via the observers $\uparrow_c$; this provides support for the reuse of specifications. (Nevertheless, the behaviour of inherited operations on objects of the subclass must be specified through constraints, even when these operations are not overridden in the subclass.)

We now specify the behaviour associated to hidden sorts $\bar{c}$. The observers associated to such sorts are of form $?_{\bar{c}} : \bar{c} \to c\ \bar{s}_1 \ldots \bar{s}_n$, where c is a class with direct subclasses $s_1, \ldots, s_n$. Such an observer captures the classification of instances of c according to their least class: the least class of an object of type $\bar{c}$ is either c or (a subclass of) one of $s_1, \ldots, s_n$, in which case destructors of form $?_{\bar{s}_i}$ must also be used in order to determine the least class of the given object. In addition, derived observers $\delta : \bar{c} \to \bar{c}_1 \ldots \bar{c}_n$ with δ an attribute of c (either declared by c or inherited from a superclass and not overridden by c) are specified using coequations:

$$[\overline{Z}]\delta = [\ [\overline{Z}]\delta\ ,\ [[[\overline{Z}]\delta] \uparrow_{s_1}, \ldots]?_{\bar{s}_1}\ ,\ \ldots\ ,\ [[[\overline{Z}]\delta] \uparrow_{s_n}, \ldots]?_{\bar{s}_n}\ ]?_{\bar{c}}$$

if δ denotes a direct attribute[18], and respectively:

$$[\overline{Z}]\delta = [[\overline{Z}]\delta, [\overline{Z}]\delta, \ldots, [\overline{Z}]\delta]?_{\bar{c}}$$

if δ denotes an indirect attribute. These definitions account for the fact that a call to an attribute results in the *least specific*, and respectively the *most specific* version of that attribute being selected, depending on whether the attribute is implemented by an instance variable or by an instance method.

The constructors associated to a sort $\bar{c}$ are as follows:

1. a constructor $!_c : c \to \bar{c}$ for each class c
2. a constructor $!_{\bar{s}} : \bar{s} \to \bar{c}$ for each class c and each direct subclass s of c
3. a (derived) constructor $\sigma_{\bar{c}} : \bar{c}\ \bar{c}_1 \ldots \bar{c}_m \to \bar{c}$ for each method σ (either declared by c or inherited from a superclass and not overridden by c) with argument types $c_1, \ldots, c_m$

(Again, the constructor associated to a method σ is derived precisely when the implementation of σ in c uses calls to other methods of the same object.) These constructors and observers are subject to the following constraints:

1. $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.!_c(C) = Z_0.C$
   $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.!_{\bar{s}_i}(S) = Z_i.S,\ \ i = 1, \ldots, n$
2. $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.\sigma_{\bar{c}}(C, \overline{C}) = Z_0.\sigma_c(Y, \overline{C})$ if $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.C = Z_0.Y$
   $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.\sigma_{\bar{c}}(C, \overline{C}) = Z_i.\sigma_{\bar{s}_i}(Y, \overline{C})$ if $[Z_0, Z_1, \ldots, Z_n]?_{\bar{c}}.C = Z_i.Y,\ \ i = 1, \ldots, n$
   provided that $\sigma_{\bar{c}}$ is not derived. (If $\sigma_{\bar{c}}$ is derived, then the constraints defining it are determined by the desired behaviour for the corresponding method.)

(If c is an *abstract class*, i.e. a class with no instances which is only used for classification purposes, then an observer $?_{\bar{c}} : \bar{c} \to \bar{s}_1 \ldots \bar{s}_n$ and constructors $!_{\bar{s}_i} : \bar{s}_i \to \bar{c}$ with $i \in \{1, \ldots, n\}$, subject to similar constraints, should be used instead.)

[18] This definition uses successive applications of operations of form $?_s$ followed by successive applications of operations of form $\uparrow_s$ to access the least specific version of δ on a given object, depending on its least class.

It follows immediately that the equations:

$$!_c(\sigma_c(C, \overline{C})) = \sigma_{\bar{c}}(!_c(C), \overline{C})$$
$$!_{\bar{s}_i}(\sigma_{\bar{s}_i}(C, \overline{C})) = \sigma_{\bar{c}}(!_{\bar{s}_i}(C), \overline{C}), \quad \text{for } i = 1, \ldots, n$$

with c having $s_1, \ldots, s_n$ as its direct subclasses, hold, up to bisimulation, in all algebras of an object signature defined using the above constraints.

