From algebras and Coalgebras to Dialgebra

E. Poll, J. Zwanenburg

Computing Science Institute/

CSI-R0101 January 2001
From Algebras and Coalgebras to Dialgebras*

Erik Poll and Jan Zwanenburg
University of Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
http://www.cs.kun.nl/~{erikpoll,janz}
{erikpoll,janz}@cs.kun.nl

Abstract
This paper investigates the notion of dialgebra, which generalises the notions of algebra and coalgebra. We show that many (co)algebraic notions and results can be generalised to dialgebras, and investigate the essential differences between (co)algebras and arbitrary dialgebras.

1 Introduction

An algebra is a set $X$ together with some functions that can be used to construct elements of $X$, i.e. functions $f_i$ that have $X$ as output type,

$$f_i : IN_i(X) \to X$$

with $IN_i$ a polynomial functor. Algebras are widely used in (theoretical) computer science. E.g. think of algebraic datatypes such as lists and trees, or algebraic specifications.

A coalgebra is a set $X$ together with some functions that can be used to observe elements of $X$, i.e. functions $f_i$ that have $X$ as input type,

$$f_i : X \to OUT_i(X)$$

with $OUT_i$ a polynomial functor. Coalgebras can be used to describe various kinds of ‘dynamical systems’, e.g. automata, processes, or (labelled) transition systems [Rut00]. Moreover, elements of coalgebras can viewed as objects in the sense of object-oriented (OO) programming [Rei95], in which case the operations are viewed as methods.

For an introduction to – and a comparison between – algebras and coalgebras we refer to [JR97]. Algebras and coalgebras are dual notions, and are in some intuitive sense ‘opposites’. However, this does not mean that algebra and coalgebra do not have certain things in common. Indeed, the standard example of an algebraic specification, stacks, also occurs in the literature as an example of a coalgebraic specification!

This paper investigates the notion of dialgebra, a straightforward generalisation of (co)algebra: a dialgebra is a set $X$ together with some functions

$$f_i : IN_i(X) \to OUT_i(X)$$

with $IN_i$ and $OUT_i$ polynomial functors. The name ‘dialgebra’ is taken from [Hag87]. Clearly all algebras and coalgebras are dialgebras.

*A shorter version of this paper appears in Coalgebraic Methods in Computer Science (CMCS’2001)
An example of a dialgebra that is neither an algebra nor a coalgebra is a type \( \text{Set} \) of sets of natural numbers with operations

- \( \text{empty} : \text{Set} \)
- \( \text{add} : \text{Set} \times \text{Nat} \rightarrow \text{Set} \)
- \( \text{elem} : \text{Set} \times \text{Nat} \rightarrow \text{bool} \)
- \( \text{union} : \text{Set} \times \text{Set} \rightarrow \text{Set} \)
- \( \text{min} : \text{Set} \rightarrow 1 + \text{Nat} \)
- \( \text{split} : \text{Set} \times \text{Nat} \rightarrow \text{Set} \times \text{Set} \)

Here \( \text{min} \) could for instance return the minimum of a set, if the set is non-empty, and an element of a unit type \( 1 \) if the set is empty, and \( \text{split}(s,n) \) could split the set \( s \) in a pair of sets, one containing the elements of \( s \) smaller than \( n \) and the other containing the rest. This is not an algebra, because for instance \( \text{split} \) and \( \text{min} \) are not ‘algebraic’, nor is it a coalgebra, because for instance \( \text{empty} \) and the binary operation \( \text{union} \) are not ‘coalgebraic’.

Many interesting examples of dialgebras that are not (co)algebras can be obtained simply by extending a coalgebra with an operation \( \text{init} : 1 \rightarrow X \) that yields some initial state. Including such an operation is a very natural thing to do for many examples of coalgebras. In particular, this is very natural when considering objects in the sense of OO: here using dialgebras rather than coalgebras makes it possible to account for the constructors as well as the methods of a class. In addition it becomes possible to account for so-called binary methods.

We will show that most (co)algebraic notions can be defined for the more general dialgebraic case, and investigate in how far properties of (co)algebras – in particular properties of invariants and bisimulations – can be generalised to arbitrary dialgebras, to get a better understanding of what the essential differences between algebras, coalgebras, and dialgebras are.

### 2 Mathematical Preliminaries

Throughout this paper, types will just be sets. Signatures are mappings from types to types, written as type expressions containing a type variable \( X \).

**Definition 2.1** A signature \( \Sigma(X) \) is a type expression, possibly containing the type variable \( X \), of the form

\[
\Sigma(X) ::= X \mid C \mid \Sigma_1(X) + \Sigma_2(X) \mid \Sigma_1(X) \times \Sigma_2(X) \mid \Sigma_1(X) \rightarrow \Sigma_2(X),
\]

where \( C \) comes from a collection of base types. Here \( A \times B \) is the Cartesian product, with projections \( \pi_1 : A \times B \rightarrow A \) and \( \pi_2 : A \times B \rightarrow B \), and \( A + B \) the disjoint sum, with injections \( \text{inl} : A \rightarrow A + B \) and \( \text{inr} : B \rightarrow A + B \). The functions \( [f_1, f_2] : A_1 + A_2 \rightarrow B \) for \( f_i : A_i \rightarrow B \) and \( (g_1, g_2) : A \rightarrow B_1 \times B_2 \) for \( g_i : A \rightarrow B_i \) are defined as usual.

A **polynomial signature** is a signature of the form

\[
F(X) ::= X \mid C \mid F_1(X) + F_2(X) \mid F_1(X) \times F_2(X).
\]

**N.B.** We deliberately exclude constant exponents of the form \( C \rightarrow F(X) \) as polynomial functors, because we have not been able to prove our most interesting result, Theorem 3.20, if we include these.

Polynomial signatures are functors, and \( F(f) : F(A) \rightarrow F(B) \) is defined in the usual way for \( f : A \rightarrow B \). The notions of predicate and relation lifting can be defined not just for polynomial signature but for all signatures: we can define predicate and relation lifting for arbitrary signatures.

**Definition 2.2** For predicates \( P \) and \( Q \) we define the predicates

- \( (P \times \text{pred} Q)(x) \iff P(\pi_1(x)) \land Q(\pi_2(x)) \)
\begin{itemize}
  \item \((P \overset{\text{pred}}{\rightarrow} Q)(x) \iff (x = \text{inl}(x') \land P(x')) \lor (x = \text{inr}(x') \land Q(x'))\)
  \item \((P \overset{\text{pred}}{\rightarrow} Q)(f) \iff \forall x \in A. \ P(x) \Rightarrow Q(f(x)), \text{ with } A \text{ the domain of } P.\)
\end{itemize}

**Definition 2.3** For relations \(R\) and \(S\) we define the relations
\begin{itemize}
  \item \(R +\text{rel}\ S = \{(\text{inl}(x), \text{inl}(y)) \mid (x,y) \in R\} \cup \{(\text{inr}(x), \text{inr}(y)) \mid (x,y) \in S\}\)
  \item \(R \times\text{rel}\ S = \{(x,y) \mid (\pi_1(x), \pi_1(y)) \in R \land (\pi_2(x), \pi_2(y)) \in S\}\)
  \item \(R \overset{\text{rel}}{\rightarrow} S = \{(f,g) \mid \forall(x,y) \in R. \ (f(x),g(y)) \in S\}\)
\end{itemize}

