Math. Struct. in Comp. Science (2017), *vol.* 27, *pp.* 1236–1264. © Cambridge University Press 2015
doi:10.1017/S0960129515000523 First published online 9 December 2015

Enhanced coalgebraic bisimulation

JURRIAAN ROT[†], FILIPPO BONCHI[‡], MARCELLO BONSANGUE[§,¶], DAMIEN POUS[‡], JAN RUTTEN[¶,‖] and ALEXANDRA SILVA[††]

[†]*Université de Lyon, CNRS, ENS de Lyon, UCBL, LIP, 46 Allée d'Italie, 69364 Lyon, France*
Email: jurriaan.rot@ens-lyon.fr
[‡]*CNRS, Plume team, LIP (UMR 5668, ENS de Lyon, UCBL, Université de Lyon), 46 Allée d'Italie, 69364 Lyon, France*
[§]*LIACS - Leiden University, Niels Bohrweg 1, 2333CA, The Netherlands*
Email: m.m.bonsangue@liacs.leidenuniv.nl
[¶]*Centrum Wiskunde en Informatica (CWI), Science Park 123, 1098 XG Amsterdam, The Netherlands*
[‖]*Radboud University Nijmegen, Toernooiveld 212, 6525 EC Nijmegen, The Netherlands*
[††]*University College London, Gower Street, London WC1E 6BT, U.K.*

Received 15 February 2013; revised 15 January 2015

We present a systematic study of bisimulation-up-to techniques for coalgebras. This enhances the bisimulation proof method for a large class of state based systems, including labelled transition systems but also stream systems and weighted automata. Our approach allows for compositional reasoning about the soundness of enhancements. Applications include the soundness of bisimulation up to bisimilarity, up to equivalence and up to congruence. All in all, this gives a powerful and modular framework for simplified coinductive proofs of equivalence.

1. Introduction

In the quest for good models of computation, the challenge of finding canonical notions of equivalence and corresponding proof methods has occupied the mind of many researchers. The pioneering work of Milner and Park (Milner 1980; Park 1981) on bisimulation has resulted in a vast amount of follow-up notions and improvements. Milner himself has proposed a powerful technique for modular reasoning about bisimilarity – bisimulation-up-to – which allows the re-use of existing bisimulation proofs and the construction of smaller relations to prove equivalence (Milner 1983). Sangiorgi (1998) has followed up on Milner's idea and proposed many enhancements to the theory of bisimulation-up-to for labelled transition systems. The gain of using bisimulations-up-to lies in the fact that they are smaller relations than usual bisimulations, thereby in many cases substantially reducing the amount of work and thus making the method more efficient. Bisimulation up to context is an example of an enhanced technique in which one can use the algebraic structure (syntax) of processes. Other examples are the notions of bisimulation up to union and bisimulation up to equivalence as well as combinations of any of these, which enable compositional, succinct reasoning on equivalence, combining both inductive and coinductive techniques.

In fact, some of the most useful up-to techniques are based on combinations of other enhancements. One of the difficulties in proving such up-to techniques to be sound is that the combination of sound enhancements is not necessarily sound. The first systematic study which addressed the issue of when such techniques can be safely combined is due to Sangiorgi (1998). While this work focused on labelled transition systems, a more general, abstract *algebra of enhancements* in terms of lattices and monotone functions has been introduced by Pous and Sangiorgi (Pous 2007; Pous and Sangiorgi 2012). An important feature there is the notion of *compatible* functions, defining a class of sound enhancements that is closed under composition.

Enhancements of the bisimulation proof method are interesting not only for labelled transition systems but also for other types of state-based systems; for example, recently an efficient algorithm for checking equivalence of non-deterministic automata was introduced, based on bisimulation up to congruence (Bonchi and Pous 2013). Other recent examples are the application of a different kind of up-to techniques for deterministic automata to proving language equivalence (Rot *et al.* 2013b), and up-to techniques for streams to facilitate coinductive definitions in Coq (Endrullis *et al.* 2013). Orthogonally to enhancements of the bisimulation proof method there is the theory of *coalgebra* in which the notion of bisimulation is extended to other models of computation, including all kinds of infinite data types, automata, and dynamical systems from a unifying perspective. By generalizing the theory of bisimulation-up-to to coalgebras one can study these techniques at a general level, with applications to many different types of state-based systems.

In the present paper, we establish the connection between coalgebraic bisimulation-up-to and the algebra of enhancements by using the characterisation of bisimulation in terms of monotone functions. This allows us to reason compositionally about the soundness of enhancements at the level of coalgebras. By showing that an up-to-technique is *compatible* one can now safely compose it with other compatible enhancements of coalgebraic bisimulation. We show that the most important enhancements are compatible.

In general many important instances of bisimulation-up-to, such as bisimulation up to equivalence and bisimulation up to bisimilarity, are not sound at the general level of coalgebras. We address this problem by a restriction to coalgebras for functors which preserve weak pullbacks; we prove the compatibility of such composition-based enhancements by using the theory of *relators* (Rutten 1998; Trnková 1980).

We show that bisimulation up to context is compatible whenever the system under consideration is a so-called λ-*bialgebra* for a distributive law λ (see, e.g. Bartels (2004); Klin (2011); Turi and Plotkin (1997)). Examples of such λ-bialgebras include non-deterministic and weighted automata but also operational models of specifications adhering to the *abstract GSOS* format (Turi and Plotkin 1997), which generalizes the well-known GSOS format (Bloom *et al.* 1995) for labelled transition systems. So even in the more classical case of labelled transition systems this generalizes the result of Sangiorgi (1998), who proved compatibility for the strictly less expressive De Simone format. Examples of operations which are expressible in GSOS but not in De Simone are the Kleene star and the priority operator (Aceto *et al.* 2001).

Most coalgebras considered in practice, such as labelled transition systems, stream systems and (non)-deterministic automata, are modelled by type functors which preserve weak pullbacks. However, there are important instances where this is not the case, including certain weighted transition systems (Bonchi *et al.* 2012; Gumm and Schröder 2001; Klin 2009). In such cases one can consider *behavioural equivalence*, which is a weaker notion of equivalence. To accommodate proofs of behavioural equivalence, in this paper we additionally introduce a compositional theory of up-to techniques for behavioural equivalence, most of which are sound independently of the type functor.

1.1. *Related work*

The first account of bisimulation-up-to at the level of coalgebras was given by Lenisa (Lenisa 1999; Lenisa *et al.* 2000). In Lenisa (1999), Lenisa considers a set-theoretic notion of coinduction and coinduction-up-to for abstract monotone operators, working in the direction of Pous and Sangiorgi (2012), and defines coalgebraic bisimulation-up-to-T for a monad T. However, in Lenisa (1999, page 22) the treatment of instances such as bisimulation up to bisimilarity are explicitly mentioned as an open problem. Interestingly, she conjectured that 'the theory of functors and relators could shed some light on this problem' which is indeed precisely the successful approach taken in the present work.

The up-to-context technique for coalgebraic bisimulation was later derived as a special case of so-called λ-coinduction (Bartels 2004). However, (Bartels 2004, pages 126, 129) mentions already that it would be ideal to combine the up-to-context technique with other enhancements. Indeed, combining up-to-context with up-to-bisimilarity or up-to-equivalence yields powerful proof techniques (see, e.g. Pous and Sangiorgi (2012) and this paper for examples). In this paper, we strengthen the soundness result of Bartels (2004) to *compatibility* of up-to-context, allowing for such combinations.

The recent paper (Zhou *et al.* 2010) introduces bisimulation-up-to where the notion of bisimulation is based on a specification language for polynomial functors (which does not include, for example, labelled transition systems). In contrast, we base our work on the standard notion of bisimulation, and only need to restrict to weak pullback preserving functors to obtain our soundness results. In the paper, (Luo 2006) coalgebraic bisimulation-up-to techniques are studied based on relation lifting. There, a concrete coalgebraic notion of compatibility, based on the notion of *consistency* proposed by Sangiorgi (Sangiorgi 1998) is introduced, and it is used to prove soundness of bisimulation up to context and of bisimulation up to bisimilarity (the latter is actually false in general, as we show in this paper). However, in Luo (2006) combinations of enhancements are not considered.

A new generalization of bisimulation-up-to to coalgebras was introduced by a subset of the authors in Rot *et al.* (2013a). In the present paper, we take this generalization as our starting point. The solution of Rot *et al.* (2013a) to the problem of unsoundness of bisimulation up to bisimilarity was, similarly to the present paper, to restrict to functors which preserve weak pullbacks. For such systems, bisimulation coincides with behavioural equivalence, and for the latter, the problematic up-to techniques were shown to be sound. In Rot *et al.* (2013a), the soundness of each of the enhancements and of their combinations had to be shown separately. Indeed, the problem of compositionality of enhancements, which we solve in this paper, was left as the main open problem.

Recently, a more abstract view on up-to techniques was introduced (Bonchi *et al.* 2014), in the setting of fibrations. The results there can be instantiated to obtain up-to techniques for coinductive predicates other than bisimilarity.

1.2. *Outline*

In Section 2, we recall coalgebras and bisimulations. Then in Section 3, we introduce bisimulation-up-to, together with the main instances and a number of examples. In Section 4, we recall the algebra of enhancements; Section 5 then presents bisimulation-up-to in terms of this theory. In Section 6, we prove compatibility results for the instances of bisimulation-up-to introduced in Section 3. Section 7 contains a similar development of up-to techniques for behavioural equivalence, and we present concluding remarks in Section 8.

1.3. *Notation*

Let Set be the category of sets and functions. Sets are denoted by capital letters $X, Y, \dots$ and functions by lower case $f, g, \dots$. We write *id* for the identity function and $g \circ f$ for function composition, defined by $(g \circ f)(x) = g(f(x))$. We write $f[S]$, for a function $f : X \to Y$ and a set $S \subseteq X$, to denote the image of S under f. Given sets X and Y, $X \times Y$ is the Cartesian product of X and Y (with the usual projection maps π_1 and π_2), X^Y is the set of functions $f : Y \to X$ and $\mathcal{P}(X)$ is the set of subsets of X. These operations, defined on sets, can analogously be defined on functions, yielding (bi-)functors. We write 2 for the two elements set $2 = \{0, 1\}$, ω for the set of natural numbers and $\mathbb{R}$ for the set of real numbers. By $\mathbb{R}^X_\omega$ we denote the set of functions $f : X \to \mathbb{R}$ with *finite support*, i.e. such that $f(x) \neq 0$ for finitely many elements x. We will write the elements v of $\mathbb{R}^X_\omega$ as a formal sum $v = f(x_1)x_1 + \cdots + f(x_n)x_n$. $\mathbb{R}^X_\omega$ carries a vector space structure where sum and scalar product (denoted by $+$ and $\cdot$) are defined pointwise: we call it the *free vector space* generated by X.

We denote the category of sets and relations by Rel. Relations are denoted by capital letters $R, S, \dots$ We write Δ for the identity relation and $R \circ S$ for relation composition, defined as usual: $R \circ S = \{(x, z) \in X \times Z \mid \exists y \text{ s.t. } xRy \text{ and } ySz\}$.

2. Coalgebra and bisimulation

We recall coalgebras and bisimulations, and make explicit the underlying notion of progression, which we need in the sequel. A *coalgebra* for a functor $F :$ Set $\to$ Set is a pair (X, α) consisting of a set X and a function $\alpha : X \to FX$. A function $f : X \to Y$ is an *(F-coalgebra) homomorphism* between (X, α) and (Y, β) if $Ff \circ \alpha = \beta \circ f$.

Definition 1. For a coalgebra $\alpha : X \to FX$ and relations $R, S \subseteq X \times X$, we say R *progresses to* S, denoted $R \rightarrowtail S$, if there exists a $\gamma : R \to FS$ making the following diagram commute:

$$\begin{tikzcd}
X \arrow[d, "\alpha"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\gamma"] & X \arrow[d, "\alpha"] \\
FX & FS \arrow[l, "F\pi_1"'] \arrow[r, "F\pi_2"] & FX
\end{tikzcd}$$

A *bisimulation* is a relation R such that $R \rightarrowtail R$.

Bisimulations are usually defined between two different systems, which however can be reduced to bisimulations on a single system by using coproducts (c.f., Appendix A). We use bisimulations on single systems for technical convenience and notational clarity. *Bisimilarity*, denoted by $\sim$, is defined as the largest bisimulation. Bisimulations can be seen as a proof technique for bisimilarity: for any two states $x, y \in X$, in order to prove that $x \sim y$ it suffices to exhibit a bisimulation R such that $x \mathrel{R} y$.