The approach to specifying inheritance outlined above exploits the observation that, from a structural point of view, inheritance can be captured by aggregation. However, this view is not extended to the computational aspect of inheritance, thus allowing concepts such as "least class" and "dynamic binding" to be accounted for.

If, for a collection of classes, the result types of all the attributes and the argument types of all the constructors and methods are types that have been previously defined, then one can separate the specification of this collection of classes into the specification of hidden sorts c, and respectively the specification of hidden sorts $\bar{c}$. (In this case, as far as the hidden sorts c are concerned, the adding of subclasses can be captured via horizontal object signature morphisms.) On the other hand, if the above is not true, then the hidden sorts c and $\bar{c}$ associated to classes in the collection must be specified all at once. In both cases, the adding of a subclass s to a class c results in a change of the constructors and observers associated to $\bar{c}$, and consequently the hidden sorts $\bar{c}$ associated to classes belonging to the same hierarchy must be specified all at once.

Correctness properties of object behaviour are captured by equations quantified over variables of type $\bar{c}$, and respectively by coequations of type $\bar{c}$. Consequently, proving such properties requires a case analysis on the least type of variables of type $\bar{c}$, and respectively on the result yielded by the observer $?_{\bar{c}}$.

Example 5.3.24. The specification of bounded stacks in Example 5.3.10 only accounts for the reusability aspect of inheritance. A specification which also accounts for its classification aspect would, instead, consist of hidden sorts `Stack`, `Stacks`, `BStack`, `BStacks`, observers:

$\texttt{top} : \texttt{Stack} \to 1\ \texttt{Nat}$
$\texttt{rest} : \texttt{Stack} \to 1\ \texttt{Stacks}$

$\texttt{depth} : \texttt{BStack} \to \texttt{Nat}$
$\texttt{stack} : \texttt{BStack} \to \texttt{Stack}$

$?_{\texttt{Stacks}} : \texttt{Stacks} \to \texttt{Stack BStacks}$

$?_{\texttt{BStacks}} : \texttt{BStacks} \to \texttt{BStack}$

(where $\uparrow_{\texttt{BStack}}$ has been renamed to `stack`), derived observers:

$\texttt{full?} : \texttt{BStack} \to 1\ 1$
$\texttt{top} : \texttt{BStack} \to 1\ \texttt{Nat}$
$\texttt{rest} : \texttt{BStack} \to 1\ \texttt{Stacks}$

$\texttt{top} : \texttt{Stacks} \to 1\ \texttt{Nat}$
$\texttt{rest} : \texttt{Stacks} \to 1\ \texttt{Stacks}$

$\texttt{full?} : \texttt{BStacks} \to 1\ 1$
$\texttt{top} : \texttt{BStacks} \to 1\ \texttt{Nat}$

```
rest : BStacks → 1 Stacks
depth : BStacks → Nat
```

constructors:

```
empty : → Stack
push : Stack Nat → Stack
pop : Stack → Stack

empty : → BStack
push : BStack Nat → BStack
pop : BStack → BStack

!Stack : Stack → Stacks
!BStacks : BStacks → Stacks
push : Stacks Nat → Stacks
pop : Stacks → Stacks

!BStack : BStack → BStacks
push : BStacks Nat → BStacks
pop : BStacks → BStacks
```

and derived constructors[19]:

```
push2 : Stack Nat Nat → Stack

push2 : BStack Nat Nat → BStack

push2 : Stacks Nat Nat → Stacks

push2 : BStacks Nat Nat → BStacks
```

subject to the following coequations and constraints:

```
[Z,N]top.empty = Z.*
[Z,S]rest.empty = Z.*
[Z,N]top.push(s,n) = N.n
[Z,S]rest.push(s,n) = S.!Stack(s)
[Z,N]top.pop(s) = Z.*
      if [Z,[Z,N]top]rest.s = Z.z
[Z,N]top.pop(s) = N.n
      if [Z,[Z,N]top]rest.s = N.n
[Z,S]rest.pop(s) = Z.*
      if [Z,[Z,S]rest]rest.s = Z.z
[Z,S]rest.pop(s) = S.s'
      if [Z,[Z,S]rest]rest.s = S.s'
S.push2(s,n,n') = S.push(push(s,n),n')
```

for the sort Stack,

```
[T,F]full? = [[F,T]<m]depth
[Z,N]top = [[Z,N]top]stack
[Z,S]rest = [[Z,S]rest]stack

[S]stack.empty = S.empty
[N]depth.empty = N.0
```