**Definition 2.4** Let \(\Sigma(X)\) be an arbitrary signature. For a predicate \(P\) on \(X\) and a relation \(\sim \subseteq X \times Y\), the predicate \(\Sigma^{\text{pred}}(P)\) on \(\Sigma(X)\) and the relation \(\Sigma^{\text{rel}}(\sim) \subseteq \Sigma(X) \times \Sigma(Y)\) are defined, by induction on the structure of \(\Sigma(X)\), as
\begin{itemize}
  \item if \(\Sigma(X) = X\) then \(\Sigma^{\text{pred}}(P) = P\) and \(\Sigma^{\text{rel}}(\sim) = \sim\),
  \item if \(\Sigma(X) = C\) then \(\Sigma^{\text{pred}}(P) = \text{True}_C\), the constant predicate ‘true’ on \(C\), and \(\Sigma^{\text{rel}}(\sim) = Id_C\), the identity relation on \(C\),
  \item if \(\Sigma(X) = \Sigma_1(X) + \Sigma_2(X)\) then \(\Sigma^{\text{pred}}(P) = \Sigma_1^{\text{pred}}(P) +\text{pred} \Sigma_2^{\text{pred}}(P)\) and \(\Sigma^{\text{rel}}(\sim) = \Sigma_1^{\text{rel}}(\sim) +\text{rel} \Sigma_2^{\text{rel}}(\sim)\),
  \item if \(\Sigma(X) = \Sigma_1(X) \times \Sigma_2(X)\) then \(\Sigma^{\text{pred}}(P) = \Sigma_1^{\text{pred}}(P) \times\text{pred} \Sigma_2^{\text{pred}}(P)\) and \(\Sigma^{\text{rel}}(\sim) = \Sigma_1^{\text{rel}}(\sim) \times\text{rel} \Sigma_2^{\text{rel}}(\sim)\),
  \item if \(\Sigma(X) = \Sigma_1(X) \rightarrow \Sigma_2(X)\) then \(\Sigma^{\text{pred}}(P) = \Sigma_1^{\text{pred}}(P) \rightarrow\text{pred} \Sigma_2^{\text{pred}}(P)\) and \(\Sigma^{\text{rel}}(\sim) = \Sigma_1^{\text{rel}}(\sim) \rightarrow\text{rel} \Sigma_2^{\text{rel}}(\sim)\).
\end{itemize}

We will sometimes omit the superscripts \(\text{rel}\) and \(\text{pred}\) if they are obvious from the context. Of course, if we identify predicates with subsets, then there is no difference between \(\Sigma\) and \(\Sigma^{\text{pred}}\).

The notion of relation lifting is not just used in the coalgebraic literature (e.g. [Jac97]), but is much more widely used, notably for logical relations in the semantics of typed lambda calculus (see [Mit96] for a comprehensive overview) and to formalise the notion of parametricity (e.g. see [PA93]).

**Lemma 2.5** Let \(\Sigma(X)\) be an arbitrary signature. Then
\begin{enumerate}
  \item \(\Sigma^{\text{pred}}(\text{True}_X) = \text{True}_{\Sigma(X)}\)
  \item \(\Sigma^{\text{rel}}(\text{Id}_X) = \text{Id}_{\Sigma(X)}\)
  \item \(F^{\text{rel}}(R^{op}) = F^{\text{rel}}(R)^{op}\)
\end{enumerate}

**Proof** Induction on the structure of \(\Sigma(X)\). \(\square\)

For polynomial signatures, there is a close connection between relation and function lifting:

**Lemma 2.6** For any polynomial signature \(F(X)\)
\[\text{graph}(F(f)) = F^{\text{rel}}(\text{graph}(f))\]
where \(\text{graph}(f) \subseteq A \times B\) is the function \(f : A \rightarrow B\) viewed as a relation.

**Proof** Induction on the structure of \(F(X)\). \(\square\)

**Lemma 2.7** Let \(F(X)\) be a polynomial signature, and let \(i\) range over \(I\), \(I\) not empty. Then
\begin{enumerate}
  \item \(P \subseteq Q \Rightarrow F^{\text{pred}}(P) \subseteq F^{\text{pred}}(Q)\)
\end{enumerate}
(ii) \( R \subseteq S \Rightarrow F^{rel}(R) \subseteq F^{rel}(S) \)

(iii) \( \bigcap_i F^{presh}(P_i) = F^{presh}(\bigcap_i P_i) \)

(iv) \( \bigcap_i F^{rel}(R_i) = F^{rel}(\bigcap_i R_i) \)

(v) \( \bigcup_i F^{presh}(P_i) \subseteq F^{presh}(\bigcup_i P_i) \)

(vi) \( \bigcup_i F^{rel}(R_i) \subseteq F^{rel}(\bigcup_i R_i) \)

(vii) \( F^{rel}(R;S) = F^{rel}(R);F^{rel}(S) \)

**Proof** All these can be proved by induction on the structure of \( F(X) \). Properties 3. and 4. are also easy consequences of 1. and 2., respectively; see Corollary A.2 in the appendix. \( \square \)

None of the properties in Lemma 2.7 hold for arbitrary signatures. Note that we have stronger properties for intersection, (iii) and (iv), than for union, (v) and (vi). The properties \( \bigcup_i F^{presh}(P_i) = F^{presh}(\bigcup_i P_i) \) and \( \bigcup_i F^{rel}(R_i) = F^{rel}(\bigcup_i R_i) \) do not hold for all polynomial signatures \( F(X) \) (for counterexamples, take \( F(X) = X \times X \)), but do hold for some polynomial signatures:

**Lemma 2.8** Let \( F(X) \) be a polynomial signature with at most one occurrence of \( X \). Then

(i) \( \bigcup_i F^{presh}(P_i) = F^{presh}(\bigcup_i P_i) \)

(ii) \( \bigcup_i F^{rel}(R_i) = F^{rel}(\bigcup_i R_i) \)

**Proof** Induction on the structure of \( F(X) \). Of course, for \( F(X) \) with no occurrence of \( X \) – i.e. \( F(X) \) a constant – the property is trivial. \( \square \)

More properties of predicate and relation lifting, and additional counterexamples, are given in the appendix.

### 3 Dialgebras

**Definition 3.1** A **dialgebraic signature** is a signature of the form

\[ \Sigma(X) = \Sigma_1(X) \times \ldots \times \Sigma_n(X), \]

with each \( \Sigma_i(X) \) of the form

\[ \Sigma_i(X) = IN_i(X) \rightarrow OUT_i(X), \]

with \( IN_i(X) \) and \( OUT_i(X) \) polynomial signatures. \( \Sigma(X) \) is called **algebraic** iff \( OUT_i(X) = X \) for all \( i \), and **coalgebraic** iff \( IN_i(X) = X \) for all \( i \).

Throughout the remainder of this paper, \( \Sigma \) will be a dialgebraic signature of the form

\[ \Sigma(X) = \prod_{i \in I} \Sigma_i = \prod_{i \in I} IN_i(X) \rightarrow OUT_i(X). \]

In examples \( \Sigma(X) \) will usually be a labelled product rather than unlabelled one; this is just syntactic sugar.

**Definition 3.2** A **\( \Sigma \)-dialgebra** is a pair \((A,f)\) consisting of a set \( A \) and a function \( f \in \Sigma(A) \), i.e. \( f = (f_1, \ldots, f_n) \) with \( f_i \in \Sigma_i(A) = IN_i(A) \rightarrow OUT_i(A) \).
For (co)algebraic signatures, this is just the definition of \( \Sigma \)-(co)algebra. An important difference between dialgebras and (co)algebras is that whereas an algebra with \( n \) operations \( f_i : F_i(X) \to X \) can be turned into a algebra with the single operation, namely \([f_1, \ldots, f_n] : F_1(X) + \ldots + F_n(X) \to X\), and, similarly, a coalgebra with \( n \) operations \( f_i : X \to F_i(X) \) can be turned into a coalgebra with just one operation \( (f_1, \ldots, f_n) \), we can not do something similar for dialgebras. A practical consequence is that most definitions and proofs for dialgebras have to be ‘point-wise’, quantifying over \( i \).

**Definition 3.3** Let \((A, f)\) and \((B, g)\) be \( \Sigma \)-dialgebras. A **\( \Sigma \)-homomorphism** \( h : (A, f) \to (B, g) \) is a function from \( A \) to \( B \) that preserves the operations, i.e.

\[
OUT_i(h) \circ f_i = g_i \circ IN_i(h)
\]

for all \( i \).

For dialgebras that are (co)algebras, this is the standard notion of (co)algebraic homomorphism.

**Lemma 3.4**

(i) The identity \( id_A \) is a homomorphism from \((A, f)\) to itself.

(ii) Homomorphisms are closed under composition, i.e. if \( h : (A, f) \to (A', f')\) and \( h' : (A', f') \to (A'', f'')\) then \( h' \circ h : (A, f) \to (A'', f'')\).

