Coalgebraic Bisimulation-Up-To

Jurriaan Rot[1,2,⋆], Marcello Bonsangue[1,2], and Jan Rutten[2,3]

[1] LIACS – Leiden University
{jrot,marcello}@liacs.nl
[2] Centrum Wiskunde en Informatica
[3] Radboud University Nijmegen
janr@cwi.nl

Abstract. Bisimulation-up-to enhances the bisimulation proof method for process equivalence. We present its generalization from labelled transition systems to arbitrary coalgebras, and show that for a large class of systems, enhancements such as bisimulation up to bisimilarity, up to equivalence and up to context are sound proof techniques. This allows for simplified bisimulation proofs for many different types of state-based systems.

1 Introduction

Bisimilarity is a fundamental notion of equivalence between labelled transition systems. Two processes are bisimilar if they are related by a bisimulation, which is a relation between states such that related pairs match each others transitions and their successors (also called derivatives) are again related. *Bisimulation-up-to* refers to a family of techniques for proving bisimilarity based on smaller relations than usual, in many cases reducing the amount of work [13,9,4]. For example, in a *bisimulation up to bisimilarity* the derivatives do not need to be related directly but may be *bisimilar* to states which are [8]; this is a valid proof method for bisimulation on labelled transition systems. *Bisimulation up to context* [13] is another such technique, in which one can use the algebraic structure (syntax) of processes to relate derivatives.

The theory of *coalgebras* provides a mathematical framework for the uniform study of many types of state-based systems, including labelled transition systems but also (non)-deterministic automata, stream systems, various types of probabilistic and weighted automata, etc. The type of a coalgebra is expressed by an endofunctor F. One of the main elements of the theory is the coalgebraic notion of bisimulation, which generalizes classical bisimulation for labelled transition systems to arbitrary coalgebras. Another notion of equivalence of coalgebras is *behavioural equivalence*. Intuitively, two states are behaviourally equivalent if they have the same observable behaviour, where the observations depend on the type functor F. For many types of systems, including labelled transition

⋆ The research of this author has been funded by the Netherlands Organisation for Scientific Research (NWO), CoRE project, dossier No.: 612.063.920.

P. van Emde Boas et al. (Eds.): SOFSEM 2013, LNCS 7741, pp. 369–381, 2013.

systems, these notions coincide, but for some types, such as certain types of weighted automata, bisimulation is stronger than behavioural equivalence (see, e.g., [3]).

In this paper we introduce a generalization of bisimulation-up-to from labelled transition systems to the theory of coalgebras. In this setting we define bisimulation up to bisimilarity, up-to-union, up-to-context, up-to-equivalence and combinations thereof. As it turns out, the general notion of coalgebraic bisimulation-up-to which we introduce is somewhat problematic: we show that bisimulation up to bisimilarity is *not* sound in general, i.e., that it can not be used as a valid proof principle. So we introduce *behavioural equivalence-up-to*, for which all of the aforementioned instances work very well. Then by the correspondence between behavioural equivalence and bisimulation which holds for many types of systems (for coalgebras for weak pullback preserving functors, to be precise) we obtain the soundness of bisimulation-up-to for such systems.

Related work. Sangiorgi [13] introduced the first systematic treatment of more general up-to techniques for labelled transition systems, and discussed the important notion of bisimulation up to context. A good reference for the current state of the art in this line of research is [9]. In [4] bisimulation up to context is applied to obtain a very efficient algorithm for checking equivalence of non-deterministic finite automata. Lenisa [7] developed bisimulation-up-to in the context of a set-theoretic notion of coinduction. Moreover in *loc. cit.* a framework for up-to techniques for coalgebraic bisimulation is studied, but as mentioned in the paper itself already, the important notion of bisimulation up to bisimilarity is problematic in this setting. Bisimulation up to context is studied at a general coalgebraic level in [7].

The up-to-context technique for coalgebraic bisimulation was later derived as a special case of so-called λ-coinduction: Bartels [2] showed that this technique can be applied in the context of operators defined by certain natural transformations, corresponding for example to the well-known GSOS format in the case of labelled transition systems. A direct corollary of this is the soundness of bisimulation up to context for CCS. However, [2, pages 126, 129] mentions already that it would be ideal to combine the up-to-context technique with other up-to techniques. Indeed, combining up-to-context with up-to-bisimilarity or up-to-equivalence yields powerful proof techniques (see, e.g., [9] and this paper for examples).