[19] A method for adding two elements to a stack is introduced here in order to illustrate the use of derived constructors.

```
[S]stack.push(b,n) = S.push(s,n)
      if [T,F]full?.b = F.f and [S]stack.b = S.s
[S]stack.push(b,n) = S.s
      if [T,F]full?.b = T.t and [S]stack.b = S.s
[N]depth.push(b,n) = N.s(d)
      if [T,F]full?.b = F.f and [D]depth.b = D.d
[N]depth.push(b,n) = N.d
      if [T,F]full?.b = T.t and [D]depth.b = D.d
[S]stack.pop(b) = S.pop(s)
      if [S]stack.b = S.s
[N]depth.pop(b) = N.0
      if [[F,D]p]depth.b = F.f
[N]depth.pop(b) = N.d
      if [[F,D]p]depth.b = D.d
B.push2(b,n,n') = B.push(push(b,n),n')
```

for the sort BStack,

```
[Z,N]top = [[Z,N]top,[[[Z,N]top]stack]?BStacks]?Stacks
[Z,S]rest = [[Z,S]rest,[[[Z,S]rest]stack]?BStacks]?Stacks

[S,B]?Stacks.!Stacks(s) = S.s
[S,B]?Stacks.!BStack(b) = B.b
[S,B]?Stacks.push(s,n) = S.push(s',n)
      if [S,B]?Stacks.s = S.s'
[S,B]?Stacks.push(s,n) = B.push(b,n)
      if [S,B]?Stacks.s = B.b
[S,B]?Stacks.pop(s) = S.pop(s')
      if [S,B]?Stacks.s = S.s'
[S,B]?Stacks.pop(s) = B.pop(b)
      if [S,B]?Stacks.s = B.b
S.push2(s,n,n') = S.push(push(s,n),n')
```

for the sort Stacks, and:

```
[T,F]full? = [[T,F]full?]?BStacks
[Z,N]top = [[Z,N]top]?BStacks
[Z,S]rest = [[Z,S]rest]?BStacks
[N]depth = [[N]depth]?BStacks

[B]?BStacks.!BStack(b) = B.b
[B]?BStacks.push(b,n) = B.push(b',n)
      if [B]?BStacks.b = B.b'
[B]?BStacks.pop(b) = B.pop(b')
      if [B]?BStacks.b = B.b'
B.push2(b,n,n') = B.push(push(b,n),n')
```

for the sort BStacks.

5.3.4 Related Work

This section briefly compares the approach to specifying objects presented here with the ones in [GM97, GM00, Jac96c, Jac96a, Jac97, Rößß00, Jac00]. A specification of stacks is used to emphasise the differences between our approach and the above-mentioned ones.

The following hidden algebraic specification of stacks of natural numbers is taken from [GM00].

```
th STACK is
     sort Stack .
     pr DATA .
     op empty : -> Stack .
     op push : Nat Stack -> Stack .
     op top : Stack -> Nat .
     op pop : Stack -> Stack .
     var S : Stack .
     var I : Nat .
     eq top(push(I,S)) = I .
     eq pop(empty) = empty .
     eq pop(push(I,S)) = S .
endth
```

The models of this specification are algebras of the hidden signature consisting of a sort `Stack` and operation symbols `empty`, `push`, `top` and `pop`, which, in addition, behaviourally satisfy the given equations. A first difference between the above specification and the specification given in Example 5.3.1 stands in the way of specifying empty stacks. The algebraic nature of the approach in [GM00] prevents operations with structured result type, such as `top` : `Stack` $\to$ `1 Nat`, to be accommodated by hidden signatures. As a result, the undefinedness of the operation denoted by `top` on empty stacks is captured in [GM00] by the absence of an equation defining a value for `top(empty)`. But this underspecification results in models being able to use *any* value (rather than *no* value) as an interpretation for `top(empty)`. Another difference between the approach in [GM00] and the one here is that, in [GM00], the contexts used for observation are defined in terms of the entire signature, whereas here, only the observers are used in this sense. As a result, underspecification is possible in [GM00] (this resulting in a more complex notion of behavioural equivalence), whereas here, all the constructors must be fully specified (in order to guarantee their well-behavedness w.r.t. bisimilarity). Nevertheless, the notion of behavioural equivalence induced by a hidden specification is typically given by indistinguishability by contexts over a subsignature[20], with the well-behavedness of the remaining operations having to be inferred from the equations in the specification.