**Proof** Easy.

**Lemma 3.5** Let \( \Sigma(X) \) be a dialgebraic signature, and let \( i \) range over \( I \), \( I \) not empty. Then

(i) \( \bigcap_i \Sigma^{pred}(P_i) \subseteq \Sigma^{pred}(\bigcap_i P_i) \)

(ii) \( \bigcap_i \Sigma^{rel}(R_i) \subseteq \Sigma^{rel}(\bigcap_i R_i) \)

(iii) \( \Sigma^{rel}(R) ; \Sigma^{rel}(S) \subseteq \Sigma^{rel}(R; S) \)

**Proof** The proofs are quite straightforward, using Lemma 2.7. We just give the proof of (i) for binary intersection; the others are similar, as shown in Lemma A.7. in the appendix.

\[
f \in \Sigma(P \cap Q) \\
\iff \forall j. f_j \in \Sigma_j(P \cap Q) \\
\iff \forall j. f_j \in \Sigma_j(P \cap Q) \to OUT_j(P \cap Q) \\
\iff \forall j. \forall x \in \Sigma_j(P \cap Q). f_j(x) \in OUT_j(P \cap Q) \\
\iff \forall j. \forall x \in \Sigma_j(P \cap Q). f_j(x) \in OUT_j(P) \cap OUT_j(Q) \\
\quad \text{by Lemma 2.7(iii) (twice)} \\
\iff \forall j. (\forall x \in \Sigma_j(P). f_j(x) \in OUT_j(P)) \\
\quad \land (\forall x \in \Sigma_j(Q). f_j(x) \in OUT_j(Q)) \\
\iff \forall j. f_j \in \Sigma_j(P) \to OUT_j(P) \land f_j \in \Sigma_j(Q) \to OUT_j(Q) \\
\iff \forall j. f_j \in \Sigma_j(P) \cap \Sigma_j(Q) \\
\iff f \in \Sigma(P) \cap \Sigma(Q)
\]
3.1 Invariants and sub-dialgebras

The notion of invariant is used in the literature both for algebras and for coalgebras. Intuitively, a predicate is an invariant if all the operations preserve it:

**Definition 3.6** A predicate $P$ on $A$ is an invariant for a $\Sigma$-dialgebra $(A, f)$ iff $f \in \Sigma^{bred}(P)$.

For dialgebras that are (co)algebras, this is the standard notion of (co)algebraic invariant. Given an invariant we can construct a sub-dialgebra:

**Definition 3.7** A sub-dialgebra of a $\Sigma$-dialgebra $(A, f)$ is another $\Sigma$-dialgebra $(A', f)$ with $A' \subseteq A$.

For dialgebras that are (co)algebras, this is the standard notion of sub-(co)algebra. For $(A', f)$ to be a sub-dialgebra of $(A, f)$ all the functions in $f$ have to be ‘closed’ under the subset $A'$ of $A$.

**Lemma 3.8** Invariants are closed under intersection.

**Proof** Follows immediately from Lemma 3.5: if $f \in \Sigma(P)$ and $f \in \Sigma(Q)$ then $f \in \Sigma(P \cap Q)$ by Lemma 3.5. □

As a consequence of this lemma, we can define the smallest invariant of a dialgebra as the intersection of all invariants. This strongest invariant expresses exactly the property of being ‘reachable’ by the dialgebra operations. Beware of the subtle difference between ‘$P$ holds for all elements of $A$’, ‘$P$ is an invariant’, and ‘$P$ holds for the reachable elements of $A$’. These notions are not equivalent, but related as follows:

- $P$ holds for all elements of $A$ $\Rightarrow$ $P$ is an invariant
- $\Rightarrow$ $P$ holds for all reachable elements of $A$.

Invariants of dialgebras are not always closed under union:

**Counterexample 3.9** Let $T(X) = X \times X \rightarrow X$ and consider the $T$-(di)algebra $(\mathbb{Z}, +)$. The predicates $Neg(x) = x < 0$ and $Pos(x) = x > 0$ on $\mathbb{Z}$ are both invariants, but clearly their union is not, because the sum of a positive and a negative number may be 0. □

For coalgebras, however, we do have this property (e.g. see [Rut00, JR97]):

**Lemma 3.10** For coalgebras, invariants are closed under union.

A useful consequence of this lemma is that, for a coalgebra, given any property $\Phi$ there exist a largest invariant $\Phi \subseteq \Phi$, namely the union of all invariants that are subsets of $\Phi$. We can slightly generalise Lemma 3.10. For this we first define

**Definition 3.11** $\Sigma(X)$ has no binary methods if none of the $IN_i(X)$ has more than one occurrence of $X$.

Clearly all coalgebras are dialgebras without binary methods. Many interesting examples of dialgebras without binary methods that are not coalgebras can be obtained in the way mentioned earlier, simply by extending a coalgebra with an operation that yields some initial state.

Note that Counterexample 3.9 involves a binary method. Binary methods are already notorious in the theoretical computer science literature on object oriented (OO) programming; see [BCC+96]. Some properties of coalgebras, that do not hold for all dialgebras, do hold for all dialgebras without binary methods, including:

**Lemma 3.12** For dialgebras without binary methods, invariants are closed under union.
Proof Let $\Sigma$ be a signature without binary methods. It suffices to prove that $\bigcup_j \Sigma^{pre}(P_j) \subseteq \Sigma^{pre}(\bigcup_j P_j)$. The crucial property of dialgebraic signature without binary methods needed to prove this is that, since there is at most one occurrence of $X$ in the $IN_i(X)$, $\bigcup_i F^{pre}(IN_i) = F^{pre}(\bigcup_i\{IN_i\})$ by Lemma 2.8:

$$f \in \Sigma(P) \cup \Sigma(Q)$$

$$\iff \forall j. f_j \in \Sigma_j(P) \cup \Sigma_j(Q)$$

$$\iff \forall j. f_j \in IN_j(P) \to OUT_j(P) \lor f_j \in IN_j(Q) \to OUT_j(Q)$$

$$\iff \forall j. (\forall x \in IN_j(P). f_j(x) \in OUT_j(P)) \lor (\forall x \in IN_j(Q). f_j(x) \in OUT_j(Q))$$

$$\iff \forall j. (\forall x \in IN_j(P \cup Q). f_j(x) \in OUT_j(P \cup Q))$$

$$\iff \forall j. f_j \in IN_j(P \cup Q) \to OUT_j(P \cup Q)$$

$$\iff f \in \Sigma(P \cup Q)$$

3.2 Bisimulations, (partial) congruences, and quotient-dialgebras

The notion of bisimulation plays an important role in the literature on coalgebras, as does the closely related notion of (partial) congruence in the literature on algebras.

Definition 3.13 A relation $\sim \subseteq A \times B$ is a bisimulation between two $\Sigma$-dialgebras $(A, f)$ and $(B, g)$ iff $(f, g) \in \Sigma^{pre}(\sim)$. Dialgebras $(A, f)$ and $(B, g)$ are bisimilar if there exists a bisimulation between $(A, f)$ and $(B, g)$.

For (co)algebras one has the property that (co)algebra homomorphisms are just functional bisimulations. This property also holds for dialgebras:

Lemma 3.14 Let $(A, f)$ and $(B, g)$ be $\Sigma$-dialgebras and $h$ be a function from $A$ to $B$. Then $h : A \to B$ is a homomorphism iff $\text{graph}(h) \subseteq A \times B$ is a bisimulation.

Proof Straightforward, using Lemma 2.6.

Lemma 3.15 Bisimulations are closed under intersection and composition.

Proof Follows immediately from Lemma 3.5.

Bisimulations between dialgebras are not always closed under union. For coalgebras this is a basic property (e.g. see [Rut00, JR97]):

Lemma 3.16 For coalgebras, bisimulations are closed under union.

An important consequence of this property of coalgebras is that between any two coalgebras there exists a largest bisimulation, namely the union of all bisimulations. For arbitrary dialgebras we do not have this property.

Lemma 3.17 For dialgebras without binary methods, bisimulations are closed under union.
Proof Similar to Lemma 3.12.

Of special interest are bisimulations between a dialgebra and itself:

**Definition 3.18** A relation \(~ \subseteq A \times A\) is a (partial) congruence on a dialgebra \((A, f)\) iff it is a bisimulation between \((A, f)\) and itself, i.e. \((f, f) \in \Sigma(\sim)\) and it is a (partial) equivalence relation.

In [Rut00], for coalgebras, what we call a congruence here is called a bisimulation equivalence. Given a (partial) congruence we can construct a quotient-dialgebra:

**Definition 3.19** Let \(~ \subseteq (A, f)\). Then the quotient-dialgebra \([\Sigma(\sim), [f]_\sim]\) is the \([\Sigma(\sim), [f]_\sim]\), where \([\Sigma(\sim)]\) is the collection of \(~\)-equivalence classes, and \([f]_\sim\) is the family of functions on \(~\)-equivalence classes induced by \(f\).