Example 1. Deterministic automata on the alphabet A are coalgebras for the functor $FX = 2 \times X^A$. Indeed, a deterministic automaton is a pair $(X, \langle o, t\rangle)$, where X is a set of states and $\langle o, t\rangle : X \to 2 \times X^A$ is a function with two components: o, the output function, determines if a state x is final ($o(x) = 1$) or not ($o(x) = 0$); and t, the transition function, returns for each input letter $a \in A$ the next state. Bisimilarity coincides with the standard notion of language equivalence, which can thus be proved by providing a suitable bisimulation. Unfolding the definition, a relation $R \subseteq X \times X$ is a bisimulation provided that for all $(x, y) \in R$: $o(x) = o(y)$ and, for all $a \in A$, $(t(x)(a), t(y)(a)) \in R$. As an example consider the automaton below, with final states y, z, v, w and transitions given by the solid arrows. The relation given by the four dashed lines together with the dotted line is a bisimulation.

$$\begin{tikzcd}
x \arrow[r, "{a,b}"] \arrow[dd, dash, dashed] & y \arrow[r, "{a,b}"] \arrow[d, dash, dashed] \arrow[dr, dash, dotted] & z \arrow[loop right, "{a,b}"] \arrow[d, dash, dashed] \arrow[dl, dash, dashed] \\
& v \arrow[r, bend left, "{a,b}"] & w \arrow[l, "{a,b}"] \\
u \arrow[ur, "a"] \arrow[urr, bend right, "b"] & &
\end{tikzcd}$$

Example 2. Labelled transition systems over a set of labels A are coalgebras for the functor $FX = \mathcal{P}(A \times X)$. An F-coalgebra (X, α) consists of a set of states X and a function $\alpha : X \to \mathcal{P}(A \times X)$ that maps each state $x \in X$ into a set of possible transitions (a, x'), where a is the label and x' is the arriving state. We write $x \xrightarrow{a} x'$ iff $(a, x') \in \alpha(x)$. Bisimilarity and bisimulation instantiate to the classical notions by Milner and Park (Milner 1980; Park 1981). A relation $R \subseteq X \times X$ is called a *bisimulation* provided that for all $(x, y) \in R$: if $x \xrightarrow{a} x'$ then there exists a state y' such that $y \xrightarrow{a} y'$ and $(x', y') \in R$, and vice versa.

Example 3. A weighted automaton with input alphabet A is a pair $(X, \langle o, t\rangle)$, where X is a set of states, $o : X \to \mathbb{R}$ is an output function associating to each state its output weight and $t : X \to (\mathbb{R}^X_\omega)^A$ is the transition relation that associates a weight to each transition. We shall use the following notation: $x \xrightarrow{a,r} y$ means that $t(x)(a)(y) = r$. Weight 0 means no transition.

Every weighted automaton induces a coalgebra for the functor $FX = \mathbb{R} \times X^A$ that is defined as $(\mathbb{R}^X_\omega, \langle o^\sharp, t^\sharp\rangle)$ where $\mathbb{R}^X_\omega$ is the free vector space generated by X and $o^\sharp : \mathbb{R}^X_\omega \to \mathbb{R}$ and $t^\sharp : \mathbb{R}^X_\omega \to (\mathbb{R}^X_\omega)^A$ are the linear extensions of o and t. For a detailed explanation see Bonchi *et al.* (2012, Section 3).

For example, consider the weighted automaton $(X, \langle o, t\rangle)$ depicted below in Equation (1), where we use $x{\downarrow}r$ to denote $o(x) = r$ and, as usual, arrows represent transitions. Part of the infinite corresponding F-coalgebra is depicted in Equation (2). Note that now states are vectors in $\mathbb{R}^X_\omega$ and that transitions are only labelled by symbols in A: the vector $v = \frac{1}{2}y_1 + \frac{1}{2}y_2 \in \mathbb{R}^X_\omega$ goes with a into $t^\sharp(\frac{1}{2}y_1 + \frac{1}{2}y_2)(a) = \frac{1}{2}t(y_1)(a) + \frac{1}{2}t(y_2)(a) = \frac{1}{4}y_1 + \frac{1}{4}y_2 + \frac{1}{2}y_2$.

In Bonchi *et al.* (2012) it is shown that bisimilarity on $(\mathbb{R}^X_\omega, \langle o^\sharp, t^\sharp\rangle)$ coincides with standard weighted language equivalence (Berstel and Reutenauer 1988; Salomaa and Soittola 1978) which can therefore be proved by means of bisimulations. A relation $R \subseteq \mathbb{R}^X_\omega \times \mathbb{R}^X_\omega$ is a bisimulation provided that for all $(v, w) \in R$: $o^\sharp(v) = o^\sharp(w)$ and, for all $a \in A$, $(t^\sharp(v)(a), t^\sharp(w)(a)) \in R$. For example, consider the weighted automaton below.

$$\begin{tikzcd}
 & x_3{\downarrow}0 \arrow[loop right, "{a,b,1}"] & & & y_3{\downarrow}0 \arrow[loop right, "{a,b,1}"] \\
x_0{\downarrow}0 \arrow[ur, bend left, "{b,1}"] \arrow[r, "{a,1}"] & x_1{\downarrow}1 \arrow[u, "{b,\frac{1}{2}}"'] \arrow[d, "{a,b,\frac{1}{2}}"] \arrow[loop right, "{a,\frac{1}{2}}"] & & y_0{\downarrow}0 \arrow[ur, bend left, "{b,1}"] \arrow[r, "{a,\frac{1}{2}}"] \arrow[dr, bend right, "{a,\frac{1}{2}}"'] & y_1{\downarrow}1 \arrow[u, "{b,1}"'] \arrow[d, "{a,\frac{1}{2}}"] \arrow[loop right, "{a,\frac{1}{2}}"] \\
 & x_2{\downarrow}1 \arrow[loop right, "{a,b,1}"] & & & y_2{\downarrow}1 \arrow[loop right, "{a,b,1}"]
\end{tikzcd} \tag{1}$$

The states x_0 and y_0 are weighted language equivalent. To formally prove it we exhibit a bisimulation $R \subseteq \mathbb{R}^X_\omega \times \mathbb{R}^X_\omega$ such that $(x_0, y_0) \in R$. Note that this relation is infinite since it must contain at least all the pairs given by the dotted lines below.

$$\begin{tikzcd}
x_0{\downarrow}0 \arrow[r, "a"] \arrow[d, dash, dotted] & x_1{\downarrow}1 \arrow[r, "a"] \arrow[d, dash, dotted] & \frac{1}{2}x_1 + \frac{1}{2}x_2{\downarrow}1 \arrow[r, "a"] \arrow[d, dash, dotted] & \frac{1}{4}x_1 + \frac{3}{4}x_2{\downarrow}1 \arrow[r, "a"] \arrow[d, dash, dotted] & \cdots \arrow[d, dash, dotted] \\
y_0{\downarrow}0 \arrow[r, "a"] & \frac{1}{2}y_1 + \frac{1}{2}y_2{\downarrow}1 \arrow[r, "a"] & \frac{1}{4}y_1 + \frac{3}{4}y_2{\downarrow}1 \arrow[r, "a"] & \frac{1}{8}y_1 + \frac{7}{8}y_2{\downarrow}1 \arrow[r, "a"] & \cdots
\end{tikzcd} \tag{2}$$

In Section 3, we will show that there exists a finite bisimulation up to context proving that x_0 and y_0 are bisimilar and therefore language equivalent.

Example 4. The notion of weighted automata from Example 3 can be generalized by replacing the field of reals $\mathbb{R}$ with any commutative semiring $\mathbb{S}$. As discussed in Bonchi *et al.* (2012), the coalgebraic characterization can be easily extended by taking the free semi-module $\mathbb{S}^X_\omega$ rather than the free vector space $\mathbb{R}^X_\omega$.

We now exhibit an example of a weighted automaton for the tropical semiring $\mathbb{T} = \langle \mathbb{R} \cup \{\infty\}, min, \infty, +, 0\rangle$. In this semiring, the addition operation is given by the function *min* having ∞ as neutral element. The multiplication is given by the function $+$ having 0 as neutral element.

The weighted automaton $(X, \langle o, t\rangle)$ below

$$\begin{tikzcd}
x{\downarrow}0 \arrow[r, bend left, "{a,2}"] \arrow[rr, bend right=40, "{a,3}"'] & y{\downarrow}0 \arrow[l, bend left, "{a,2}"] & z{\downarrow}0 \arrow[l, "{a,2}"'] & & u{\downarrow}0 \arrow[loop right, "{a,2}"]
\end{tikzcd} \tag{3}$$

induces the coalgebra $(\mathbb{T}^X_\omega, \langle o^\sharp, t^\sharp\rangle)$ which is partially depicted below.

$$\begin{tikzcd}
x{\downarrow}0 \arrow[r, "a"] \arrow[d, dash, dotted] & min(2+y, 3+z){\downarrow}2 \arrow[r, "a"] \arrow[d, dash, dotted] & min(4+x, 5+y){\downarrow}4 \arrow[r, "a"] \arrow[d, dash, dotted] & \cdots \arrow[d, dash, dotted] \\
u{\downarrow}0 \arrow[r, "a"] & (2+u){\downarrow}2 \arrow[r, "a"] & (4+u){\downarrow}4 \arrow[r, "a"] & \cdots
\end{tikzcd} \tag{4}$$

The states x and u are weighted language equivalent. To prove it we would need an infinite bisimulation, since it should relate all the pairs of states linked by the dotted lines in the above figure. In Section 3, we will exhibit a finite bisimulation up to congruence proving that x and u are language equivalent.

Example 5. We now consider *stream systems* (over the reals), which are coalgebras for the functor $FX = \mathbb{R} \times X$. At first, we take the set $\mathbb{R}^\omega = \{\sigma \mid \sigma : \omega \to \mathbb{R}\}$ of all streams (infinite sequences) of elements of $\mathbb{R}$ and we define the *initial value* $(-)_0 : \mathbb{R}^\omega \to \mathbb{R}$ and the *derivative* $(-)' : \mathbb{R}^\omega \to \mathbb{R}^\omega$ function as $(\sigma)_0 = \sigma(0)$, returning the first element or head of the stream, and $(\sigma)'(n) = \sigma(n+1)$, which returns the tail of the stream. The F-coalgebra $(\mathbb{R}^\omega, \langle(-)_0, (-)'\rangle)$ is called *final*, which means that from any F-coalgebra there exists a unique homomorphism into it Rutten (2000).

Then, we define operations on $\mathbb{R}^\omega$ by means of *behavioural differential equations* (Rutten 2003), in which an operation is defined by specifying its initial value $(-)_0$ and its derivative $(-)'$. These operations will become relevant in the examples in Section 3.

Differential equation	Initial value	Name
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)_0 = \sigma_0 + \tau_0$	sum
$(\sigma \otimes \tau)' = \sigma' \otimes \tau + \sigma \otimes \tau'$	$(\sigma \otimes \tau)_0 = \sigma_0 \times \tau_0$	shuffle product
$(\sigma^{-1})' = -\sigma' \otimes (\sigma^{-1} \otimes \sigma^{-1})$	$(\sigma^{-1})_0 = (\sigma_0)^{-1}$	shuffle inverse

In the second column, the operations $+$, $\times$ and $(-)^{-1}$ on the right of the equations are the standard operations on $\mathbb{R}$. The inverse is only defined on streams σ for which $\sigma_0 \neq 0$. With every real number r we associate a stream $\mathsf{r} = (r, 0, 0, 0, \ldots)$, and we abbreviate $(-\mathsf{1}) \otimes \sigma$ by $-\sigma$. The set of *terms* $T(\mathbb{R}^\omega)$ is defined by the grammar

$$t ::= \ \sigma \mid t_1 + t_2 \mid t_1 \otimes t_2 \mid t_1^{-1}$$

where σ ranges over $\mathbb{R}^\omega$. We call a term *well-formed* if the inverse is never applied to a subterm with initial value 0; this notion can be straightforwardly defined by induction. We can turn the set $T_{wf}(\mathbb{R}^\omega)$ of well-formed terms into a stream system, that is, an F-coalgebra $\mathcal{S} = (T_{wf}(\mathbb{R}^\omega), \langle(-)_0, (-)'\rangle)$ by defining $(-)_0 : T_{wf}(\mathbb{R}^\omega) \to \mathbb{R}$ and $(-)' : T_{wf}(\mathbb{R}^\omega) \to T_{wf}(\mathbb{R}^\omega)$ by induction on the structure of terms, using the above specification. For the base case, $\sigma \in \mathbb{R}^\omega$, we just use the final coalgebra structure given above by the initial value and derivative functions.