As far as coalgebraic approaches to specification are concerned, their main drawback stands in their inability to specify arbitrary constructors. This becomes increasingly important as structured specifications are considered, since the functionality of objects may include methods taking other objects as arguments. Also, the resulting notion of observational indistinguishability is usually unnecessarily complex, as no distinction is made between structural and computational features. For instance, the notion of bisimilarity associated to the stack specification given in Example 3.1.46

[20] In the above example, this subsignature is given by `top` and `pop`.

is similar to the notion of behavioural equivalence associated to the above hidden specification of stacks.

Using the approach in [Röß00], a specification of stacks of natural numbers involves modal operators $\texttt{<push}(n)\texttt{>}$, <popF>, <popS>, <topF> and <topS>. (This particular choice of modal operators is obtained by considering coalgebras of the endofunctor G : Set $\to$ Set given by $\mathrm{G}X = X^{\mathbb{N}} \times (1 + X) \times (1 + \mathbb{N})$.) The following modal formula is then used to specify the effect of adding an element to a stack:

$$\texttt{<push}(n)\texttt{><topS>}n$$

with $n \in \mathbb{N}$. (That is, the resulting stack has a top element, and moreover, this element is precisely the element that was previously added to the stack.) However, in order to specify the property of stacks stating that extracting an element from the stack immediately after adding an element to the stack yields a stack which is observationally the same as the original stack, one has to allow quantification over formulae. Specifically, the previously-mentioned property can be specified by:

$$\texttt{<push}(n)\texttt{><popS>}\varphi \leftrightarrow \varphi$$

with φ denoting an arbitrary formula. (A similar approach is used in [Jac00] for specifying bounded stacks.) In addition, no account can be given by such an approach to even basic object constructors such as the stack constructor yielding an empty stack. This appears to suggest that, while modal logic is suitable for specifying structural/observational properties of systems (see Section 4.2.7 for examples of such properties), correctness properties regarding the behaviour of systems, including the equivalence of certain computations are better captured within equational logic. An integration of modal and equational approaches to specification constitutes the subject of future work. The main ideas underlying such an integration are briefly outlined in Section 6.2.

6 Conclusions

This chapter summarises the approach presented in Chapters 3, 4 and 5, and briefly outlines possible directions for future research.

6.1 Summary of Results

The underlying idea of this work was to combine the complementary contributions of algebra and coalgebra to specification, in order to increase the expressiveness of unilateral (either purely algebraic or purely coalgebraic) approaches to the specification of state-based, dynamical systems on the one hand, and to provide simpler and more natural formalisations of the relevant concepts on the other. Our approach was to distinguish between computational and observational features in such systems, and to use algebra and respectively coalgebra for the formalisation of these features and of the concepts deriving from them. In particular, such an approach yielded an algebraically-defined notion of reachability under computations, as well as a coalgebraically-defined notion of indistinguishability by observations. This, in turn, resulted in the availability of inductive and coinductive techniques for proving properties about reachable and respectively observable behaviours.

The compatibility between observational and computational features in structures having both an observational and a computational component has been shown to arise naturally from a layered approach to the specification of such structures. While allowing for a smooth integration of the two categories of features, the use of liftings of monads/comonads to categories of coalgebras/algebras also amounts to fully specifying the behaviour of the system being considered. One can argue that such an approach is somewhat restrictive, as it does not support underspecification. However, similar (but less natural) restrictions need to be imposed, either on models [Dia98, HB99] or on specifications [RG00], in any non-layered approach to system specification which accommodates non-trivial constructors, in order to ensure the preservation of observational equivalence relations by the constructors (and therefore the soundness of the associated proof techniques).

Equational sentences have been used to formalise correctness properties of the specified behaviours. The choice of an equational approach was motivated, on the algebraic side, by the existence of characterisability results as well as of a complete deduction calculus for equational logic, and on the coalgebraic side, by an attempt to unify some of the existing equational coalgebraic approaches to system specification, as well as to investigate the suitability of equational specification in a coalgebraic setting. This investigation resulted in an abstract coalgebraic framework for the specification of structures involving observation, as well as in a concrete formalism for the specification of observational structures allowing for a choice in the result type of observations, for which a sound and complete deduction calculus has been formulated.