As mentioned above, bisimulations are not closed under union. Congruences, however, are, in the following sense:

**Theorem 3.20** Let \(R_j\) be congruences on the \(\Sigma\)-dialgebra \((A, f)\), for \(j \in J, J \neq \emptyset\). Then \((\bigcup_j R_j)^*\) is also a congruence relation for \((A, f)\).

**Proof** See the appendix, Theorem A.13

So, for any dialgebra there exists a largest congruence relation, namely (the transitive closure of) the union of all congruences. Intuitively, this is the notion of observational equality for that dialgebra. In Section 3.3 below, we discuss this difference between dialgebras and coalgebras – the existence of largest congruences vs largest bisimulations – in more detail.

The property above does not hold for partial congruences:

**Counterexample 3.21** Let \(T(X) = X \times X \rightarrow \mathbb{Z}\) and consider the \(T\)-(di)algebra \((\mathbb{Z}, f)\) where \(f\) is the function 
\[
f(x, y) = \begin{cases} 
0 & \text{if } y < 0 \\
23 & \text{else}
\end{cases}
\]
The relations \(R = \mathbb{N} \times \mathbb{N}\) and \(Id_{\mathbb{Z}}\) on \(\mathbb{Z}\) are both bisimulations, i.e.
\[
(x, x') \in R \land (y, y') \in R \implies f(x, y) = f(x', y')
\]
\[
(x, x') \in Id \land (y, y') \in Id \implies f(x, y) = f(x', y')
\]
However, neither \(R \cup Id\) nor \((R \cup Id)^*\) is a bisimulation, since for instance \((4, 5) \in R \cup Id\) and \((-1, -1) \in R \cup Id\), but \(f(4, -1) \neq f(5, -1)\).

### 3.3 Coalgebras vs dialgebras: the problem with binary methods

Moving from coalgebras to dialgebras we gain something, notably the possibility of having binary methods. The price for this is that some properties are lost, namely

- the existence of final coalgebras,
- the existence of unique largest bisimulation between any two coalgebras.

These two properties are intimately connected, as the largest bisimulation relates precisely those elements that have the same image under the unique homomorphisms to the final coalgebra. The properties are useful because they provide a canonical notion of observational equality between elements of different coalgebras. Two different coalgebras \((A, f)\) and \((B, g)\) with the same signature \(\Sigma\) can be regarded as different implementations for classes with the same interface. The properties

\footnote{The property does hold for partial congruences if these all have the same domain; instead of the transitive and reflexive closure \((\bigcup_j R_j)^*\) one then has to consider the transitive closure \((\bigcup_j R_j)^+\).}
above then provide a canonical notion of equality between objects from these two classes: an
object with an internal state \( a \in A \) – using implementation \((A, f)\) – is observationally equal to an
object with an internal state \( b \in B \) – using implementation \((B, g)\) – iff \( a \simeq b \), where \( a \simeq b \) is the
greatest bisimulation between \((A, f)\) and \((B, g)\).

Below a simple (counter)example to illustrate the fundamental problem with the union of
bisimulations caused by a binary method:

**Counterexample 3.22** Consider \( \Sigma(X) = (X \times X \to X) \times (X \to \mathbb{N}) \) and the \( \Sigma \)-dialgebras
\( Z = (\mathbb{Z}, (+, \text{abs})) \) and \( L = (\text{List}, (++., \text{length})) \) with List the set of lists over some type, \( ++ \) the
concatenation operation, and \( \text{abs} : \mathbb{Z} \to \mathbb{N} \) the function returning the absolute value.

The relations \( \sim_1 \) and \( \sim_2 \),
\[
\sim_1 = \{ (z, l) \mid z = \text{length}(l) \} \\
\sim_2 = \{ (z, l) \mid z = -\text{length}(l) \}
\]
are bisimulations between \( Z \) and \( L \). However, \( \sim_1 \cup \sim_2 \) is not a bisimulation between \( Z \) and \( L \); for example, if \( l \) is some list with three elements, then
\[
-3(\sim_1 \cup \sim_2)l \wedge 3(\sim_1 \cup \sim_2)l \neq -3 + 3 (\sim_1 \cup \sim_2)l + l .
\]
The problem is that when the binary method \((+ \text{ or } ++)\) is used to observe an individual element
(of \( \mathbb{Z} \) or List, respectively), the operation \(+\) of \( Z \) offers different – more – observations than the
operation \( ++ \) of \( L \).  \( \square \)

Binary methods are notorious in the literature on the theoretical foundations of object-oriented
programming: in the presence of binary methods defining a satisfactory notion of subtyping be­
comes a problem [BCC+96]. This problem is closely related to the fact that there is no canonical
notion of ‘observational equivalence’ in the presence of binary methods. Indeed, the coalgebraic
definition of subtyping (see [Pol00]) crucially depends on the existence of final coalgebras, or the
existence of unique largest bisimulation between any two coalgebras.

4 Dialgebraic Specification

Just like algebras provide the basis for algebraic specification, coalgebras have been used as
the basis for coalgebraic specification, notably in the experimental specification language CCSL
[HHJT98, RJT01]. We now consider a notion of dialgebraic specification, defined in exactly the
same way. Because of lack of space, we have to omit many details of definitions and proofs.

For example, an equational dialgebraic specification for the dialgebraic signature of \( \text{Set} \) men­
tioned in the introduction could include

\[
\begin{align*}
\text{union}(s, \text{empty}) & = s \\
\text{union}(s, t) & = \text{union}(t, s) \\
\text{elem}(\text{add}(s, n), m) & = (n = m \text{ or } \text{elem}(s, m)) \\
\text{elem}(\text{union}(s, t), n) & = \text{elem}(s, n) \text{ or } \text{elem}(t, n) \\
\text{elem}(\text{empty}, n) & = \text{false} \\
\text{min}(s) & = \text{inl}(x) \iff s = \text{empty} \\
\text{min}(s) & = \text{inr}(m) \Rightarrow \text{elem}(s, m) = \text{true} \\
\text{min}(s) & = \text{inr}(m) \wedge \text{elem}(s, n) = \text{true} \Rightarrow m \leq n \\
\text{union}(\text{split}(s, m)) & = s \\
\end{align*}
\]

A model of this dialgebraic specification would be a dialgebra providing an implementation of all
the operations for which these equations hold. However, instead of insisting that models satisfy
the equations above, one could also only require that they satisfy the equations above up to some
congruence relation. This weaker notion of model makes sense because intuitively a congruence
relation provides a notion of ‘observational equivalence’.
For example, consider a simple implementation of the specification above using lists, in which union is implemented as concatenation ++, i.e. some dialgebra \(L = (\text{List}_{\text{Nat}}, \ldots, +, \ldots)\). This implementation does not satisfy

\[
\text{union}(s, t) = \text{union}(t, s)
\]
as concatenation is not commutative, but it does satisfy

\[
\text{union}(s, t) \sim_{\text{perm}} \text{union}(t, s)
\]
where \(\sim_{\text{perm}}\) is the relation on lists that relates permutations. If \(\sim_{\text{perm}}\) is a congruence for \(L\), then \(L\) can be regarded as a correct implementation of the specification.

To formally introduce a notion of equational specification for dialgebras, we need to introduce some syntax. We fix a dialgebraic signature \(\Sigma(X) = \prod_{i \in I} \text{IN}_i(X) \to \text{OUT}_i(X)\) and use names \(o_{pi}\) for the operations of the dialgebra:

**Definition 4.1** The type expressions are given by

\[
E ::= X \mid C \mid E_1 + E_2 \mid E_1 \times E_2 \mid E_1 \to E_2
\]

where \(X\) stands for the carrier of the dialgebra, and \(C\) ranges over some collection of base types. The term expressions is given by

\[
e ::= o_{pi} \mid x^E \mid c^E \mid \lambda x^E.e \mid e(e) \mid \text{inl}_{E+E}(e) \mid \text{inr}_{E+E}(e) \mid \pi_1(e) \mid \pi_2(e)
\]

where \(x^E\) is a variable of type \(E\), and \(c^E\) a constant of type \(E\) not containing \(X\). We assume an infinite number of variables for every type \(E\). The collection of well-typed terms is defined in the obvious way.

The propositions are given by

\[
\Phi ::= \neg \Phi \mid \Phi_1 \land \Phi_2 \mid e_1 =_{E} e_2 \mid \forall x^E. \Phi
\]

where \(E\) is some type expression possibly containing \(X\), and \(e_1\) and \(e_2\) are well-formed expressions of type \(E\).