In Rutten (2003) it is shown that every term $t \in T_{wf}(\mathbb{R}^\omega)$ denotes a stream in $\mathbb{R}^\omega$ and that two terms t_1 and t_2 denote the same stream iff $t_1 \sim t_2$. As a result, in order to prove that two terms denote the same stream it is enough to exhibit a bisimulation relating them. A relation $R \subseteq T_{wf}(\mathbb{R}^\omega) \times T_{wf}(\mathbb{R}^\omega)$ is called a bisimulation provided that for all $(t_1, t_2) \in R$ it holds that $(t_1)_0 = (t_2)_0$ and $((t_1)', (t_2)') \in R$.

3. Bisimulation-up-to

The following definition generalizes the notion of bisimulation-up-to (Pous 2007; Pous and Sangiorgi 2012; Sangiorgi 1998) from labelled transition systems to coalgebras.

Definition 2. Let (X, α) be a coalgebra and $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ be a function on relations. A *bisimulation up to* f is a relation R such that $R \rightarrowtail f(R)$. We say that f is *sound* if $R \subseteq \sim$ for all R such that $R \rightarrowtail f(R)$.

If a function f is sound then giving a bisimulation up to f relating two states x and y is enough to prove that $x \sim y$. We now exhibit some functions that we will prove to be sound, under certain conditions, in Section 6. These conditions are satisfied in all the examples presented in this section.

3.1. *Up-to-equivalence*

Consider the function e mapping a relation R to its equivalence closure $e(R)$. A bisimulation up to e is called a bisimulation *up to equivalence*. Similarly one can define *up-to-transitivity* and *up-to-symmetry*.

Example 6. The relation R denoted by the four dashed lines in the automaton of Example 1 is *not* a bisimulation, since $((t(x)(b), t(u)(b)) = (y, w) \notin R$. However, R *is* a bisimulation up to equivalence, since the pair (y, w) is in $e(R)$. Hopcroft and Karp's algorithm (Hopcroft and Karp 1971) exploits this technique for checking equivalence of deterministic automata: rather than exploring n^2 pairs of states (where n is the number of states), the algorithm visits at most n pairs (that is the number of equivalence classes).

3.2. *Up-to-union*

For a fixed relation $S \subseteq X \times X$ consider the function $u_S(R) = R \cup S$. We call a bisimulation up to u_S a bisimulation *up to union with* S. Intuitively the successor states may be related either by R again or by S.

3.3. *Up-to-union-and-equivalence*

By composing the above functions e and u_S we obtain a new interesting up-to technique. If R is a bisimulation up to $e \circ u_S$ then we say R is a bisimulation *up to* S*-union and equivalence*.

Example 7. Recall Example 5 and suppose that we want to prove that the stream $\mathsf{1} = (1, 0, 0, \ldots)$ is the unit for the shuffle product $\otimes$, that is, $\sigma \otimes \mathsf{1} \sim \sigma$. We make use of the relation $R = \{(\sigma \otimes \mathsf{1}, \sigma) \mid \sigma \in T_{wf}(\mathbb{R}^\omega)\}$. For any $\sigma \in T_{wf}(\mathbb{R}^\omega)$, we have $(\sigma \otimes \mathsf{1})_0 = \sigma_0 \times \mathsf{1}_0 = \sigma_0$. Further $(\sigma \otimes \mathsf{1})' = \sigma' \otimes \mathsf{1} + \sigma \otimes \mathsf{1}' = \sigma' \otimes \mathsf{1} + \sigma \otimes \mathsf{0}$; this element is not in relation with σ', so R is not a bisimulation. However, given some basic laws of stream calculus, in particular $\sigma \otimes \mathsf{0} \sim \mathsf{0}$, $\sigma + \mathsf{0} \sim \sigma$ and the fact that $\sim$ is a congruence, we obtain

$$\sigma' \otimes \mathsf{1} + \sigma \otimes \mathsf{0} \sim \sigma' \otimes \mathsf{1} + \mathsf{0} \sim \sigma' \otimes \mathsf{1} \; R \; \sigma'$$

so R is a bisimulation up to $\sim$-union and equivalence and it proves that $\sigma \otimes \mathsf{1} \sim \sigma$.

3.4. *Up-to-bisimilarity*

Consider the function $b(R) = \sim \circ R \circ \sim$ which composes a relation on both sides with bisimilarity. A bisimulation up to b corresponds to the well-known concept of bisimulation *up to bisimilarity*, in which derivatives (i.e. the arriving states) do not need to be related directly but may be bisimilar to elements that are. Notice that every bisimulation up to bisimilarity is also a bisimulation up to $\sim$-union and equivalence. Since $\sim$ is transitive on stream systems, the relation R in Example 7 is also a bisimulation up to bisimilarity.

3.5. *Up-to-context*

When the state space of a coalgebra carries some kind of algebraic structure (as it is the case, for instance, with process algebras and regular expressions) it can be interesting to consider bisimulation up to *context*.

A T-algebra for an endofunctor T is a pair (X, β) where X is a set and $\beta : TX \to X$ is a function. For a T-algebra (X, β), the *contextual closure* of a relation $R \subseteq X \times X$ is defined as

$$c_\beta(R) = \langle \beta \circ T\pi_1, \beta \circ T\pi_2 \rangle [TR] = \{(\beta \circ T\pi_1(t), \beta \circ T\pi_2(t)) \mid t \in TR\}$$

Whenever β is clear from the context we simply write $c(R)$. If R is a bisimulation up to c then, we call R a *bisimulation up to context*.

Example 8. Given a signature Σ, i.e. a set of operations with associated arities, we consider the free T_Σ-algebra $\mu : T_\Sigma T_\Sigma X \to T_\Sigma X$. Intuitively, $T_\Sigma X$ consists of all Σ-terms with variables in X. Now, given a relation $R \subseteq T_\Sigma X \times T_\Sigma X$ on these terms, the contextual closure $c(R) \subseteq T_\Sigma X \times T_\Sigma X$ can be inductively characterized by the following rules, where g is any operator of Σ with arity n.

$$\frac{s \; R \; t}{s \; c(R) \; t} \qquad \frac{s_i \; c(R) \; t_i \qquad i = 1 \ldots n}{g(s_1, \ldots, s_n) \; c(R) \; g(t_1, \ldots, t_n)}$$

This slightly differs from the definition in Pous and Sangiorgi (2012) where the contextual closure is defined as

$$c'(R) = \{(C[s_1, \ldots s_n], C[t_1, \ldots t_n]) \mid C \text{ a context and for all } i\text{: } (s_i, t_i) \in R\}$$

(a context C is a term with $n \geqslant 0$ holes $[\cdot]_i$ in it). In our case c' can be obtained as $c \circ r$, i.e. by precomposing c with the *reflexive closure* function r.

Example 9. Recall from Example 3 that every weighted automaton $(X, \langle o, t \rangle)$ induces a coalgebra whose state space is the free vector space $\mathbb{R}^X_\omega$, that is, an algebra for the monad $\mathbb{R}^-_\omega$. Given a relation $R \subseteq \mathbb{R}^X_\omega \times \mathbb{R}^X_\omega$ its contextual closure $c(R) \subseteq \mathbb{R}^X_\omega \times \mathbb{R}^X_\omega$ can be inductively characterized by the following rules.

$$\frac{v \; R \; w}{v \; c(R) \; w} \qquad \frac{-}{0 \; c(R) \; 0} \qquad \frac{v_1 \; c(R) \; w_1 \qquad v_2 \; c(R) \; w_2}{v_1 + v_2 \; c(R) \; w_1 + w_2} \qquad \frac{v \; c(R) \; w \qquad r \in \mathbb{R}}{r \cdot v \; c(R) \; r \cdot w}$$

With the above characterization, it is easy to introduce bisimulation up to context for weighted automata: a relation $R \subseteq \mathbb{R}^X_\omega \times \mathbb{R}^X_\omega$ is a bisimulation up to context provided that for all $(v, w) \in R$ it holds that $o_1^\sharp(v) = o_2^\sharp(w)$ and for all $a \in A$, $(t_1^\sharp(v)(a), t_2^\sharp(w)(a)) \in c(R)$.

As an example consider the weighted automaton in Equation (1). It is easy to see that the relation $R = \{(x_2, y_2), (x_3, y_3), (x_1, \frac{1}{2}y_1 + \frac{1}{2}y_2), (x_0, y_0)\}$ is a bisimulation up to context: consider $(x_1, \frac{1}{2}y_1 + \frac{1}{2}y_2)$ (the other pairs are trivial) and observe that

$$\begin{tikzcd}
x_1 \arrow[r, "a"] \arrow[d, dotted, dash, "R"] & \frac{1}{2}x_1 + \frac{1}{2}x_2 \arrow[d, dotted, dash, "c(R)"] & x_1 \arrow[r, "b"] \arrow[d, dotted, dash, "R"] & \frac{1}{2}x_3 + \frac{1}{2}x_2 \arrow[d, dotted, dash, "c(R)"] \\
\frac{1}{2}y_1 + \frac{1}{2}y_2 \arrow[r, "a"'] & \frac{1}{4}y_1 + \frac{3}{4}y_2 & \frac{1}{2}y_1 + \frac{1}{2}y_2 \arrow[r, "b"'] & \frac{1}{2}y_3 + \frac{1}{2}y_2
\end{tikzcd}$$

It is worth noting that the above bisimulation up to context is finite, while one would need an infinite bisimulation to prove the equivalence of x_0 and y_0.

Example 10 (Rot *et al.* 2013b). The set $\mathcal{P}(A^*)$ of all languages forms a deterministic automaton as follows: the set of states is precisely the set of languages $\mathcal{P}(A^*)$ itself; a state $L \in \mathcal{P}(A^*)$ is accepting, i.e. $o(L) = 1$, if and only if the empty word ε is in L, and for every $a \in A$, the next state after an a-transition is given by the language derivative $t(L)(a) = \{w \mid aw \in L\}$. One can readily show that the language accepted by a state L is precisely L itself, and so whenever two languages L and K are bisimilar, they are in fact equal. The operations of language union $+$, composition $\cdot$ and Kleene star *, defined as usual, define an algebra on $\mathcal{P}(A^*)$. We have the following properties of derivatives of these operations due to Brzozowski; we formulate this in terms of languages (e.g. Conway (1971, page 41)):

$$\begin{array}{ll}
t(\mathsf{0})(a) = \mathsf{0} & o(\mathsf{0}) = 0 \\
t(\mathsf{1})(a) = \mathsf{0} & o(\mathsf{1}) = 1 \\
t(b)(a) = \begin{cases} \mathsf{1} & \text{if } b = a \\ \mathsf{0} & \text{otherwise} \end{cases} & o(b) = 0 \\
t(L + K)(a) = t(L)(a) + t(K)(a) & o(L + K) = o(L) \vee o(K) \\
t(L \cdot K)(a) = t(L)(a) \cdot K + [o(L)] \cdot t(K)(a) & o(L \cdot K) = o(L) \wedge o(K) \\
t(L^*)(a) = t(L)(a) \cdot L^* & o(L^*) = 1
\end{array}$$

for any languages L, K. Here, $\mathsf{0}$ denotes the empty language and $\mathsf{1}$ the language $\{\varepsilon\}$. Given $b \in 2$ we define $[b] \in \mathcal{P}(A^*)$ as $[0] = \mathsf{0}$ and $[1] = \mathsf{1}$.

Arden's rule states that if $L = KL + M$ for some languages L, K and M, and K does not contain the empty word, then $L = K^*M$. In order to prove its validity coinductively, let L, K, M be languages such that $\varepsilon \notin K$ and $L = KL + M$, and let $R = \{(L, K^*M)\}$. Then $o(K) = 0$ since by assumption $\varepsilon \notin K$, so

$$\begin{aligned}
o(L) &= o(KL + M) = (o(K) \wedge o(L)) \vee o(M) = (0 \wedge o(L)) \vee o(M) \\
&= o(M) = 1 \wedge o(M) = o(K^*) \wedge o(M) = o(K^*M)
\end{aligned}$$

and for any $a \in A$:

$$\begin{aligned} t(L)(a) &= t(KL+M)(a) = t(K)(a)\cdot L + [o(K)]\cdot t(L)(a) + t(M)(a) \\ &= t(K)(a)\cdot L + t(M)(a) \;\; c(R) \;\; t(K)(a)\cdot K^{*}M + t(M)(a) = t(K^{*}M)(a) \end{aligned}$$

So R is a bisimulation up to context, where the contextual closure is taken with respect to the operators of union and composition.