Although not sufficiently expressive to characterise arbitrary covarieties, many-sorted coequations

have been shown capable of capturing in a concise manner observational properties quantified over state spaces. In particular, structural properties of state-based systems, including various dependencies between their components have been successfully formalised using coequations.

A suitable instantiation of the abstract specification framework has yielded a formalism for the specification of objects, with variants of many-sorted algebra and many-sorted coalgebra being used to specify object functionality and respectively object structure. The presence of both an algebraic and a coalgebraic component has resulted in an increased expressiveness compared to unilateral approaches, both w.r.t. the kinds of behaviours that can be specified, and w.r.t. the kinds of correctness properties that can be formulated. In particular, systems whose structure is variable have been shown to be specifiable in the resulting formalism, and state invariants involving the undefinedness of certain system components have been shown to be supported by this formalism.

6.2 Directions for Future Research

The abstract equational framework introduced in Chapter 3 has only been instantiated to (extended) polynomial endofunctors. Other possible instantiations, including the use of certain powerset functors for the coalgebraic component (as in [TP97] or [CHM99]) should also be considered. Such functors are not ω^{op}-continuous, and therefore do not give rise to abstract cosignatures. However, the approach presented in Section 3.1 can easily be generalised to include functors which are not necessarily ω^{op}-continuous, but for which final/cofree coalgebras exist (with endofunctors of form $\mathcal{P}_{\mathsf{k}}(L \times \_), \mathcal{P}_{\mathsf{k}}(\_)^L$: Set $\to$ Set with k an arbitrary cardinal being instances of such functors[1]). The requirement regarding the existence of cofree coalgebras results in such endofunctors inducing comonads. As a consequence, many of the results in Section 3.1 generalise to the new setting. The only results that do not immediately generalise are the existence of largest subcoalgebras satisfying an enumerable set of coequations (Proposition 3.1.33)[2] and the compositionality results in Section 3.1.3 (Theorems 3.1.58 and 3.1.60)[3]. However, it is not yet clear whether coequations over cosignatures involving powerset functors are able to express the kinds of properties one expects to formulate about transition systems, or whether other kinds of formulae (such as modal formulae) should be considered in this case.

Another issue which deserves some study is the use of an alternative notion of sentence in formalising properties of observational structures. Possible candidates in this sense are the laws of [Fok96] (see Section 3.3.8) and the formulae of a multi-modal logic in the style of [Rößß00] or [Jac00] (see Section 4.2.7). In particular, an approach based on modal logic would benefit from the existence of Birkhoff-style characterisability results, not only at an abstract level but also in concrete formalisms. In addition, since a distinction would still be made between observational and computational features, and since the modal operators of such a logic would only be induced by the observational features, we

[1] The endofunctor $\mathcal{P}_{\mathsf{k}}(\_)$: Set $\to$ Set takes a set X to the set of all subsets of X of cardinality smaller than k.

[2] Note however, that a weaker version of Proposition 3.1.33, stating the existence of largest subcoalgebras satisfying single coequations, can be formulated. The proof of this result uses the fact that abstract cosignatures induce comonads, and is similar to the proof of Proposition 3.2.27.

[3] More specific constraints than simply the existence of cofree coalgebras for the endofunctors defining abstract cosignatures are likely to be necessary in order to guarantee the existence of finite colimits in the category of abstract cosignatures and strong abstract cosignature morphisms.

expect the resulting notion of sentence to be less complex than the sentences typically used in [Röß00] or [Jac00] (as observational features do not give rise to as many cyclic structures as a combination of observational and computational features does). For instance, (the strong versions of) the modal operators associated to the type of stacks would be: `<topF>`, `<topS>`, `<restF>`, `<restS>`. These modal operators are determined by the endofunctor G : Set $\to$ Set, $(\mathsf{G}X) = (1 + \mathbb{N}) \times (1 + X)$ using the approach in [Röß00] (see also Section 4.2.7). As already noted in Section 4.2.7, the stack invariant would in this case be formalised by the modal formula:

`<restF>*` $\leftrightarrow$ `<topF>*`

Finally, the formalism developed in Chapter 5 could constitute the basis of a specification language for objects. However, more elaborate examples of object specifications should be considered, as part of a case study aimed at further justifying the suitability of this formalism for object specification as well as at comparing it with existing formalisms. In particular, the specification of certain Java class libraries would provide a suitable setting for comparing our approach with the one in [Jac98].