A dialgebraic specification \(\Phi\) is simply a closed proposition.

Note that the definition above allows more propositions than usually allowed in algebraic specification; for example, it allows universal quantifications over the type \(X \to X\).

**Definition 4.2** The interpretation of a type \(E\), term \(e\), and proposition \(\Phi\) in the dialgebra \((A, f)\), \([E]_{A,f}\), \([e]_{A,f}\), and \([\Phi]_{A,f}\) are defined in the obvious way, by induction on the structure, interpreting \(X\) as \(A\) and \(o_{pi} : \Sigma_i(X)\) as \(f_i : \Sigma_i(A)\).

Because terms and propositions can contain free variables, we have to define the interpretation of terms and propositions wrt. an environment \(\eta\), \([e]^{\eta}_{A,f}\) and \([\Phi]^{\eta}_{A,f}\), where the environment \(\eta\) assigns to every free variable \(x^E\) an interpretation in \(E(A)\).

**Definition 4.3** A dialgebra \((A, f)\) satisfies specification \(\Phi\), written \((A, f) \models \Phi\), iff \([\Phi]_{A,f}\) is true.

Now that we have a syntax, we can define a notion of observational equivalence:

**Definition 4.4** Two \(\Sigma\)-dialgebras \((A, f)\) and \((B, g)\) are observationally equivalent iff for all closed expressions \(e\) of some closed type \(E\) (i.e. a type \(E\) not containing \(X\)) the interpretation of \(e\) in \((A, f)\) is equal to its interpretation in \((B, g)\).

As one would expect, bisimilar dialgebras are observationally equivalent:
Theorem 4.5 Let \((A, f)\) and \((B, g)\) be \(\Sigma\)-dialgebras. If \((A, f)\) and \((B, g)\) are bisimilar then they are observationally equivalent.

**Proof** Let \(\sim\) a bisimulation between \((A, f)\) and \((B, g)\). For environments \(\eta\) and \(\xi\) we write \(\eta \sim \xi\) if \((\rho(x^E), \xi(x^E)) \in E^{rel}(\sim)\) for all \(x^E\) in their domains. Then we can prove that \([e]_\eta^A, [e]_\xi^B\) \(\in E^{rel}(\sim)\) for all \(e : E\) and for all \(\eta \sim \xi\), by induction on the derivation of \(e\). \(\square\)

**Behavioural Satisfaction**

We now consider a weaker notion of *behavioural* satisfaction of dialgebras satisfying specifications ‘up to some congruence relation’. First we define \([\Phi]_{A, f, \sim}\), the interpretation of \(\Phi\) in the dialgebra \((A, f)\) wrt. a congruence \(\sim\):

**Definition 4.6** For \(\sim\) a congruence on the dialgebra \((A, f)\), \([\Phi]_{A, f, \sim}^\eta\) is defined as \([\Phi]_{A, f}^\eta\), except that equality is interpreted as follows:
\[
[e_1 = e_2]_{A, f, \sim} = ([e_1]_{A, f, \sim}^\eta, [e_2]_{A, f, \sim}^\eta) \in E^{rel}(\sim)
\]

Note that if \(X\) does not occur in \(E\), this simply reduces to
\[
[e_1 = e_2]_{A, f, \sim} = ([e_1]_{A, f, \sim}^\eta, [e_2]_{A, f, \sim}^\eta)
\]

A notion of behavioural satisfaction can now be defined as follows:

**Definition 4.7 (Behavioural Satisfaction)** Let \((A, f)\) be a \(\Sigma\)-dialgebra and \(\sim\) a congruence relation for it. Then \((A, f)\) satisfies \(\Phi\) with respect to \(\sim\), written \((A, f) \models_\sim \Phi\), iff \([\Phi]_{A, f, \sim}\) is true.

**Theorem 4.8** \((A, f) \models_\sim \Phi\) iff \((A, f)/\sim \models \Phi\)

**Proof** By induction on the structure of \(\Phi\) we can prove that \([\Phi]_{A, f, \sim} = [\Phi]_{(A, f)/\sim}\). To prove this we must first prove relations between \([E]_{A, f}\) and \([E]_{(A, f)/\sim}\), and \([e]_{A, f}\) and \([e]_{(A, f)/\sim}\), namely \([E]_{A, f}/\sim = [E]_{(A, f)/\sim}\) and \([e]_{A, f}/\sim = [e]_{(A, f)/\sim}\). \(\square\)

Definitions and results similar to the ones above can be found in the literature for algebraic specifications, e.g., in [BHW95, HS96, Rei98b]. We do not know of any similar definitions or results in the literature on coalgebras or coalgebraic specifications. However, given that the notion of ‘observability’ plays a much more central role in the coalgebraic setting than in the algebraic setting, we believe that a notion of behavioural satisfaction makes even more sense for coalgebraic specifications than for algebraic ones.

More work would be needed to really exploit the opportunities offered by the notion of behavioural satisfaction when reasoning about specifications. In particular, one would want to establish that any consequences of a specification – in a particular logic – are not just valid for models satisfying the specification, but also for models behaviourally satisfying the specification. For algebras and algebraic specifications this idea is pursued in [BHW95, HS96, Rei98b]. In a type-theoretic setting, this idea is illustrated in [PZ99] and further investigated in [Zwa99]; the abstract data types considered here are more general than dialgebras. To be precise, the so-called first-order signatures considered in [Zwa99] are the \(\Sigma(X)\) generated by
\[
\begin{align*}
F(X) & ::= C \mid X \mid F(X) \times F(X) \\
\Sigma(X) & ::= F(X) \mid F(X) \rightarrow \Sigma(X) \mid \Sigma(X) \times \Sigma(X)
\end{align*}
\]
\(^2\)One could also take a partial congruence, but then the semantics of types would have to be changed with \([X] = \text{dom}(\sim)\).
5 Related Work

Several ways to combine algebras with coalgebras have been investigated over the past few years. One way of combining algebra with coalgebra is to consider pairs consisting of an algebra and a coalgebra, sometimes called bi-algebras. This is done in [HK99] and [COO]. Dialgebras are clearly more general than algebra-coalgebra pairs. Using an algebra-coalgebra pair rules out operations \( f : I N(X) \rightarrow OUT(X) \) where both \( I N_i(X) \neq X \) and \( OUT_i(X) \neq X \), for example a ‘partial’ binary operation \( f : X \times X \rightarrow 1 + X \).

In [TewOO] Tews introduces extended polynomial functors and coalgebras for these extended polynomial functors. This setting allows operations that are not possible in our dialgebra-setting, because a (restricted) use of \( \rightarrow \) is possible in output types. For example, \( g : X \rightarrow (C_1 \rightarrow X) + C_2 \) is a coalgebra for some extended polynomial functor, but cannot be an operation of any dialgebra. However, whereas our notion of dialgebras subsumes algebras, the setting of [Tew00] does not; this setting is still strictly coalgebraic and does not allow algebraic operations, not even one as simple as \( g : C \rightarrow X \). In OO terminology, the setting of [Tew00] allows binary methods but not constructors.

As for our dialgebras, for the coalgebras in [Tew00] bisimulations turn out to be closed under intersection and composition, but not under union. It would be interesting to see if a result similar to Theorem 3.20, i.e. closure under union for congruences, could be proved for extended polynomial functors.

Dialgebras and dialgebraic specifications can be regarded as special cases of the abstract data types and the specifications for abstract data types considered in a type-theoretic setting in [PZ99, Zwa99]. Such a type-theoretic setting is also used in [Han99, HanOO]. The crucial observation to link dialgebras with type theory is that dialgebras – and hence algebras and coalgebras – can be regarded as abstract datatypes. Abstract datatypes can be elegantly described in type theory using so-called existential types [MP88], and the logic for the notion of parametricity described in [PA93] then offers the expected proof rule for these existential types, namely that two implementations of an abstract datatype are equal if there exists a bisimulation between them.

This type-theoretic setting allows much wilder signatures than the dialgebraic signatures considered in this paper. This suggests further generalisations, for example with

- operations with higher-order types, e.g.
  \[
  \text{map} : Set_{Nat} \times (Nat \rightarrow Nat) \rightarrow Set_{Nat},
  \]
  or

- polymorphic operations, i.e. operations with type parameters, e.g.
  \[
  \text{polymorphicMap} : Set_{Nat} \rightarrow \forall \alpha. (Nat \rightarrow \alpha) \rightarrow Set_{\alpha}.
  \]

Finally, one of the referees drew our attention to [Rei98a], which introduces a notion of ‘nested sketches’ that support operations with arbitrarily structured in- and output types and seem more general than our dialgebras.