3.6. *Up-to-congruence*

By composing the functions e, c and r described above, we obtain another interesting up to technique. A bisimulation up to $e \circ c \circ r$ is called a bisimulation *up to congruence*. A recently introduced algorithm (Bonchi and Pous 2013), for checking the equivalence of non-deterministic automata, exploits this technique. The bisimulations up to congruence built by this algorithm can be exponentially smaller than bisimulation up to context. This is due to the use of *transitivity*.

Example 11. Recall from Example 4 the tropical semiring $\mathbb{T}$. Given a relation $R \subseteq \mathbb{T}_\omega^X \times \mathbb{T}_\omega^X$, its congruence closure can be inductively characterized by the following rules.

$$\frac{v\ R\ w}{v\ ecr(R)\ w} \qquad \frac{-}{v\ ecr(R)\ v} \qquad \frac{v\ ecr(R)\ w}{w\ ecr(R)\ v} \qquad \frac{u\ ecr(R)\ v\ ecr(R)\ w}{u\ ecr(R)\ w}$$

$$\frac{v_1\ ecr(R)\ w_1 \qquad v_2\ c(R)\ w_2}{min(v_1,v_2)\ ecr(R)\ min(w_1,w_2)} \qquad \frac{v\ ecr(R)\ w \qquad r \in \mathbb{R}\cup\{\infty\}}{r+v\ ecr(R)\ r+w}$$

For an example of bisimulation up to congruence consider the weighted automaton depicted in Equation (3) and the relation $R = \{(x,u),\ (min(2+y,3+z), 2+u)\}$. To prove that R is a bisimulation up to congruence we only have to show that $(min(4+x,5+y), 4+u) \in ecr(R)$:

$$\begin{aligned} min(4+x,5+y)\,ecr(R)\ & min(4+u,5+y) && ((x,u)\in R) \\ ecr(R)\ & min(min(4+y,5+z),5+y) && ((min(2+y,3+z),2+u)\in R) \\ =\ & 2+min(2+y,3+z) && \\ ecr(R)\ & 4+u && ((min(2+y,3+z),2+u)\in R) \end{aligned}$$

Note that R is not a bisimulation up to context, since $(min(4+x,5+y),4+u) \notin c(R)$. Here transitivity is really needed.

3.7. *Up-to-union-context-and-equivalence*

A bisimulation up to $e \circ c \circ u_S$ is called a *bisimulation up to S-union, context and equivalence.* This is an important extension of bisimulation up to context because the equivalence closure allows us to relate derivatives of R using $c(R \cup S)$ in 'multiple steps,' similar to the case of up-to-congruence.

Example 12. Recall the operations of shuffle product and inverse from Example 5 and suppose that we want to prove that the inverse operation is really the inverse of shuffle product, that is, $\sigma \otimes \sigma^{-1} \sim 1$ for all $\sigma \in T_{wf}(\mathbb{R}^{\omega})$ such that $\sigma_0 \neq 0$. Suppose we are given that $\otimes$ is associative and commutative (so $\sigma \otimes \tau \sim \tau \otimes \sigma$, etc.) and that $\sigma + (-\sigma) \sim 0$ (note that these assumptions actually hold in general (Rutten 2003)). Let

$$R = \{(\sigma \otimes \sigma^{-1}, 1) \mid \sigma \in T_{wf}(\mathbb{R}^{\omega}), \sigma_0 \neq 0\}.$$

We can now establish that R is a bisimulation up to $\sim$-union, context and equivalence. First consider the initial values:

$$(\sigma \otimes \sigma^{-1})_0 = \sigma_0 \times (\sigma^{-1})_0 = \sigma_0 \times (\sigma_0)^{-1} = 1 = 1_0$$

Next, we relate the derivatives by $e(c(R \cup \sim))$:

$$\begin{aligned}(\sigma \otimes \sigma^{-1})' &= \sigma' \otimes \sigma^{-1} + \sigma \otimes (\sigma^{-1})' \\ &= \sigma' \otimes \sigma^{-1} + \sigma \otimes (-\sigma' \otimes (\sigma^{-1} \otimes \sigma^{-1})) \\ &t(c(\sim))\ (\sigma' \otimes \sigma^{-1}) + (-(\sigma' \otimes \sigma^{-1}) \otimes (\sigma \otimes \sigma^{-1})) \\ &c(R \cup \sim)\ (\sigma' \otimes \sigma^{-1}) + (-(\sigma' \otimes \sigma^{-1}) \otimes 1) \\ &t(c(\sim))\ 0 = 1'\end{aligned}$$

where $t(c(\sim))$ denotes the transitive closure of $c(\sim)$; in the above we apply multiple substitutions of terms for bisimilar terms. Since $t(c(\sim)) \subseteq e(c(R \cup \sim))$ and $c(R \cup \sim) \subseteq e(c(R \cup \sim))$ we may conclude that R is a bisimulation up to $\sim$-union, context, and equivalence. Notice that R is not a bisimulation; establishing that it is a bisimulation-up-to is much easier than finding a bisimulation which contains R.

In the above, rather than $c(R \cup \sim)$ we could have used $c(r(R))$. Moreover, since in this example $\sim = t(c(\sim))$, the above is also an example of *bisimulation up to context, reflexivity and bisimilarity*, that is, a bisimulation up to $b \circ c \circ r$. (Any bisimulation up to context, reflexivity and bisimilarity is also a bisimulation up to $\sim$-union, context and equivalence.)

4. An algebra of enhancements

The above examples illustrate the large range of enhancements available for bisimilarity, and the need to combine such enhancements. For instance, up-to-union is rarely used on its own: it needs to be combined with up-to-equivalence or up-to-context. However, the soundness of such a combination does not necessarily follow from the soundness of its basic constituents, and it can be hard to prove it from scratch. This calls for a theory of enhancements which would allow one to freely combine them. Such a theory was developed at the rather abstract level of complete lattices (Pous 2007; Pous and Sangiorgi 2012). We rephrase it here at the level of binary relations, for the sake of clarity. We instantiate it in the following sections to obtain our general theory of coalgebraic bisimulations and behavioural equivalences up-to.

Let b be a monotone function on binary relations. By the Knaster–Tarski theorem b has a greatest fixpoint, denoted by $\mathsf{gfp}(b)$, which is also the greatest post-fixpoint:

$\mathsf{gfp}(b) = \bigcup\{R \mid R \subseteq b(R)\}$. The intuition is that by choosing b in an appropriate way, $\mathsf{gfp}(b)$ will be the desired notion of bisimilarity. This motivates the following terminology:

— A *b-simulation* is a relation R such that $R \subseteq b(R)$.
— *b-similarity* is the greatest b-simulation, i.e. $\mathsf{gfp}(b)$.

The bisimulation proof method can now be rephrased as follows: to prove that some states x, y are b-similar it suffices to exhibit a b-simulation R such that $x \ R \ y$. Enhancements of the bisimulation proof method allow one to weaken the requirement that R is a b-simulation: rather than checking $R \subseteq b(R)$, we would like to check $R \subseteq b(S)$ for a relation S which is possibly larger than R. The key idea consists in using a function f to obtain this larger relation out of R: $S = f(R)$.

Definition 3. Let f be a function on binary relations.

— A *b-simulation up to f* is a relation R such that $R \subseteq b(f(R))$.
— f is *b-sound* if all b-simulations up to f are contained in b-similarity.
— f is *b-compatible* if it is monotone and $f \circ b \subseteq b \circ f$.

The notion of b-compatible function is introduced to get around the fact that b-sound functions cannot easily be composed: b-compatible functions are b-sound and they enjoy nice compositionality properties:

Theorem 1. All b-compatible functions are b-sound.

Proof. Suppose that R is a b-simulation up to f, i.e. that $R \subseteq b(f(R))$. Using compatibility of f and by a simple induction on n, we get $\forall n, f^n(R) \subseteq b(f^{n+1}(R))$. Therefore, we have

$$\bigcup_n f^n(R) \subseteq \bigcup_n b(f^n(R)) \subseteq b\left(\bigcup_n f^n(R)\right).$$

(The second inclusion holds by monotonicity of b.) In other words, $f^\omega(R) = \bigcup_n f^n(R)$ is a b-simulation. This latter relation trivially contains R, by taking $n = 0$, so that we can conclude that R is contained in b-similarity. □

Proposition 1. The following functions are b-compatible:

1. *id* — the identity function;
2. $cons_S$ — the constant-to-S function, for any b-simulation S;
3. $f \circ g$ for any b-compatible functions f and g;
4. $\bigcup F$ for any set F of b-compatible functions.

The last two items allow one to freely combine b-compatible functions using functional composition and pointwise union. There is a third way of combining two functions f, g on relations, using relational composition: $f \bullet g(R) = f(R) \circ g(R)$. This composition operator does *not* always preserve b-compatible functions; the following lemma gives a sufficient condition:

Proposition 2. If b satisfies the following condition:

$$\text{for all relations } R, S, \quad b(R) \circ b(S) \subseteq b(R \circ S) \ , \qquad (\dagger)$$

then $f \bullet g$ is b-compatible for all b-compatible functions f and g.

We show in the following section that for all functors F there exists a function φ such that the F-bisimulations are the φ-simulations. Any such function is monotone and the property (†) holds iff the functor F preserves weak pullbacks.

We conclude this section with two lemmas which will be useful in the sequel: the first one gives an alternative characterization of b-compatible functions; the second one shows that b-similarity is closed under any b-compatible function.

Lemma 1. A monotone function f is b-compatible iff for all relations R, S, $R \subseteq b(S)$ implies $f(R) \subseteq b(f(S))$.

Lemma 2. For all b-compatible functions f, $f(\mathsf{gfp}(b)) \subseteq \mathsf{gfp}(b)$.

5. Bisimulation and φ-simulation

In this section, we show how to characterize bisimulation and bisimulation-up-to in terms of monotone functions. This allows us to study bisimulation-up-to, as introduced in Section 3, in terms of the abstract framework of Section 4.

Let (X, α) be an F-coalgebra. We define an endofunction φ_α on the complete lattice of relations on X ordered by inclusion $(\mathcal{P}(X \times X), \subseteq)$ as follows, based on *relation lifting* (Hermida and Jacobs 1998; Rutten 1998):

$$\begin{aligned}\varphi_\alpha(R) &= \{(x,y) \mid (\alpha(x), \alpha(y)) \in F(\pi_1^R)^{-1} \circ F(\pi_2^R)\} \\ &= \{(x,y) \mid \exists z \in FR \text{ s.t. } F(\pi_1^R)(z) = \alpha(x) \text{ and } F(\pi_2^R)(z) = \alpha(y)\}\end{aligned}$$

We write φ instead of φ_α if α is clear from the context.

Example 13. We describe φ for several concrete types of systems.

1. For deterministic automata, φ corresponds to the classical functional exploited by the Hopcroft minimization algorithm:

$$\varphi(R) = \{(x,y) \mid o(x) = o(y) \text{ and, for all } a \in A, (t(x)(a), t(y)(a)) \in R\}$$

2. In the case of labelled transition systems, φ corresponds to the well-known functional of bisimilarity (e.g. Sangiorgi (2012)):

$$\begin{aligned}\varphi(R) = \{(x,y) \mid\ & \text{if } x \xrightarrow{a} x' \text{ then there exists } y' \text{ s.t. } y \xrightarrow{a} y' \text{ and } x'Ry', \text{ and} \\ & \text{if } y \xrightarrow{a} y' \text{ then there exists } x' \text{ s.t. } x \xrightarrow{a} x' \text{ and } x'Ry'\}\end{aligned}$$

3. For stream systems, i.e. coalgebras for the functor $FX = \mathbb{R} \times X$, φ instantiates to $\varphi(R) = \{(x,y) \mid x_0 = y_0 \text{ and } x'Ry'\}$.