Bibliography

[AHS90] Jiří Adámek, Horst Herrlich, and George E. Strecker. *Abstract and Concrete Categories*. John Wiley and Sons, 1990.

[AM74a] Michael A. Arbib and Ernest G. Manes. Foundations of system theory: Decomposable systems. *Automatica*, 10:285–302, 1974.

[AM74b] Michael A. Arbib and Ernest G. Manes. Machines in a category: an expository introduction. *SIAM Review*, 16(2):163–192, 1974.

[Bar93] Michael Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114(2):299–315, 1993.

[BB91] Gilles Bernot and Michel Bidoit. Proving correctness of algebraically specified software: Modularity and observability issues. In *Proceedings of the Second International Conference on Algebraic Methodology and Software Technology*, pages 139–161. The University of Iowa, 1991.

[BH76] Bernhard Banaschewski and Horst Herrlich. Subcategories defined by implications. *Houston Journal of Mathematics*, 2(2):149–171, 1976.

[BH95] Michel Bidoit and Rolf Hennicker. Behavioural theories. In *Recent Trends in Data Type Specification*, volume 906 of *Lecture Notes in Computer Science*, pages 153–169. Springer, 1995.

[BH96] Michel Bidoit and Rolf Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165(1):3–55, 1996.

[Bor94a] Francis Borceux. *Handbook of Categorical Algebra*, volume I. Cambridge University Press, Cambridge, 1994.

[Bor94b] Francis Borceux. *Handbook of Categorical Algebra*, volume II. Cambridge University Press, Cambridge, 1994.

[BR85] Rod Burstall and D.E. Rydeheard. Monads and theories: a survey for computation. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 575–605. Cambridge University Press, 1985.

[BW90] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.

[BW95] Michael Barr and Charles Wells. Category theory for computing science. The electronic supplement, 1995.

[CGRH98] Andrea Corradini, Martin Große-Rhode, and Reiko Heckel. Structured transition systems as lax coalgebras. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 23–42. Elsevier Science, 1998.

[CHM99] Andrea Corradini, Reiko Heckel, and Ugo Montanari. From SOS specifications to structured coalgebras: How to make bisimulation a congruence. In B. Jacobs and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 149–172. Elsevier Science, 1999.

[Cîr98] Corina Cîrstea. Coalgebra semantics for hidden algebra: parameterised objects and inheritance. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques*, volume 1376 of *Lecture Notes in Computer Science*, pages 174–189. Springer, 1998.

[Cîr99a] Corina Cîrstea. A coequational approach to specifying behaviours. In B. Jacobs and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 173–194. Elsevier Science, 1999.

[Cîr99b] Corina Cîrstea. Semantic constructions for hidden algebra. In J.L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques*, volume 1589 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 1999.

[Cîr00] Corina Cîrstea. An algebra-coalgebra framework for system specification. In B. Jacobs and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 81–112. Elsevier Science, 2000.

[Cor98] Andrea Corradini. A completeness result for equational deduction in coalgebraic specification. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques*, volume 1376 of *Lecture Notes in Computer Science*, pages 190–205. Springer, 1998.

[Dia98] Răzvan Diaconescu. Behavioural coherence in object-oriented algebraic specification. Technical Report IS-RR-98-0017F, Japan Advanced Institute for Science and Technology, 1998.

[EBO93] Hartmut Ehrig, Michael Baldamus, and Fernando Orejas. New concepts for amalgamation and extension in the framework of specification logics. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification*, volume 655 of *Lecture Notes in Computer Science*, pages 199–221. Springer, 1993.

[EKMP82] Hartmut Ehrig, Hans-Jörg Kreowski, Bernd Mahr, and Peter Padawitz. Algebraic implementation of abstract data types. *Theoretical Computer Science*, 20(3):209–263, 1982.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.

[EM90] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specifications 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1990.

[Fok96] Maarten M. Fokkinga. Datatype laws without signatures. *Mathematical Structures in Computer Science*, 6:1–32, 1996.

[FS90] Peter Freyd and Andrej Scedrov. *Categories, Alegories*. North Holland, 1990.