6 Conclusions

We have shown that the notion of dialgebra is a well-behaved generalisation of the notions of algebra and coalgebra. Dialgebras are more general than both algebras and coalgebras. The coalgebraic setting does for instance not allow ‘binary methods’, i.e. operations of type \( f : X \times X \rightarrow X \), or operations returning an initial state i.e. operations of type \( f : 1 \rightarrow X \). The algebraic setting does not allow operations with complicated return types, e.g. ‘partial’ operations \( f : X \rightarrow 1 + X \).

We have shown that many notions used in the fields of algebra and coalgebra are essentially identical, and can already be defined for the more general dialgebraic case: the (co)algebraic notions of homomorphism, invariant, bisimulation, and (partial) congruence can all be extended to dialgebras, preserving many of the essential properties.
We have also shown that dialgebraic specification provide a generalisation of (co)algebraic specification, and indicated how the notion of behavioural satisfaction, used in the field algebraic specification, can be extended to dialgebraic specifications (and hence also to coalgebraic specifications).

Given that (co)algebras are special cases of dialgebras, it is to be expected that some properties are lost when moving from algebras or coalgebras to dialgebras.

Most obviously, we no longer have the existence of initial c.q. final models. However, as far as dialgebraic specifications are concerned losing these properties maybe is not too bad, given that one is usually interested in loose semantics anyway.

In addition to this, useful properties of coalgebras that do not hold for arbitrary dialgebras are closure under union for invariants and bisimulations (Lemmas 3.10 and 3.16). The fact that we do not have these closure properties can be traced back to so-called binary methods, which are already notorious in the literature on object-oriented programming because of the problems they cause with subtyping (see [BCC+96]). For dialgebras without binary methods we do still have the properties that invariants and bisimulations are closed under union.

For a given functor there exists a canonical notion of ‘observational equality’ between elements of different coalgebras for that functor. An important consequence of the fact that for arbitrary dialgebras bisimulations are not closed under union, is that for a dialgebraic signature such a notion may not exist, as discussed in Section 3.3. However, in the dialgebraic setting we do still have a canonical notion of equality between elements of a single dialgebra, thanks to Theorem 3.20.

Future Work

Given the duality between algebras and coalgebras it is surprising that, whereas we did come across properties of coalgebras that do not hold for arbitrary dialgebras (namely closure properties for union, Lemmas 3.10 and 3.16), we did not come across (dual) properties of algebras that do not hold for arbitrary dialgebras. Carefully dualising Lemmas 3.10 and 3.16 might reveal such properties.

Another direction for future work is to further investigate which notions and results from the fields of algebra and coalgebra – notably the well-developed field of algebraic specifications – could be generalised to dialgebras.

Finally, the notion of dialgebra we have introduced is fairly ad-hoc and very syntactic. The motivation behind the definition of dialgebra was that it is the natural ‘unification’ of the definitions of algebra and coalgebra. It would be interesting to investigate more semantical characterisations of some notion of dialgebra, and to investigate in how far the restriction to polynomial functors could be relaxed, e.g. allowing the extended polynomial functors of [Tew00].

Acknowledgments

We would like to thank Bart Jacobs and Hendrik Tews for comments on earlier versions of this paper.

A More properties, proofs, and counterexamples

This appendix gives the details of the proofs of many properties mentioned in the paper, lists a few more properties, and gives some counterexamples.

A.1 Polynomial Signatures

Recall that polynomial signatures are signatures of the form

\[ F(X) ::= X \mid C \mid F_1(X) + F_2(X) \mid F_1(X) \times F_2(X). \]
Also recall that we deliberately exclude constant exponents of the form $C \rightarrow F(X)$ as polynomial
functors, because we have not been able to prove our most interesting result, Theorem 3.20, if we
include these.

Since polynomial signatures are functors, they have some nice properties:

**Lemma A.1** Let $F(X)$ be a polynomial signature. Then for predicate lifting

- $P \subseteq Q \Rightarrow F^{\text{pred}}(P) \subseteq F^{\text{pred}}(Q)$

and for relation lifting

- $R \subseteq S \Rightarrow F^{\text{rel}}(R) \subseteq F^{\text{rel}}(S)$

**Proof** Induction on the structure of $F(X)$. □

An easy consequence of these properties is

**Corollary A.2** Let $F(X)$ be a polynomial signature. Then for predicate lifting

- $F^{\text{pred}}(\bigcap_i P_i) \subseteq \bigcap_i F^{\text{pred}}(P_i)$
- $\bigcup_i F^{\text{pred}}(P_i) \subseteq F^{\text{pred}}(\bigcup_i P_i)$

and for relation lifting

- $F^{\text{rel}}(\bigcap_i R_i) \subseteq \bigcap_i F^{\text{rel}}(R_i)$
- $\bigcup_i F^{\text{rel}}(R_i) \subseteq F^{\text{rel}}(\bigcup_i R_i)$

**Proof** These properties easily follow from Lemma A.1:

- $P_i \subseteq \bigcup_i P_i$, so $F(P_i) \subseteq F(\bigcup_i P_i)$, and then $\bigcup_i F(P_i) \subseteq F(\bigcup_i P_i)$ by Lemma A.1.
- $\bigcap_i P_i \subseteq P_i$, so $F(\bigcap_i P_i) \subseteq F(P_i)$, and then $(\bigcap_i P_i) \subseteq \bigcap_i F(P_i)$ by Lemma A.1. □

We also have the properties

**Lemma A.3** Let $F(X)$ be a polynomial signature. Then for predicate lifting

- $F^{\text{pred}}(\bigcap_i P_i) = \bigcap_i F^{\text{pred}}(P_i)$

and for relation lifting

- $F^{\text{rel}}(\bigcap_i R_i) = \bigcap_i F^{\text{rel}}(R_i)$
- $F^{\text{rel}}(R; S) = F^{\text{rel}}(R) \cap F^{\text{rel}}(S)$

**Proof** Induction on the structure of $F(X)$. □

We do not have the corresponding properties for union, as shown by the counterexample below:

**Counterexample A.4** ($F^{\text{pred}}(\bigcup_i P_i) \neq \bigcup_i F^{\text{pred}}(P_i)$) Take $F(X) = X \times X$, and let $P$ and

$Q$ be predicates on some set. Then

$F^{\text{pred}}(P \cup Q) = (P \cup Q) \times (P \cup Q)$

$F^{\text{pred}}(P) \cup F^{\text{pred}}(Q) = (P \times P) \cup (Q \times Q)$

Clearly $F^{\text{pred}}(P \cup Q) \subseteq F^{\text{pred}}(P \cup Q)$, (as already stated by Corollary A.2) but not necessarilyvice versa. So in general we do not have preservation of unions,

$F^{\text{pred}}(\bigcup_i P_i) = \bigcup_i F^{\text{pred}}(P_i)$,

for polynomial signatures. □
For certain polynomial signatures $F(X)$ preservation of unions does hold, notably if there is at most one occurrence of $X$ in $F(X)$; see Lemma 2.8.

**Counterexample A.5** The properties of polynomial signatures $F$ given above in lemmas A.1, A.2, and A.3 do not hold for all signatures $\Sigma$, or even all dialgebraic signatures $\Sigma$:

- $P \subseteq Q \not\Rightarrow \Sigma^{\text{pred}}(P) \subseteq \Sigma^{\text{pred}}(Q)$
  For an example, simply take $\Sigma(X) = X \to X$. If $P \subseteq Q$, then it does not follow that $P \to P \subseteq Q \to Q$.

- $\Sigma^{\text{pred}}(\bigcap_i P_i) \not\subseteq \bigcap_i \Sigma^{\text{pred}}(P_i)$
  For an example, again take $\Sigma(X) = X \to X$. Then
  \[
  f \in (P \cap Q) \to (P \cap Q) \quad \iff \quad (\forall x \in P \cap Q. f(x) \in P \cap Q)
  \]
  and clearly
  \[
  \forall x \in P \cap Q. f(x) \in P \cap Q \not\Rightarrow (\forall x \in P. f(x) \in P) \land (\forall x \in Q. f(x) \in Q)
  \]
  Of course, we do have $\Leftarrow$.