Notice that φ can be characterized as a pullback (e.g. Staton (2011)):

$$\begin{tikzcd} \varphi(R) \arrow[r] \arrow[d, hook] & F(\pi_1^R)^{-1} \circ F(\pi_2^R) \arrow[d, hook] \\ X \times X \arrow[r, "\alpha\times\alpha"] & FX \times FX \end{tikzcd}$$

Lemma 3. For any coalgebra (X, α): φ_α is monotone.

The following lemma establishes the connection of the above monotone functions to bisimulation and bisimulation-up-to.

Lemma 4. For any coalgebra (X, α) and for any relations $R, S \subseteq X \times X$:

$$R \subseteq \varphi_\alpha(S) \text{ iff } R \rightarrowtail S.$$

Proof. Follows easily from the second characterization of φ as given above. $\square$

From the above lemma we directly obtain the following known result (Rutten 1998):

Corollary 1. For any coalgebra (X, α): R is a bisimulation iff $R \subseteq \varphi_\alpha(R)$.

In other words, a φ-simulation (Section 4) is the same as a bisimulation. Thus, the greatest fixpoint of φ is precisely $\sim$. Lemma 4 also establishes a tight connection between bisimulation-up-to and φ-simulation-up-to.

Corollary 2. Let $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ be monotone. For any coalgebra (X, α):

1. $R \subseteq X \times X$ is a bisimulation up to f iff it is a φ_α-simulation up to f;
2. If f is φ_α-compatible (Definition 3), then f is sound (Definition 2).

Proof.

1. Follows directly from Lemma 4: $R \subseteq \varphi(f(R))$ iff $R \rightarrowtail f(R)$.
2. Suppose $R \rightarrowtail f(R)$; then $R \subseteq \varphi(f(R))$ by (1). If f is φ-compatible, then by Theorem 1 it is φ-sound. So $R \subseteq \mathsf{gfp}(\varphi) = \sim$.

$\square$

Via the above results we can apply the general theory of Section 4 to reason about coalgebraic bisimulation-up-to.

6. Compatibility

In this section, we study the φ-compatibility of the instances of bisimulation-up-to introduced in Section 3. By proving the compatibility of a function f we obtain the soundness of bisimulation up to f and we can compose it to other compatible functions, knowing that the result is again compatible.

Theorem 2. Let (X, α) be a coalgebra for a functor F. The following functions are φ_α-compatible:

1. r — the reflexive closure;
2. s — the symmetric closure;
3. u_S — union with S (for a bisimulation S);

If F preserves weak pullbacks, then the following are φ_α-compatible:

4. t — the transitive closure;
5. e — the equivalence closure;

6. b — bisimilarity;
7. $e \circ u_S$ — S-union and equivalence (for a bisimulation S).

The functions exploiting the contextual closure c will be considered later (Section 6.1). We will prove the above theorem below. But first, notice that for the compatibility of several functions we require the functor to preserve weak pullbacks. Indeed, bisimulation up to bisimilarity and bisimulation up to equivalence are not sound in general, and consequently not compatible either. This is illustrated by the following example, which is strongly inspired by an example from Aczel and Mendler (1989).

Example 14. Define the functor $F : \mathsf{Set} \to \mathsf{Set}$ as

$$FX = \{(x_1, x_2, x_3) \in X^3 \mid |\{x_1, x_2, x_3\}| \leqslant 2\}$$
$$F(f)(x_1, x_2, x_3) = (f(x_1), f(x_2), f(x_3))$$

Consider the F-coalgebra with states $X = \{0, 1, 2, \tilde{0}, \tilde{1}\}$ and transition structure

$$\begin{array}{lll} 0 \mapsto (0,1,0) & \tilde{0} \mapsto (0,0,0) & 2 \mapsto (2,2,2) \\ 1 \mapsto (0,0,1) & \tilde{1} \mapsto (1,1,1) & \end{array}$$

Then $0 \not\sim 1$. To see this, note that in order for the pair $(0, 1)$ to be contained in a bisimulation R, there must be a transition structure on this relation which maps $(0, 1)$ to $((0, 0), (1, 0), (0, 1))$. But this triple cannot be in FR, because it contains three different elements. However, it is easy to show that $0 \sim 2$ and $1 \sim 2$: the relation $\{(0, 2), (1, 2)\}$ is a bisimulation.

Now consider the relation $S = \{(\tilde{0}, \tilde{1}), (2, 2)\}$. S is not a bisimulation, since for that there should be a function from S to FS mapping the elements as follows:

$$(\tilde{0}, \tilde{1}) \mapsto ((0, 1), (0, 1), (0, 1)) \qquad (2, 2) \mapsto ((2, 2), (2, 2), (2, 2))$$

and neither $((0, 1), (0, 1), (0, 1))$ nor $((2, 2), (2, 2), (2, 2))$ are contained in FS. However, since $0 \sim 2\ S\ 2 \sim 1$ (and $2 \sim S \sim 2$), they *are* contained in $F(\sim S \sim)$; so S is a bisimulation up to bisimilarity. Thus if up-to-bisimilarity is sound, then $S \subseteq \sim$ so $0 \sim 1$, which is a contradiction.

Below we will show that if the functor preserves weak pullbacks then φ-compatible functions are closed under the operation $\bullet$ (defined in Section 4), which allows to prove items 4,5,6 and 7. In order to proceed we recall some fundamental results relating preservation of weak pullbacks to composition of relations.

Theorem 3. Let $F : \mathsf{Set} \to \mathsf{Set}$ be a functor. The following are equivalent:

1. F preserves weak pullbacks.
2. $\tilde{F} : \mathsf{Rel} \to \mathsf{Rel}$, defined as

$$\tilde{F}X = FX$$
$$\tilde{F}R = F(\pi_1^R)^{-1} \circ F(\pi_2^R)$$

   is a functor (i.e. $\tilde{F}$ preserves composition).
3. The composition of two F-bisimulations is again a bisimulation.

The equivalence of (1) and (2) is due to Trnková Trnková (1980). Notice that φ is in fact defined in terms of the action of $\tilde{F}$ on relations: $\varphi_\alpha(R) = \{(x,y) \mid (\alpha(x),\alpha(y)) \in \tilde{F}R\}$ (Rutten 1998). Rutten (2000) established the implication from (1) to (3). The reverse implication is due to Gumm and Schröder (2000). Their result is based on bisimulations on two coalgebras (X,α) and (Y,β) but for our notion of bisimulation (restricted to one coalgebra) the implication still holds, as we show in Appendix A.

Using Theorem 3 we show that preservation of weak pullbacks coincides precisely with the property (†) of Section 4. Then by Proposition 2 we obtain that φ-compatible functions are closed under $\bullet$ in the case of a functor which preserves weak pullbacks.

Proposition 3. F preserves weak pullbacks iff for any F-coalgebra (X,α), φ_α satisfies (†), i.e. for all relations R,S : $\varphi_\alpha(R) \circ \varphi_\alpha(S) \subseteq \varphi_\alpha(R \circ S)$.

Proof. Suppose F preserves weak pullbacks. Let (X,α) be an F-coalgebra, $R,S \subseteq X \times X$ relations, and $(x,z) \in \varphi_\alpha(R) \circ \varphi_\alpha(S)$, so there is some y such that $(x,y) \in \varphi_\alpha(R)$ and $(y,z) \in \varphi_\alpha(S)$. Then $(\alpha(x),\alpha(y)) \in \tilde{F}(R)$ and $(\alpha(y),\alpha(z)) \in \tilde{F}(S)$, so $(\alpha(x),\alpha(z)) \in \tilde{F}(R)\circ\tilde{F}(S)$. But by assumption and Theorem 3 $\tilde{F}$ is functorial, so $\tilde{F}(R)\circ\tilde{F}(S) = \tilde{F}(R\circ S)$. Consequently $(x,z) \in \varphi_\alpha(R \circ S)$ as desired.

Conversely, suppose that (†) holds; then by Proposition 2, compatible functions are closed under $\bullet$. Let R,S be bisimulations, so con_R and con_S are compatible by Proposition 1. By assumption $con_R \bullet con_S$ is compatible, so by Lemma 1 we have $R \circ S \subseteq \varphi(R \circ S)$. Now by Corollary 1, $R \circ S$ is a bisimulation. From Theorem 3 we conclude that F preserves weak pullbacks. □

The inverse function is compatible as well, which will be useful to prove compatibility of the equivalence closure:

Proposition 4. For any coalgebra (X,α): the inverse map $i(R) = R^{-1}$ is φ_α-compatible.

Proof. Suppose $R \subseteq \varphi(S)$, and let $(x,y) \in R^{-1}$, so $(y,x) \in R$. Then $(\alpha(y),\alpha(x)) \in (F(\pi_1^S))^{-1} \circ F(\pi_2^S)$. But $\pi_1^S = \pi_2^{S^{-1}}$ and $\pi_2^S = \pi_1^{S^{-1}}$, so $(\alpha(y),\alpha(x)) \in (F(\pi_2^{S^{-1}}))^{-1} \circ F(\pi_1^{S^{-1}})$. Consequently

$$(\alpha(x),\alpha(y)) \in ((F(\pi_2^{S^{-1}}))^{-1} \circ F(\pi_1^{S^{-1}}))^{-1} = (F(\pi_1^{S^{-1}}))^{-1} \circ F(\pi_2^{S^{-1}})$$

so $x,y \in \varphi(S^{-1})$. By Lemma 1, i is φ-compatible. □

We proceed with the proof of Theorem 2. Below we use the general compatibility results of Proposition 1 without further reference.

1. The identity relation Δ is a bisimulation (Rutten 2000) and thus, by Proposition 1, con_Δ is compatible and thus $r = id \cup con_\Delta$ is compatible.
2. The inverse function is compatible by Proposition 4. Compatibility of $s = id \cup i$ then follows directly.
3. $u_S = id \cup con_S$ is compatible for a bisimulation S, since con_S is compatible.
4. First, we define $(-)^n$ as $(-)^1 = id$ and $(-)^{n+1} = id\bullet(-)^n$. We prove that for all $n \geqslant 1$, $(-)^n$ is compatible, by induction on n. For the base case, notice that id is compatible. Now suppose $(-)^n$ is compatible. Then, by Proposition 3 and Proposition 2, $(-)^{n+1} = id\bullet(-)^n$

is also compatible. Now notice that $t = \bigcup_{n \geqslant 1} (-)^n$; so the function t is the (infinite) union of compatible functions, and consequently by Proposition 1 it is compatible.

5. $e = t \circ s \circ r$ is compatible, since r, s, and t are compatible.
6. $con_{\sim}$ is compatible since $\sim$ is a bisimulation. By Proposition 3 and Proposition 2, the function $b = con_{\sim} \bullet id \bullet con_{\sim}$ is compatible.
7. $e \circ u_S$ is compatible since e and u_S are compatible.

6.1. *Bisimulation up to context*

In order to define the contextual closure c, we need a T-algebra $\beta \colon TX \to X$ on the states of an F-coalgebra (X, α). For compatibility of c one might expect that it is enough to know that bisimilarity is a congruence with respect to this algebra; however, it is known that this is not even enough for soundness of bisimulation up to context (Pous and Sangiorgi 2012). As we will show below, in order to prove that c is compatible, it is sufficient to assume that (X, β, α) is a *λ-bialgebra* for some *distributive law* between T and F: a natural transformation $\lambda \colon TF \Rightarrow FT$. We refer to Klin (2011) for a nice overview on λ-bialgebras and report their formal definition below.

Definition 4. Let T, F be endofunctors on Set.

— An (F, T)-*bialgebra* is a triple (X, β, α) where X is a set, (X, β) is a T-algebra and (X, α) is an F-coalgebra.
— Given a natural transformation $\lambda \colon TF \Rightarrow FT$ we say (X, β, α) is a *λ-bialgebra* if $\alpha \circ \beta = F\beta \circ \lambda_X \circ T\alpha$.

For example, the coalgebra $(\mathbb{R}_\omega^X, \langle o^\sharp, t^\sharp \rangle)$ induced by a weighted automaton (Example 3) is a λ-bialgebra, where λ is a certain distributive law of the (underlying functor of the) free vector space monad $\mathbb{R}_\omega^-$ over the functor $FX = \mathbb{R} \times X^A$. Other important examples include certain types of process algebras and stream coalgebras induced by behavioural differential equations as well as regular expressions, but these examples involve a technicality treated in Section 6.2.