[GB84] Joseph Goguen and Rod Burstall. Some fundamental algebraic tools for the semantics of computation. Part 1: Comma categories, colimits, signatures and theories. *Theoretical Computer Science*, 31(2):175–209, 1984.

[GB92] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GD94] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 1–29. Springer, 1994.

[GM81] Joseph Goguen and José Meseguer. Completeness of many-sorted equational logic. *SIGPLAN Notices*, 16(7):24–32, 1981. Extended version as Technical Report CSLI-84-15, Center for the Study of Language and Information, Stanford University, 1984.

[GM82] Joseph Goguen and José Meseguer. Universal realization, persistent interconnection and implementation of abstract modules. In M. Nielsen and E.M. Schmidt, editors, *Proceedings, 9th International Conference on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Mathematics*, pages 265–281. Springer, 1982.

[GM97] Joseph Goguen and Grant Malcolm. A hidden agenda. Technical Report CS97-538, UCSD, 1997.

[GM99] Joseph Goguen and Grant Malcolm. Hidden coinduction. *Mathematical Structures in Computer Science*, 9(3):287–319, 1999.

[GM00] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, 2000.

[Gog75] Joseph Goguen. Semantics of computation. In E. Manes, editor, *Category Theory Applied to Computation and Control*, volume 25 of *Lecture Notes in Computer Science*, pages 151–163. Springer, 1975.

[Gog91] Joseph Goguen. Types as theories. In G.M. Reed, A.W. Roscoe, and R.F. Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford University Press, 1991.

[Gol] Robert Goldblatt. What is the coalgebraic analogue of Birkhoff's variety theorem? To appear in Theoretical Computer Science.

[GS98] H. Peter Gumm and Tobias Schröder. Covarieties and complete covarieties. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 43–56. Elsevier Science, 1998.

[GTW78] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In R. Yeh, editor, *Current Trends in Programming Methodology. Data Structuring*, volume 4, pages 80–149. Prentice-Hall, 1978.

[Gum99] H. Peter Gumm. Elements of the general theory of coalgebras. LUATCS'99, 1999.

[HB99] Rolf Hennicker and Michel Bidoit. Observational logic. In *Proceedings, AMAST '98*, volume 1548 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1999.

[Hen90] Rolf Hennicker. Context induction: a proof principle for behavioural satisfaction. In A. Miola, editor, *Proceedings, International Symposium on the Design and Implementation of Symbolic Computation Systems*, volume 429 of *Lecture Notes in Computer Science*, pages 101–110. Springer, 1990.

[HK99] Rolf Hennicker and Alexander Kurz. (Ω, Ξ)-logic: On the algebraic extension of coalgebraic specifications. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 195–211. Elsevier Science, 1999.

[HR95] Ulrich Hensel and Horst Reichel. Defining equations in terminal coalgebras. In E. Astesiano, G.Reggio, and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, volume 906 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 1995.

[Jac95] Bart Jacobs. Mongruences and cofree coalgebras. In V.S. Alagar and M. Nivat, editors, *Algebraic Methods and Software Technology*, volume 936 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 1995.

[Jac96a] Bart Jacobs. Behaviour refinement of coalgebraic specifications with coinductive correctness proofs. Technical Report CSI-R9618, University of Nijmegen, 1996.

[Jac96b] Bart Jacobs. Inheritance and cofree constructions. In P. Cointe, editor, *European Conference on Object-Oriented Programming*, volume 1098 of *Lecture Notes in Computer Science*, pages 210–231. Springer, 1996.

[Jac96c] Bart Jacobs. Objects and classes, coalgebraically. In B. Freitag, C.B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object Orientation with Parallelism and Persistence*, pages 83–103. Kluwer Academic Publishers, 1996.

[Jac97] Bart Jacobs. Invariants, bisimulations and the correctness of coalgebraic refinements. In M. Johnson, editor, *Algebraic Methodology and Software Technology*, volume 1349 of *Lecture Notes in Computer Science*, pages 276–291. Springer, 1997.

[Jac98] Bart Jacobs. Coalgebraic reasoning about classes in object-oriented languages. In H. Reichel, editor, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 235–246. Elsevier Science, 1998.

[Jac00] Bart Jacobs. Towards a duality result in coalgebraic modal logic. In H. Reichel, editor, *Coalgebraic Methods in Computer Science*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 163–198. Elsevier Science, 2000.