- $\bigcup_i \Sigma^{\text{pred}}(P_i) \not\subseteq \Sigma^{\text{pred}}(\bigcup_i P_i), \Sigma^{\text{pred}}(\bigcup_i P_i) \not\subseteq \bigcup_i \Sigma^{\text{pred}}(P_i)$
  For an example, taking $\Sigma(X) = X \to X$ doesn’t work. Instead, take $\Sigma(X) = X \times X \to X$.
  Then
  \[
  f \in (P \times P \to P) \cup (Q \times Q \to Q) \quad \iff \quad (\forall x \in P \times P. f(x) \in P) \lor (\forall x \in Q \times Q. f(x) \in Q)
  \]
  and it is not hard to see that
  \[
  (\forall x \in P \times P. f(x) \in P) \lor (\forall x \in Q \times Q. f(x) \in Q)
  \]
  for arbitrary $P$ and $Q$.

The three counterexamples above apply to relation lifting as well as predicate lifting.

- $\Sigma^{\text{rel}}(R; S) \not\Rightarrow \Sigma^{\text{rel}}(R); \Sigma^{\text{rel}}(S)$
  Take $A = \{a_1, a_2, a_3, a_4\}, B = \{b_1, b_2\}$ and $C = \{c_1\}$. Consider the identity function $h$ on $C$ and the function $f : A \to A$ which maps $a_2$ to $a_3$, $a_3$ to $a_2$, and is the identity for $a_1$ and $a_4$ (as indicated in the diagram below). Consider the relations $R \subseteq A \times B$ and $S \subseteq B \times C$ as below, where a dotted line between e.g. $a_1$ and $b_1$ means that $a_1 R b_1$. 

![Diagram](image-url)
Note that $R; S = A \times C$, so it is not hard to see that $(f, h) \in \Sigma^{rel}(R; S)$. However, $(f, h) \not\in \Sigma(R); \Sigma(S)$, since there exist no $g : B \rightarrow B$ such that $(f, g) \in \Sigma^{rel}(R)$.

Suppose towards a contradiction that $(f, g) \in \Sigma^{rel}(R) = R \rightarrow^{rel} R$. Since $a_1 R b_1$ and $a_2 R b_1$ then $f(a_1) R g(b_1)$ and $f(a_2) R (b_1)$, i.e. $a_1 R g(b_1)$ and $a_3 R g(b_1)$, but clearly no such $g(b_1)$ exists.

**Lemma A.6** Let $F(X)$ be a polynomial signature.

(i) If $R$ is a reflexive, then so is $F^{rel}(R)$.

(ii) If $R$ is a symmetric, then so is $F^{rel}(R)$.

(iii) If $R$ is a transitive, then so is $F^{rel}(R)$.

(iv) $(F^{rel}(R^{*}))^{*} = F^{rel}(R^{*})$

**Proof**

(i) Let $R$ be reflexive, i.e. $Id_A \subseteq R$. Then $F(Id_A) = Id_{F(A)} \subseteq F(R)$ by Lemma 2.5.

(ii) Let $R$ be symmetric, i.e. $R = R^{op}$. Then $F(R)^{op} = F(R^{op}) = F(R)$ by Lemma 2.5.

(iii) Let $R$ be transitive, i.e. $R; R \subseteq R$. Then $F(R); F(R) \subseteq F(R; R) \subseteq F(R)$ by Lemma 2.7 (vii) and (i).

(iv) As $R^{*}$ is transitive and reflexive, then by (i) & (iii) so is $F(R^{*})$, and hence $(F(R^{*}))^{*} = F(R^{*})$.

**A.2 Dialgebraic Signatures**

Recall that a *dialgebraic signature* is a signature of the form

$$\Sigma(X) = \Sigma_1(X) \times \ldots \times \Sigma_n(X),$$

with each $\Sigma_i(X)$ of the form

$$\Sigma_i(X) = IN_i(X) \rightarrow OUT_i(X),$$

with $IN_i$ and $OUT_i$ polynomial signatures.

For dialgebraic signatures we do *not* have the properties mentioned in Lemma A.1 and Corollary A.2; the counterexamples in the previous sections are all dialgebraic signatures. However, we can salvage some of the properties mentioned in Lemma A.3:

**Lemma A.7** (aka Lemma 3.5) Let $\Sigma(X)$ be a dialgebraic signature, and let $i$ range over $I$, $I$ not empty. Then for predicate lifting

(i) $\bigcap_i \Sigma^{pred}(P_i) \subseteq \Sigma^{pred}(\bigcap_i P_i)$

and for relation lifting

(i) $\bigcap_i \Sigma^{rel}(R_i) \subseteq \Sigma^{rel}(\bigcap_i R_i)$

(ii) $\Sigma^{rel}(R); \Sigma^{rel}(S) \subseteq \Sigma^{rel}(R; S)$

**Proof**

(i) The proof (for binary intersection) was given in Lemma 3.5. The proofs of (ii) and (iii) below are very similar.
(ii) We just give the proof for the binary intersection.

\[(f, f') \in \Sigma(R \cap S)\]
\[\iff \forall j. (f_j, f'_j) \in \Sigma_j(R \cap S)\]
\[\iff \forall j. (f_j, f'_j) \in IN_j(R \cap S) \rightarrow OUT_j(R \cap S)\]
\[\iff \forall j. \forall (x, x') \in IN_j(R \cap S). (f_j(x), f'_j(x')) \in OUT_j(R \cap S)\]
\[\iff \forall j. \forall (x, x') \in IN_j(R) \cap OUT_j(S). (f_j(x), f'_j(x')) \in OUT_j(R) \cap OUT_j(S)\]
by Lemma A.3 (twice)
\[\iff \forall j. (f_j, f'_j) \in IN_j(R) \rightarrow OUT_j(R) \land (f_j, f'_j) \in IN_j(S) \rightarrow OUT_j(S)\]
\[\iff \forall j. (f_j, f'_j) \in IN_j(R) \land (f_j, f'_j) \in \Sigma(R \cap S)\]
\[\iff (f, f') \in \Sigma(R \cap S)\]

(iii) Similar to the proof of above:

\[(f, h) \in \Sigma(R); \Sigma(S)\]
\[\iff \exists g. (f, g) \in \Sigma(R) \land (g, h) \in \Sigma(S)\]
\[\iff \exists g. \forall j. (f_j, g_j) \in \Sigma_j(R) \land (g_j, h_j) \in \Sigma_j(S)\]
\[\iff \exists g. \forall j. (f_j, g_j) \in IN_j(R) \rightarrow OUT_j(R) \land (g_j, h_j) \in IN_j(S) \rightarrow OUT_j(S)\]
\[\iff \exists g. \forall j. (f_j, g_j) \in \Sigma(R) \land (g_j, h_j) \in \Sigma(S)\]
\[\iff \forall j. \forall (x, y, z) \in IN_j(R \cap S). (f_j(x), g_j(y), h_j(z)) \in OUT_j(R \cap S)\]
by Lemma A.3 (twice)
\[\iff \forall j. (f_j, h_j) \in IN_j(R; S) \rightarrow OUT_j(R; S)\]
\[\iff \forall j. (f_j, h_j) \in \Sigma_j(R; S)\]
\[\iff (f, h) \in \Sigma(R; S)\]
\[\square\]

We can say something interesting about union if we impose an additional condition on the dialgebraic signature.