Theorem 4. Let (X, β, α) be a λ-bialgebra for $\lambda \colon TF \Rightarrow FT$. The contextual closure function c_β is φ_α-compatible. If F preserves weak pullbacks then the following are φ_α-compatible as well:

1. $e \circ c_\beta \circ r$ — congruence;
2. $e \circ c_\beta \circ u_S$ — context, S-union and equivalence;
3. $b \circ c_\beta \circ r$ — context, reflexivity and bisimilarity.

Proof. We prove compatibility of c; then items 1,2 and 3 follow directly from Theorem 2 and Proposition 1. Suppose $R \subseteq \varphi(S)$ for some R and S. Consider the following diagram:

$$\begin{tikzcd}
X \arrow[dd, "\alpha"'] & TX \arrow[l, "\beta"'] \arrow[d, "T\alpha"] & TR \arrow[l, "T\pi_1^R"'] \arrow[r, "T\pi_2^R"] \arrow[d, "T\gamma"] & TX \arrow[r, "\beta"] \arrow[d, "T\alpha"] & X \arrow[dd, "\alpha"] \\
 & TFX \arrow[d, "\lambda_X"] & TFS \arrow[l, "TF\pi_1^S"'] \arrow[r, "TF\pi_2^S"] \arrow[d, "\lambda_S"] & TFX \arrow[d, "\lambda_X"] & \\
FX & FTX \arrow[l, "F\beta"'] & FTS \arrow[l, "FT\pi_1^S"'] \arrow[r, "FT\pi_2^S"] & FTX \arrow[r, "F\beta"] & FX
\end{tikzcd}$$

The existence of γ and commutativity of the upper squares follow from Lemma 4 and an application of T. The lower squares commute by naturality. Finally, the outer rectangles commute since (X, β, α) is a λ-bialgebra.

Let $f_R \colon TR \to c(R)$ be the corestriction of $\langle \beta \circ T\pi_1^R, \beta \circ T\pi_2^R \rangle \colon TR \to X \times X$ to its range, so that $f_R[TR] = c(R)$. Let $f_S \colon TS \to c(S)$ be defined analogously, and take f_R^{-1} to be any right inverse of f_R. Then the following diagram commutes:

$$\begin{tikzcd}
 & & c(R) \arrow[dll, "\pi_1^{c(R)}"'] \arrow[drr, "\pi_2^{c(R)}"] \arrow[d, shift left, "f_R^{-1}"] & & \\
X \arrow[d, "\alpha"] & TX \arrow[l, "\beta"'] & TR \arrow[l, "T\pi_1^R"'] \arrow[r, "T\pi_2^R"] \arrow[u, shift left, "f_R"] \arrow[d, "\lambda_S \circ T\gamma"] & TX \arrow[r, "\beta"] & X \arrow[d, "\alpha"] \\
FX & FTX \arrow[l, "F\beta"'] & FTS \arrow[l, "FT\pi_1^S"'] \arrow[r, "FT\pi_2^S"] \arrow[d, "F(f_S)"] & FTX \arrow[r, "F\beta"] & FX \\
 & & Fc(S) \arrow[ull, "F\pi_1^{c(S)}"] \arrow[urr, "F\pi_2^{c(S)}"'] & &
\end{tikzcd}$$

So $c(R)$ progresses to $c(S)$, and consequently $c(R) \subseteq \varphi(c(S))$ by Lemma 4. By Lemma 1 we conclude that c is φ-compatible. $\square$

Remark 1. The greatest bisimulation on a λ-bialgebra is closed under the algebraic operations. This was first shown by Turi and Plotkin (1997) under the assumption that F preserves weak pullbacks; Bartels (2004) showed that this assumption is not necessary. We obtain the same result as a direct consequence of the above theorem and Lemma 2.

6.2. *Coalgebras for copointed functors*

As was first shown by Turi and Plotkin (1997) one can obtain process algebras whose operational rules conform to the GSOS format (Bloom *et al.* 1995) as λ-bialgebras. Every GSOS specification over some signature Σ induces an operational model

$$T_\Sigma T_\Sigma \emptyset \xrightarrow{\beta} T_\Sigma \emptyset \xrightarrow{\alpha} \mathcal{P}_f(A \times T_\Sigma \emptyset)$$

on closed terms, where β is the initial algebra and α is a transition structure induced by the specification. However, in several concrete cases there is no natural transformation

λ such that $(T_\Sigma\emptyset, \beta, \alpha)$ is a λ-bialgebra. Instead, one needs to consider the bialgebra $(T_\Sigma\emptyset, \beta, \langle\alpha, id\rangle)$; $\langle\alpha, id\rangle$ now is a coalgebra for the so-called *cofree copointed endofunctor* $\mathcal{P}_f(A \times Id) \times Id$ (see, e.g. Klin (2011)). Analogously, any (non-partial)[†] specification of operations on streams in terms of behavioural differential equations (Rutten 2003) corresponds to a natural transformation involving not the functor $FX = \mathbb{R} \times X$ but the functor $F \times Id$. Yet another example is given by the coalgebraic characterization of context-free grammars as in Winter *et al.* (2011); as discussed in Bonsangue *et al.* (2013), this construction involves a bialgebra which is a λ-bialgebra when the coalgebra is paired with the identity function.

All of the above are examples of bialgebras (X, β, α) such that $(X, \beta, \langle\alpha, id\rangle)$ is a λ-bialgebra. In such cases one wants to consider bisimulation(-up-to) on the coalgebra α and not on $\langle\alpha, id\rangle$. However, while Theorem 4 gives us $\varphi_{\langle\alpha,id\rangle}$-compatibility of the contextual closure c_β, it does *not* provide φ_α-compatibility. For the convenience of the reader, we recall a counterexample from Pous and Sangiorgi (2012).

Example 15 (Pous and Sangiorgi 2012). Consider the following specification of the prefix and the replication operation on labelled transition systems:

$$\frac{}{a.x \xrightarrow{a} x} \qquad\qquad \frac{x \xrightarrow{a} x'}{!x \xrightarrow{a} !x \mid x'}$$

together with the standard definition of the parallel operator $x \mid y$, and the constant 0. This specification is in the GSOS format, and since the variable x occurs on the right-hand side in the rule for replication, the use of a copointed functor is necessary (e.g. Klin (2011)). While this is arguably not the best way to specify replication in the context of CCS (Pous and Sangiorgi 2012) it suffices for our purposes. This specification induces a coalgebra on closed terms. Now consider the relations $R = \{(!a.b, !a.c)\}$ and $S = \{((!a.b) \mid b, (!a.c) \mid c)\}$ (where b and c abbreviate $b.0$ and $c.0$ respectively). Then R progresses to S, but $c(R)$ does not progress to $c(S)$. For example, $(d.!a.b, d.!a.c) \in c(R)$ but $!a.b$ is not related to $!a.c$ by $c(S)$. Thus, by Lemma 1 the contextual closure c is not φ_α-compatible.

The solution of Pous and Sangiorgi (2012) is to define a different function φ' as

$$\varphi'(R) = \varphi(R) \cap R\,.$$

But $\varphi'_\alpha = \varphi_{\langle\alpha,id\rangle}$ (a nice exercise in relation lifting), so in our framework this function arises naturally from the fact that one needs to consider a coalgebra for the cofree copointed functor in order to obtain compatibility.

In terms of progressions, we have $R \subseteq \varphi'(S)$ if and only if R progresses to S and $R \subseteq S$. Thus if R progresses to $f(R)$ for a function satisfying $R \subseteq f(R)$, then $R \subseteq \varphi'(f(R))$. But notice that for most functions f considered in Theorem 2 and Theorem 4 we have $R \subseteq f(R)$—an exception is the constant-to function. For the context closure function, a sufficient condition is that the functor T is *pointed*, i.e. there is a natural transformation $\eta : Id \Rightarrow T$, and β is an algebra for this pointed functor, meaning that $\beta \circ \eta = id$. In fact,

[†] The partial specification of Example 5 can be completed by fixing the initial value of 0^{-1} to some arbitrary constant.

in all the examples considered in this paper, T has a stronger property: is the underlying functor of a monad.

7. Behavioural equivalence-up-to

Whenever the functor F does not preserve weak pullbacks (as it is the case, for instance, with certain types of weighted transition systems (Bonchi *et al.* 2012; Gumm and Schröder 2001; Klin 2009)) one can consider *behavioural equivalence*, rather than bisimilarity.

Definition 5. For a coalgebra $\alpha : X \to FX$ and relations $R, S \subseteq X \times X$, we say R *progresses to* S (with respect to behavioural equivalence), denoted $R \rightsquigarrow S$, if the following diagram commutes:

$$\begin{tikzcd} R \arrow[r, shift left, "\pi_1"] \arrow[r, shift right, "\pi_2"'] & X \arrow[r, "\alpha"] & FX \arrow[r, "Fq"] & F(X/e(S)) \end{tikzcd}$$

where q is the quotient map of $e(S)$. If $R \rightsquigarrow R$ then R is called a *behavioural equivalence.*

Equivalently, R progresses to S if $Fq \circ \alpha$ factors through the quotient map of $e(R)$. In particular, R is a behavioural equivalence iff the quotient map of R is a coalgebra homomorphism.

The largest behavioural equivalence is denoted by $\approx$. An equivalent definition of $\approx$ is: $x \approx y$ iff there exists some homomorphism f from (X, α) to some coalgebra (Y, β) such that $f(s) = f(t)$ (Gumm 1999).

The relation R of Example 6 is a behavioural equivalence: note that, intuitively, behavioural equivalences are implicitly 'up-to-equivalence,' since the arriving states can be related by $e(R)$. Note also that in Aczel and Mendler (1989) behavioural equivalences are called *pre-congruences.*

Definition 6. If $R \rightsquigarrow f(R)$ for a function $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ then we say R is a *behavioural equivalence up to* f. We say that f is *sound* (w.r.t. behavioural equivalence) if $R \subseteq \approx$ for all R such that $R \rightsquigarrow f(R)$.

We proceed with a similar development as for bisimulation-up-to: first, we characterize behavioural equivalence as a fixed point of a monotone function, as done already in Aczel and Mendler (1989). Define the function $\psi_\alpha : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ as

$$\psi_\alpha(R) = \{(x, y) \mid Fq \circ \alpha(x) = Fq \circ \alpha(y)\}$$

i.e. as the kernel of $Fq \circ \alpha$, where $q : X \to X/e(R)$ is the quotient map of $e(R)$. Notice that we can also define q as the coequalizer of R and its projection maps, and ψ_α as the pullback of $Fq \circ \alpha$ along itself.

Lemma 5. For any coalgebra (X, α): ψ_α is monotone.

The correspondence between progression and functions ψ is given by the following lemma:

Lemma 6. For any coalgebra (X, α) and for any relations $R, S \subseteq X \times X$:

$$R \subseteq \psi(S) \text{ iff } R \rightsquigarrow S\,.$$

Consequently, behavioural equivalence up to any ψ-compatible function is sound. Unfortunately, the property (†) does *not* hold for ψ, that is, in general it does not hold that $\psi(R) \circ \psi(S) \subseteq \psi(R \circ S)$. This is shown by the following example:

Example 16. Consider the identity functor $FX = X$ and the F-coalgebra with states $\{x, y\}$ and transitions $x \mapsto x$ and $y \mapsto y$. Let $R = \{(x, y)\}$. Then $\psi(\emptyset) = \{(x, x), (y, y)\}$ and $\psi(R) = \{(x, x), (y, y), (x, y), (y, x)\}$. Now $\psi(R) \circ \psi(\emptyset) = \{(x, x), (y, y), (x, y), (y, x)\}$, whereas $\psi(R \circ \emptyset) = \psi(\emptyset) = \{(x, x), (y, y)\}$. So $\psi(R) \circ \psi(\emptyset)$ is not included in $\psi(R \circ \emptyset)$.

This motivates to prove the compatibility of the equivalence closure e directly, which is in fact quite easy in the case of behavioural equivalence.

Theorem 5. Let (X, α) be any coalgebra. The following are ψ_α-compatible:

1. r — the reflexive closure;
2. e — the equivalence closure;
3. u_S — union with S (for a behavioural equivalence S);

Proof. Items 1 and 3 are analogous to the case of φ-compatibility in Theorem 2. We proceed with the compatibility of the equivalence closure. First, notice that $e \circ \psi = \psi$ since $\psi(R)$ is an equivalence relation for any relation R. Second, since $e(R) = e(e(R))$ for any R, the quotient maps in the definition of $\psi(R)$ and $\psi(e(R))$ are equal, so $\psi(R) = \psi(e(R))$. Thus $e \circ \psi = \psi = \psi \circ e$. □

Notice that the ψ-compatibility of the equivalence closure does not require any assumptions on the functor.