[JR97] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1997.

[KH] Alexander Kurz and Rolf Hennicker. On institutions for modular coalgebraic specifications. To appear in Theoretical Computer Science.

[Kur98] Alexander Kurz. Specifying coalgebras with modal logic. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 57–71. Elsevier Science, 1998.

[Kur00] Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, Ludwig-Maximilians-Universität München, 2000.

[Lan71] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

[Mal96] Grant Malcolm. Behavioural equivalence, bisimilarity and minimal realisation. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications*, volume 1130 of *Lecture Notes in Computer Science*, pages 359–378. Springer, 1996.

[Man76] Ernest G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer, 1976.

[Mes89] José Meseguer. General logics. In H.-D. Ebbinghaus et al., editor, *Logic Colloquium'87*, pages 275–329. North-Holland, 1989.

[MG94] Grant Malcolm and Joseph Goguen. Proving correctness of refinement and implementation. Technical Monograph PRG-114, Oxford University, 1994.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mos99] Lawrence S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999.

[MT92] Karl Meinke and John V. Tucker. Universal algebra. In S. Abramski, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 189–411. Oxford University Press, 1992.

[NO88] Pilar Nivela and Fernando Orejas. Initial behavioural semantics for algebraic specifications. In *5th Workshop on Algebraic Specifications of Abstract Data Types*, volume 332 of *Lecture Notes in Computer Science*, pages 184–207. Springer, 1988.

[ONE89] Fernando Orejas, Pilar Nivela, and Hartmut Ehrig. Semantical constructions for categories of behavioural specifications. In H. Ehrig, H. Herrlich, and H.J. Kreowski, editors, *Computer Science - with Aspects from Topology*, volume 393 of *Lecture Notes in Computer Science*, pages 220–243. Springer, 1989.

[Pad96] Peter Padawitz. Swinging data types: syntax, semantics and theory. In O.-J. Dahl, M. Haveraaen, and O. Owe, editors, *Recent Trends in Data Type Specification*, volume 1130 of *Lecture Notes in Computer Science*, pages 409–435. Springer, 1996.

[Poi92] Axel Poigné. Basic category theory. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 413–640. Oxford University Press, 1992.

[Rei81] Horst Reichel. Behavioural equivalence – A unifying concept for initial and final specification methods. In M. Arata and L. Varga, editors, *Proceedings, Third Hungarian Computer Science Conference*. Akademiai Kiado, 1981.

[Rei95] Horst Reichel. An approach to object semantics based on terminal coalgebras. *Mathematical Structures in Computer Science*, 5:129–152, 1995.

[Rei98] Horst Reichel. Dialgebraic logics. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 247–255. Elsevier Science, 1998.

[RG00] Grigore Roșu and Joseph Goguen. Hidden congruent deduction. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2000.

[Roș98] Grigore Roșu. A Birkhoff-like axiomatizability result for hidden algebra and coalgebra. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Coalgebraic Methods in Computer Science*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 179–196. Elsevier Science, 1998.

[Röß98] Martin Rößiger. From modal logic to terminal coalgebras. Technical Report MATH-AL-3-1998, Technical University Dresden, 1998.

[Röß00] Martin Rößiger. Coalgebras and modal logic. In H. Reichel, editor, *Coalgebraic Methods in Computer Science*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 299–320. Elsevier Science, 2000.

[RT94] Jan Rutten and Daniele Turi. Initial algebra and final coalgebra semantics for concurrency. Technical Report CS-R9409, CWI, 1994.

[Rut95] Jan Rutten. A calculus of transition systems (towards universal coalgebra). Technical Report CS-R9503, CWI, 1995.

[Rut96] Jan Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI, 1996.

[Sch72] Horst Schubert. *Categories*. Springer, 1972.

[SP82] Michael B. Smyth and Gordon D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 4(11):761–783, 1982.

[TP97] Daniele Turi and Gordon Plotkin. Towards a mathematical operational semantics. In *Proceedings, LICS*, pages 280–291, 1997.

[Tur96] Daniele Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Free University Amsterdam, 1996.

[TWW82] James Thatcher, Eric Wagner, and Jesse Wright. Data type specification: Parameterization and the power of specification techniques. *ACM Transactions on Programming Languages and Systems*, 4(4):711–732, 1982.

Index