Lemma A.8
Let \(\Sigma(X)\) be a dialgebraic signature, with

\[IN_j^{pred}(\bigcup_i P_i) = \bigcup_i IN_j^{pred}(P_i)\]
\[IN_j^{rel}(\bigcup_i R_i) = \bigcup_i IN_j^{rel}(R_i)\]

for all \(\Sigma_j(X) = IN_j(X) \rightarrow OUT_j(X)\).\(^3\) Then for predicate lifting

\[\bigcup_i \Sigma_j^{pred}(P_i) \subseteq \Sigma^{pred}(\bigcup_i P_i)\]

and for relation lifting

\[\bigcup_i \Sigma_j^{rel}(R_i) \subseteq \Sigma^{rel}(\bigcup_i R_i)\]

\(^3\) Note that by Corollary A.2 we already have \(\subseteq\).
Proof We just give the proof for the binary union of predicates:

\[
\begin{align*}
& f \in \Sigma(P) \cup \Sigma(Q) \\
\iff & \forall j. f_j \in \Sigma_j(P) \cup \Sigma_j(Q) \\
\iff & \forall j. f_j \in \text{IN}_j(P) \rightarrow \text{OUT}_j(P) \lor f_j \in \text{IN}_j(Q) \rightarrow \text{OUT}_j(Q) \\
\iff & \forall j. (\forall x \in \text{IN}_j(P), f_j(x) \in \text{OUT}_j(P)) \\
& \lor (\forall x \in \text{IN}_j(Q), f_j(x) \in \text{OUT}_j(Q)) \\
\Rightarrow & \forall j. \forall x \in \text{IN}_j(P) \cup \text{IN}_j(Q), f_j(x) \in \text{OUT}_j(P) \cup \text{OUT}_j(Q) \\
\iff & \forall j. \forall x \in \text{IN}_j(P) \cup \text{IN}_j(Q), f_j(x) \in \text{OUT}_j(P) \cup \text{OUT}_j(Q) \\
& \text{by assumption } \text{IN}_j(P \cup Q) = \text{IN}_j(P) \cup \text{IN}_j(Q) \\
\Rightarrow & \forall j. \forall x \in \text{IN}_j(P \cup Q), f_j(x) \in \text{OUT}_j(P \cup Q) \\
& \text{since } \text{OUT}_j(P) \cup \text{OUT}_j(Q) \subseteq \text{OUT}_j(P \cup Q) \text{ by Corollary A.2} \\
\iff & \forall j. f_j \in \text{IN}_j(P \cup Q) \rightarrow \text{OUT}_j(P \cup Q) \\
\iff & \forall j. f_j \in \Sigma_j(P \cup Q) \\
\iff & f \in \Sigma(P \cup Q)
\end{align*}
\]

If there is at most one occurrence of \(X\) in each of the \(J_i(X)\), then, by Lemma 2.8, Lemma A.8 above applies. This leads to Lemmas 3.12 and 3.17, which are immediate consequences of Lemma A.8 above.

### A.3 Dialgebraic Signatures & Unions of Bisims

For dialgebraic signatures, we do not have the property that bisimulations are closed under union, which we do have for coalgebras. Consequently, a largest bisimulation between two dialgebras cannot be defined in the same way as the unique largest bisimulation for two coalgebras, namely as the union of all bisimulations. However, we can salvage some of the coalgebras properties for dialgebras, by considering \textit{congruences} as defined in Def. 3.18, i.e. bisimulations between a dialgebra and itself.

**Lemma A.9** Basic properties of the operations \(+\) and \(\times\) on relations are:

(i) \((R + S)^+; (R' + S') = (R; R') + (S; S')\)

(ii) \((R \times S); (R' \times S') = (R; R') \times (S; S')\)

(iii) \(R^+ + S^+ = (R + S)^+\)

(iv) \(R^+ \times S^+ = (R \times S)^+\) if \(R\) and \(S\) are partially reflexive.

Here \(R^+\) denotes the transitive closure of \(R\), and \(R\) is \textit{partially reflexive} if \(\forall (x, y) \in R. (x, x) \in R \land (y, y) \in R\).

**Proof** Easy. \(\square\)

**Lemma A.10** Let \(F\) be a polynomial signature. Then \(F^{rel}(R^+) = F^{rel}(R)^+\) if \(R\) is partially reflexive.

**Proof** Induction on the structure of \(F\), using Lemmas A.9(iii) and (iv). \(\square\)

**Lemma A.11** Let \(i\) and \(j\) range over \(I\) and \(J\), respectively.

(i) \(\bigcup_i R_i + \bigcup_j S_j = \bigcup_{i,j} R_i + S_j\) if \(I\) and \(J\) are not empty.

(ii) \(\bigcup_i R_i \times \bigcup_j S_j = \bigcup_{i,j} R_i \times S_j\)

18
Lemma A.12 Let $F$ be a polynomial signature. Let $R_j$ be an equivalence relation on $A$, for all $j \in J$, $J \neq \emptyset$. Then $F^{ref}(\bigcup_j R_j) \subseteq \left( \bigcup_j F^{ref}(R_j) \right)^+$.

**Proof** Induction on the structure of $F$.

- $F(X) = C$ or $F(X) = X$: trivial.
- $F(X) = F_1(X) + F_2(X)$:
  \[
  \begin{align*}
  F_1(R_i) + F_2(R_j) &= F_1(R_i; Id_A) + F_2(Id_A; R_j) \\
  &= (F_1(R_i); F_1(Id_A)) + (F_2(Id_A); F_2(R_j)) & \text{by Lemma 2.7} \\
  &= (F_1(R_i) + F_2(Id_A)) ; (F_1(Id_A) + F_2(R_i)) & \text{by Lemma A.9(i)} \\
  &\subseteq (F_1(R_i) + F_2(R_i)) ; (F_1(R_i) + F_2(R_i)) & \text{since $Id_A \subseteq R_i$, so $F_k(Id_A) \subseteq F_k(R_i)$} \\
  &= F(R_i) ; F(R_j)
  \end{align*}
\]

so
\[
F\left( \bigcup_j R_j \right) = F_1\left( \bigcup_j R_j \right) + F_2\left( \bigcup_j R_j \right) \\
\subseteq \left( \bigcup_j F_1(R_i) \right)^+ + \left( \bigcup_j F_2(R_j) \right)^+ & \text{by IH} \\
= \left( \bigcup_j F_1(R_i) + \bigcup_j F_2(R_j) \right)^+ & \text{by Lemma A.9(iii)} \\
= \left( \bigcup_{i,j} F_1(R_i) + F_2(R_j) \right)^+ & \text{by Lemma A.11(i)} \\
\subseteq \left( \bigcup_{i,j} F_1(R_i) ; F_2(R_j) \right)^+ & \text{by the result above} \\
= \left( \left( \bigcup_j F(R_j) \right) ; \left( \bigcup_j F(R_j) \right) \right)^+ & \text{by Lemma A.9(i)} \\
\subseteq \left( \left( \bigcup_j F(R_j) \right) \right)^+ & \text{by definition of $+$}.
\]

- $F(X) = F_1(X) \times F_2(X)$: Analogous. \hfill \Box

Theorem A.13 (Closure of congruences under union)
Let $R_j$ be congruences on the $\Sigma$-dialgebra $(A, f)$, for all $j \in J$, $J \neq \emptyset$.

Then $\left( \bigcup_j R_j \right)^+$ is a congruence on $(A, f)$.

**Proof** We just do the proof for binary union.

Let $R$ and $S$ be congruences on $(A, f)$, i.e. $R$ and $S$ are equivalence relations, $(f, f) \in \Sigma(R)$, and $(f, f) \in \Sigma(S)$. To prove: $(f, f) \in \Sigma((R \cup S)^+)$, i.e.
\[
(f_i, f_i) \in \Sigma_i((R \cup S)^+) = IN_i\left((R \cup S)^+\right) \rightarrow OUT_i\left((R \cup S)^+\right)
\]
for all $i$.

Let $(x, x') \in IN_i\left((R \cup S)^+\right)$. To prove $(f_i(x), f_i(x')) \in OUT_i\left((R \cup S)^+\right)$.

\[
\begin{align*}
IN_i\left((R \cup S)^+\right) &= \left(IN_i(R \cup S)\right)^+ \text{ by Lemma A.10} \\
\subseteq \left(IN_i(R) \cup IN_i(S)\right)^+ & \text{by Lemma A.12} \\
= \left(IN_i(R) \cup IN_i(S)\right)^+ & \text{by definition of $+$}.
\end{align*}
\]

So $(x, x') \in (IN_i(R) \cup IN_i(S))^+$, i.e. there exist $x_1, \ldots, x_n$ such that
\[
(x, x_1), (x_1, x_2), \ldots, (x_n, x') \in IN_i(R) \cup IN_i(S).
\]
Since $R$ and $S$ are bisimulations it then follows that
\[
(f_1(x), f_1(x_1)), (f_1(x_1), f_1(x_2)), \ldots, (f_1(x_n), f_1(x')) \\
\in \text{OUT}_i(R) \cup \text{OUT}_i(S) \\
\subseteq \text{OUT}_i(R \cup S)
\]
by Lemma 2.7
and hence
\[
(f_1(x), f_1(x')) \in (\text{OUT}_i(R \cup S))^+ = \text{OUT}_i((R \cup S)^+) \quad \text{by Lemma A.10}
\]
\[\square\]

References