In order to proceed recall that a *monad* is a triple (T, μ, η) where T is an endofunctor, $\mu\colon TT \Rightarrow T$ and $\eta\colon Id \Rightarrow T$ are natural transformations such that $\mu \circ T\eta = id = \mu \circ \eta T$ and $\mu \circ \mu T = \mu \circ T\mu$. A (T, μ, η)-algebra is T-algebra (X, β) such that $\beta \circ \eta_X = id$ and $\beta \circ \mu_X = \beta \circ T\beta$.

For the compatibility of context closure a λ-bialgebra is required, similar to the case of bisimulation in Theorem 4. However, in the case of behavioural equivalence, we require an algebra for a *monad*, although λ is still only required to be a distributive law between functors, that is, a plain natural transformation. Further, in the proof we need to assume preservation of reflexive coequalizers†, which is a non-trivial condition in Set; see (Adámek *et al.* 2000, page 538) for a counterexample. It is sufficient to restrict to *finitary* monads, that is, monads where the underlying functor preserves filtered colimits. For a free monad over a signature, this means that each operation has finite arity (but there may be infinitely many operations). Other examples include the free vector space monad and the finite powerset monad.

Lemma 7. If (T, η, μ) is a finitary Set monad then it preserves reflexive coequalizers.

† A reflexive coequalizer is a coequalizer of a reflexive pair, that is, a pair of functions $f, g\colon X \to Y$ such that there is a function $h\colon Y \to X$ with $f \circ h = g \circ h = id$.

Now we are ready to prove compatibility of up-to-context. However, notice that the above results depend on the relations being reflexive; thus we will directly prove compatibility of $c \circ r$ instead of c.

Theorem 6. Let (X, β, α) be a λ-bialgebra for a distributive law $\lambda \colon TF \Rightarrow FT$ (between functors), where β is an algebra for a finitary monad (T, η, μ). The following are ψ_α-compatible:

1. $c_\beta \circ r$ — contextual closure;
2. $e \circ c_\beta \circ r$ — congruence;
3. $e \circ c_\beta \circ r \circ u_S$ — context, S-union, reflexivity and equivalence;
4. $b \circ c_\beta \circ r$ — context, reflexivity and bisimilarity.

Proof. We only need to prove ψ-compatibility of $c_\beta \circ r$. Suppose $R \subseteq \psi_\alpha(S)$ for some relations $R, S \subseteq X \times X$. By Theorem 5, r is ψ_α-compatible, so $r(R) \subseteq \psi_\alpha \circ r(S)$. Further $r(S) \subseteq c_\beta \circ r(S)$, using the fact that we have a pointed functor (T, η). Therefore

$$r(R) \subseteq \psi_\alpha \circ c_\beta \circ r(S)\,. \tag{5}$$

Let $q : X \to X'$ be the quotient map of $e \circ c_\beta \circ r(S)$ and its projections, or, equivalently, the coequalizer of the two composite arrows $\beta \circ T\pi_1$, $\beta \circ T\pi_2$ in the bottom of the diagram below:

$$\begin{tikzcd}
TT(r(S)) \arrow[r, "TT\pi_1{,}TT\pi_2"] \arrow[d, "\mu_{r(S)}"] & TTX \arrow[r, "T\beta"] \arrow[d, "\mu_X"] & TX \arrow[r, "Tq"] \arrow[d, "\beta"] & TX' \arrow[d, dashed, "\beta'"] \\
T(r(S)) \arrow[r, "T\pi_1{,}T\pi_2"] & TX \arrow[r, "\beta"] & X \arrow[r, "q"] & X'
\end{tikzcd} \tag{6}$$

The left square commutes (for $T\pi_1$ and $T\pi_2$ separately) by naturality, and the middle since β is an algebra for the monad. Since T is finitary, it preserves reflexive coequalizers, so Tq is a coequalizer. The map β' making the right square commute arises by its universal property.

Now consider the following diagram:

$$\begin{tikzcd}
T(r(R)) \arrow[r, shift left, "T\pi_1"] \arrow[r, shift right, "T\pi_2"'] & TX \arrow[r, "T\alpha"] \arrow[dd, "\beta"] & TFX \arrow[r, "TFq"] \arrow[d, "\lambda_X"] & TFX' \arrow[d, "\lambda_{X'}"] \\
 & & FTX \arrow[r, "FTq"] \arrow[d, "F\beta"] & FTX' \arrow[d, "F\beta'"] \\
 & X \arrow[r, "\alpha"] & FX \arrow[r, "Fq"] & FX'
\end{tikzcd}$$

The top horizontal paths commute by Equation (5) and functoriality. The rectangle commutes by the assumption that (X, β, α) is a λ-bialgebra. The upper square commutes by naturality of λ, and the lower square by Equation (6) and functoriality. Thus we have

$Fq \circ \alpha \circ \beta \circ T\pi_1 = Fq \circ \alpha \circ \beta \circ T\pi_2$, and consequently

$$\begin{tikzcd} c_\beta(r(R)) \arrow[r, shift left, "\pi_1"] \arrow[r, shift right, "\pi_2"'] & X \arrow[r, "\alpha"] & FX \arrow[r, "Fq"] & FX' \end{tikzcd}$$

commutes, which means $c_\beta \circ r(R) \rightsquigarrow c_\beta \circ r(S)$. Thus $c_\beta \circ r(R) \subseteq \psi_\alpha \circ c_\beta \circ r(S)$ by Lemma 6, and by Lemma 1 now $c_\beta \circ r$ is ψ_α-compatible. □

For compatibility of the contextual closure function for coalgebras which are (part of) λ-bialgebras when paired with the identity function, we can perform a similar development as in Section 6.2, by defining $\psi'(R) = \psi(R) \cap R$.

Example 17. For an example of behavioural equivalence up-to, we consider the 'general process algebra with transitions costs' (GPA) from Buchholz and Kemper (2001). GPA processes are defined for a given set of labels A and a semiring $\mathbb{S}$ which, for this example, we fix to be the semiring of reals $\mathbb{R}$. The operational semantics of GPA is expressed in terms of *weighted transition systems* which are coalgebras for the functor $(\mathbb{R}_\omega^-)^A$ where $\mathbb{R}_\omega^- : \mathsf{Set} \to \mathsf{Set}$ is defined as follows:

— For each set X, $\mathbb{R}_\omega^X$ is the set of functions from X to $\mathbb{R}$ with finite support (see the *Notation* paragraph in the introduction).
— For each function $h : X \to Y$, $\mathbb{R}_\omega^h : \mathbb{R}_\omega^X \to \mathbb{R}_\omega^Y$ is the function mapping each $\varphi \in \mathbb{R}_\omega^X$ into $\varphi^h \in \mathbb{R}_\omega^Y$ defined, for all $y \in Y$, by

$$\varphi^h(y) = \sum_{x' \in h^{-1}(y)} \varphi(x')$$

In a nutshell, a weighted transition system is a weighted automaton without output in the states. Formally, it is a pair (X, t) where X is a set of states and $t : X \to (\mathbb{R}_\omega^X)^A$ is the transition relation that associates a weight to each transition. We use the same notation of weighted automata: $x \xrightarrow{a,r} y$ means that $t(x)(a)(y) = r$ and $r \neq 0$.

As shown in Section 2.3 of Bonchi *et al.* (2012), the functor $(\mathbb{R}_\omega^-)^A$ does not preserve weak pullbacks and therefore bisimulations up-to cannot be used in this context. However, thanks to Theorems 5 and 6 we can use behavioural equivalence up-to. First observe that, by instantiating the definition of ψ above to an $(\mathbb{R}_\omega^-)^A$-coalgebra (X, t), one obtains the function $\psi_t : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ defined for all relations $R \subseteq X \times X$ as

$$\psi_t(R) = \{(x_1, x_2) \mid \forall a \in A, y \in X, \sum_{y' \in [y]_R} t(x_1)(a)(y') = \sum_{y' \in [y]_R} t(x_2)(a)(y')\}$$

where $[y]_R$ denotes the equivalence class of y w.r.t. R. With this explicit definition, it is easy to see that our notion of behavioural equivalence coincides with the one of bisimulation in Buchholz and Kemper (2001).

In order to illustrate our example is enough to consider a small fragment of GPA. The set P of basic GPA processes is defined by

$$p ::= 0 \mid p + p \mid (a, r).p$$

where $a \in A$, $r \in \mathbb{R}$. The operational semantics of basic GPA processes is given by the coalgebra $t : P \to (\mathbb{R}^P)^A$ defined for all $a' \in A$ and $p' \in P$ as follows:

$$\begin{aligned} t(0)(a')(p') &= 0 \\ t((a,r).p)(a')(p') &= \begin{cases} r & \text{if } a = a', p = p' \\ 0 & \text{otherwise} \end{cases} \\ t(p_1 + p_2)(a')(p') &= t(p_1)(a')(p') + t(p_2)(a')(p') \end{aligned}$$

Equivalently, it is described by the following rules:

$$\frac{r \neq 0}{(a,r).p \overset{a,r}{\to} p} \qquad \frac{p_1 \overset{a,r}{\to} p' \quad r \neq 0}{p_1 + p_2 \overset{a,s}{\to} p'} \qquad \frac{p_2 \overset{a,r}{\to} p' \quad r \neq 0}{p_1 + p_2 \overset{a,s}{\to} p'}$$

where $s = t(p_1)(a)(p') + t(p_2)(a)(p')$. For instance, the operational semantics of $(a,1).0 + (a,-1).(a,0).0$ is depicted below.

$$\begin{tikzcd} & 0 \\ (a,1).0 + (a,-1).(a,0).0 \arrow[ur, "{a,1}"] \arrow[dr, "{a,-1}"'] & \\ & (a,0).0 \end{tikzcd}$$

Since $0 \approx (a,0).0$, we have that $(a,1).0 + (a,-1).(a,0).0 \approx 0$. More generally, it holds that for all $a \in A$, $r \in R$, p_1 and $p_2 \in P$,

$$\text{if } p_1 \approx p_2 \text{ then } 0 \approx (a,r).p_1 + (a,-r).p_2. \tag{7}$$

We are going to prove Equation (7) by means of behavioural equivalence up to union with $\approx$ (Theorem 5). To this end, consider the relation

$$R = \{(0, (a,r).p_1 + (a,-r).p_2) \mid p_1 \approx p_2\}$$

and note that R is *not* a behavioural equivalence by taking $p_1 = 0$ and $p_2 = (a,0).0$ (namely, $R \not\subseteq \psi_t(R)$). However, R is a behavioural equivalence up to $u_\approx$: to see that $R \subseteq \psi_t(R \cup \approx)$, fix $p = (a,r).p_1 + (a,-r).p_2$ and observe that for all processes $q \in P$

$$\sum_{y' \in [q]_{R\cup\approx}} t(0)(a)(y') = 0 = \sum_{y' \in [q]_{R\cup\approx}} t(p)(a)(y').$$

The leftmost equality comes from the semantics of the process 0. For the rightmost, we have that either $q \not\approx p_1$ or $q \approx p_1$. In the first case, the above rightmost equivalence is obvious. In the second case,

$$\sum_{y' \in [q]_{R\cup\approx}} t(p)(a)(y') = t(p)(a)(p_1) + t(p)(a)(p_2) = r - r = 0$$

since $p_1 \in [q]_{R\cup\approx}$ and $p_2 \in [q]_{R\cup\approx}$.

8. Conclusions

Coalgebraic bisimulation-up-to enhances the proof method for bisimilarity, allowing for smaller proofs and equational reasoning on bisimulation equivalence for a large class of state-based systems and calculi. We presented a compositional framework for up-to-techniques and showed the compatibility (and thus the soundness) of the more common techniques: any novel compatible enhancements could be combined with these as well, without the necessity of re-proving soundness.

While showing this we also obtained interesting side results, such as Proposition 3, which provides a novel characterization of weak pullback preservation. This result is based on a definition of relational lifting for weak pullback preserving functors. We leave as future work the study of either a more broadly applicable notion of bisimulation, as in Gorín and Schröder (2013), or a more general definition of relation lifting, which applies to arbitrary functors on Set, as in Marti and Venema (2012). While in our work relators must preserve binary composition, in Levy (2011) a framework has been developed which only laxly preserves composition. The combination of this theory with our framework would allow for a study of up-to-techniques for simulations, rather than bisimulations.

The research of the first author was carried out at Leiden University and CWI, and was funded by the Netherlands Organisation for Scientific Research (NWO), CoRE project, dossier number: 612.063.920. The first author is supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). The second and the fourth author acknowledge support by project `ANR 12IS0 2001 PACE`.

Appendix A. Bisimulations on different systems

The validity of the implication (3) $\Rightarrow$ (1) of Theorem 3 is shown in Gumm and Schröder (2000), based on the standard notion of bisimulation on different systems: given F-coalgebras (X, α_X) and (Y, α_Y) a relation $R \subseteq X \times Y$ is a bisimulation if there exists a transition structure $\gamma : R \to FR$ such that the following diagram commutes:

$$\begin{tikzcd}
X \arrow[d, "\alpha_X"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\gamma"] & Y \arrow[d, "\alpha_Y"] \\
FX & FR \arrow[l, "F\pi_1"] \arrow[r, "F\pi_2"'] & FY
\end{tikzcd}$$

The notion of bisimulation which we adopted in this paper is based on single systems, i.e. where $(X, \alpha_X) = (Y, \alpha_Y)$. We proceed to show that in Set, if bisimulations on single systems are closed under composition, then bisimulations on different systems are closed under composition as well; this proves that the implication from (3) to (1) of Theorem 3 holds in our setting as well.

We denote a coproduct by $X + Y$ and the associated injections by i_X and i_Y.

Proposition 5. Let (X, α_X) and (Y, α_Y) be F-coalgebras (F is a Set endofunctor) and $R \subseteq X \times Y$ a relation. Then R is a bisimulation on X and Y iff $(i_X \times i_Y)[R]$ is a bisimulation on $X + Y$.

Proof. Let $R \subseteq X \times Y$. The cases $X = \emptyset$ or $Y = \emptyset$ are trivial, so we may assume $X \neq \emptyset$ and $Y \neq \emptyset$. Let $(X + Y, \alpha_{X+Y})$ be the coproduct coalgebra (Rutten 2000). So in the diagram below, the outer two squares commute. Suppose R is a bisimulation on X and Y; then there exists a γ such that the middle squares of the diagram below commute:

$$\begin{tikzcd}
X+Y \arrow[d, "\alpha_{X+Y}"] & X \arrow[l, "i_X"'] \arrow[d, "\alpha_X"] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "\gamma"] & Y \arrow[r, "i_Y"] \arrow[d, "\alpha_Y"'] & X+Y \arrow[d, "\alpha_{X+Y}"] \\
F(X+Y) & X \arrow[l, "Fi_X"'] & FR \arrow[l, "F\pi_1"'] \arrow[r, "F\pi_2"] & FY \arrow[r, "Fi_Y"] & F(X+Y)
\end{tikzcd}$$

which means the entire diagram commutes and $(i_X \times i_Y)(R)$ is a bisimulation on $X + Y$.

Conversely suppose $((i_X \times i_Y)[R], \gamma)$ is a bisimulation on $X + Y$ for some relation $R \subseteq X \times Y$; so in the above diagram, the outer rectangles commute (i.e. $\alpha_{X+Y} \circ i_X \circ \pi_1 = Fi_X \circ F\pi_1 \circ \gamma$ and similarly for Y). Now since i_X is mono, X is nonempty, and F is a Set functor, Fi_X is mono as well (see, e.g. Rutten (2000)). Further $Fi_X \circ \alpha_X \circ \pi_1 = \alpha_{X+Y} \circ i_X \circ \pi_1 = Fi_X \circ F\pi_1 \circ \gamma$, and since Fi_X is mono we may conclude $\alpha_X \circ \pi_1 = F\pi_1 \circ \gamma$. Similarly we derive $\alpha_Y \circ \pi_2 = F\pi_2 \circ \gamma$. So R is a bisimulation on X and Y. □

Proposition 6. Let (X, α_X) and (Y, α_Y) be F-coalgebras, and $R \subseteq X \times X$ a relation. Then R is a bisimulation on X iff $(i_X \times i_Y)[R]$ is a bisimulation on $X + Y$.

Proof. Similar to that of Proposition 5. □

From the above two propositions, one can deduce the following:

Corollary 3. Let F be a Set endofunctor. Suppose F-bisimulations on single systems (i.e. of type $R \subseteq X \times X$) are closed under composition. Then F-bisimulations on different coalgebras (i.e. of type $R \subseteq X \times Y$) are closed under composition as well.

Proof. The outline of the proof is as follows. Let $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ be bisimulations. Then by applying first Proposition 5 and then Proposition 6, we can turn both into bisimulations on $X + Y + Z$. The composition of these two then is a bisimulation by assumption; and applying Propositions 6 and 5 in the other direction again, we obtain that $R \circ S$ is a bisimulation on X and Z. □

References

Aceto, L., Fokkink, W. and Verhoef, C. (2001). Structural operational semantics. In: *Handbook of Process Algebra*, Elsevier Science, 197–292.

Aczel, P. and Mendler, N. (1989). A final coalgebra theorem. In: *Category Theory and Computer Science*, *LNCS*, volume 389, Springer, 357–365.

Adámek, J., Koubek, V. and Velebil, J. (2000). A duality between infinitary varieties and algebraic theories. *Commentationes Mathematicae Universitatis Carolinae* **41** (3) 529–542.

Bartels, F. (2004). *On Generalised Coinduction and Probabilistic Specification Formats*. Ph.D. thesis, CWI, Amsterdam.

Berstel, J. and Reutenauer, C. (1988). *Rational Series and Their Languages*, Springer.

Bloom, B., Istrail, S. and Meyer, A. (1995). Bisimulation can't be traced. *Journal of ACM* **42** (1) 232–268.

Bonchi, F., Bonsangue, M., Boreale, M., Rutten, J. and Silva, A. (2012). A coalgebraic perspective on linear weighted automata. *Information and Computation* **211** 77–105.

Bonchi, F., Petrisan, D., Pous, D. and Rot, J. (2014). Coinduction up-to in a fibrational setting. In: Henzinger, T.A. and Miller, D. (eds.) *CSL-LICS 2014*, ACM, 20.

Bonchi, F. and Pous, D. (2013). Checking NFA equivalence with bisimulations up to congruence. In: Giacobazzi, R. and Cousot, R. (eds.) *POPL*, ACM, 457–468.

Bonsangue, M.M., Hansen, H.H., Kurz, A. and Rot, J. (2013). Presenting distributive laws. In: Heckel, R. and Milius, S. (eds.) CALCO. *Lecture Notes in Computer Science* **8089**, Springer, Berlin, 95–109.

Buchholz, P. and Kemper, P. (2001). Quantifying the dynamic behavior of process algebras. In: de Alfaro, L. and Gilmore, S. (eds.) PAPM-PROBMIV. *Lecture Notes in Computer Science*, Springer, Berlin, 184–199.

Conway, J. (1971). *Regular Algebra and Finite Machines*, Chapman and Hall.

Endrullis, J., Hendriks, D. and Bodin, M. (2013). Circular coinduction in Coq using bisimulation-up-to techniques. In: Blazy, S., Paulin-Mohring, C. and Pichardie, D. (eds.) ITP 2013. *LNCS* **7998**, Springer, Berlin, 354–369.

Gorín, D. and Schröder, L. (2013). Simulations and bisimulations for coalgebraic modal logics. In: Heckel, R. and Milius, S. (eds.) Proceedings of the 5th International Conference on Algebra and Coalgebra in Computer Science (CALCO 2013). *Lecture Notes in Computer Science* **8089**, Springer, Berlin, 253–266.

Gumm, H. (1999). Elements of the general theory of coalgebras. In: *LUATCS 99*, Rand Afrikaans University, South Africa.

Gumm, H. and Schröder, T. (2000). Coalgebraic structure from weak limit preserving functors. *Electronic Notes in Theoretical Computer Science* **33** 111–131.

Gumm, H.P. and Schröder, T. (2001). Monoid-labeled transition systems. *Electronic Notes in Theoretical Computer Science* **44** (1) 185–204.

Hermida, C. and Jacobs, B. (1998). Structural induction and coinduction in a fibrational setting. *Information and Computation* **145** (2) 107–152.

Hopcroft, J. and Karp, R. (1971). A linear algorithm for testing equivalence of finite automata. *Technical Report* 114, Cornell Univ.

Klin, B. (2009). Structural operational semantics for weighted transition systems. In: Palsberg, J. (eds.) Semantics and Algebraic Specification. *Lecture Notes in Computer Science* **5700**, Springer, Berlin, 121–139.

Klin, B. (2011). Bialgebras for structural operational semantics: An introduction. *TCS* **412** (38) 5043–5069.

Lenisa, M. (1999). From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. *ENTCS* **19** 2–22.

Lenisa, M., Power, J. and Watanabe, H. (2000). Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. *ENTCS* **33** 230–260.

Levy, P.B. (2011). Similarity quotients as final coalgebras. In: Hofmann, M. (eds.) FOSSACS 2011. *Lecture Notes in Computer Science* **6604**, Springer, Berlin, 27–41.

Luo, L. (2006). An effective coalgebraic bisimulation proof method. *Electronic Notes in Theoretical Computer Science* **164** (1) 105–119.

Marti, J. and Venema, Y. (2012). Lax extensions of coalgebra functors. In: Pattinson, D. and Schröder, L. (eds.) 11th International Workshop on Coalgebraic Methods in Computer Science (CMCS 2012). *Lecture Notes in Computer Science*, Springer, Berlin, **7399** 150–169.

Milner, R. (1980). *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, volume 92, Springer.

Milner, R. (1983). Calculi for synchrony and asynchrony. *TCS* **25** (3) 267–310.

Park, D. (1981). Concurrency and automata on infinite sequences. In: Deussen, P. (eds.) Theoretical Computer Science. *Lecture Notes in Computer Science* **104**, Springer, Berlin, 167–183.

Pous, D. (2007). Complete lattices and up-to techniques. In: Proceedings of the APLAS. *Springer Lecture Notes in Computer Science* **4807** 351–366.

Pous, D. and Sangiorgi, D. (2012). Enhancements of the bisimulation proof method. In: *Advanced Topics in Bisimulation and Coinduction*, Cambridge University Press, 233–289.

Rot, J., Bonsangue, M. and Rutten, J. (2013a). Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F., Italiano, G., Nawrocki, J. and Sack, H. (eds.) SOFSEM. *Springer Lecture Notes in Computer Science* **7741** 369–381.

Rot, J., Bonsangue, M. and Rutten, J. (2013b). Coinductive proof techniques for language equivalence. In: Dediu, A.-H., Martín-Vide, C. and Truthe, B. (eds.) LATA. *Lecture Notes in Computer Science* **7810**, Springer, Berlin, 480–492.

Rutten, J. (1998). Relators and metric bisimulations. *ENTCS* **11** 252–258.

Rutten, J. (2000). Universal coalgebra: A theory of systems. *TCS* **249** (1) 3–80.

Rutten, J. (2003). Behavioural differential equations: A coinductive calculus of streams, automata, and power series. *TCS* **308** (1–3) 1–53.

Salomaa, A. and Soittola, M. (1978). *Automata-Theoretic Aspects of Formal Power Series*, Texts and Monographs on Computer Science, Springer.

Sangiorgi, D. (1998). On the bisimulation proof method. *Mathematical Structures in Computer Science* **8** (5) 447–479.

Sangiorgi, D. (2012). *An introduction to Bisimulation and Coinduction*, Cambridge University Press.

Staton, S. (2011). Relating coalgebraic notions of bisimulation. *LMCS* **7** (1).

Trnková, V. (1980). General theory of relational automata. *Fundamenta Informaticae* **3** (2) 189–234.

Turi, D. and Plotkin, G. (1997). Towards a mathematical operational semantics. In: *LICS*, IEEE Computer Society, 280–291.

Winter, J., Bonsangue, M.M. and Rutten, J.J.M.M. (2011). Context-free languages, coalgebraically. In: Corradini, A., Klin, B. and Cîrstea, C. (eds.) CALCO. *Lecture Notes in Computer Science* **6859**, Springer, Berlin, 359–376.

Zhou, X., Li, Y., Li, W., Qiao, H. and Shu, Z. (2010). Bisimulation proof methods in a path-based specification language for polynomial coalgebras. In: APLAS. *Lecture Notes in Computer Science* **6461**, Springer, Berlin, 239–254.