The recent paper [14] 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 ourselves on the standard notion of bisimulation, and only need to restrict to weak pullback preserving functors, to obtain our soundness results.

Outline. The following section contains the necessary preliminaries. Then in Section 3 we introduce bisimulation-up-to for coalgebras, together with important instances and examples, and we discuss their soundness in Section 4. In Section 5 we recall behavioural equivalence and its relation with bisimulation,

and in Section 6 we present the main soundness results of bisimulation-up-to via behavioural equivalence-up-to. We conclude in Section 7.

2 Preliminaries

By Set we denote the category of sets and total functions. For a relation $R \subseteq X \times X$ we denote by $\overline{R}$ the smallest equivalence relation containing R, i.e., its reflexive, symmetric and transitive closure, or *equivalence closure*. If R is an equivalence relation then we denote by $q_R : X \to X/R$ the quotient map, which sends each element to its equivalence class. For any relation $R \subseteq X \times Y$ we denote by $\pi_1^R : R \to X$ and $\pi_2^R : R \to Y$ the respective projection maps, and we omit R if it is clear from the context. Given two functions $f : X \to Y$ and $g : X \to Z$, the *pairing* of f and g is the unique function $\langle f, g \rangle : X \to Y \times Z$ with the property that $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$. The *kernel* of a function $f : X \to Y$ is defined as $\ker(f) = \{(x, y) \in X \times X \mid f(x) = f(y)\}$.

A *coalgebra* for a functor $F : \mathsf{Set} \to \mathsf{Set}$ is a pair (X, α) consisting of a set X and a function $\alpha : X \to FX$. We call X the *carrier* or the set of *states*, and α the *transition structure* or *dynamics*. A function $X \to Y$ between the respective carriers of two coalgebras (X, α) and (Y, β) is a *homomorphism* if $Ff \circ \alpha = \beta \circ f$. For a coalgebra $\alpha : X \to FX$, a relation $R \subseteq X \times X$ is an $(F\text{-})$*bisimulation* if R itself can be equipped with a transition structure $\gamma : R \to FR$ such that the following diagram commutes:

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \alpha} \\ FX & \xleftarrow[F\pi_1]{} & FR & \xrightarrow[F\pi_1]{} & FX \end{array}$$

The largest bisimulation on α exists [11] and is denoted by $\sim_\alpha$, or simply by $\sim$ if α is clear from the context. Two states $x, y \in X$ of a coalgebra are called *bisimilar* if $x \sim y$.

Example 1. We recall several types of coalgebras and their corresponding notions of bisimulation. In (3) and (4) below we introduce operations on coalgebras; these will become relevant in the examples in Section 3.

1. Finitely branching *labelled transition systems* (lts's) over a set of labels A are coalgebras for the functor $FX = \mathcal{P}_f(A \times X)$. For an lts $\alpha : X \to \mathcal{P}_f(A \times X)$ we write $x \xrightarrow{a} x'$ iff $(a, x') \in \alpha(x)$. So intuitively, for a state $x \in X$, $\alpha(x)$ contains all the outgoing labelled transitions from x. Bisimulation instantiates to the classical definition. 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.

2. One of the simplest interesting types of coalgebras is given by the functor $FX = X + \mathbb{1}$, where $\mathbb{1}$ is the singleton $\{*\}$. We call such coalgebras *deterministic systems with termination*. An F-coalgebra $\alpha : X \to X + \mathbb{1}$ can, for a given state x, either terminate ($\alpha(x) = *$) or make a transition to another state $x' \in X$ ($\alpha(x) = x'$). If a state x terminates we write $x\downarrow$, otherwise we write x' for $\alpha(x)$, and call x' the *derivative* of x. In this case a *bisimulation* is a relation $R \subseteq X \times X$ such that for all $(x, y) \in R$: either $x\downarrow$ and $y\downarrow$, or $(x', y') \in R$.
3. Coalgebras for the functor $FX = \mathbb{R} \times X$, where $\mathbb{R}$ is the set of real numbers, are called *stream systems* (over the reals). For a stream system $\langle o, t\rangle : X \to \mathbb{R} \times X$ and a state $x \in X$, if o and t are clear from the context we denote $o(x)$ by x_0 and $t(x)$ by x'. A relation $R \subseteq X \times X$ is a bisimulation if for each $(x, y) \in R$: $x_0 = y_0$ and $(x', y') \in R$. A special stream system is formed by taking as carrier the set $\mathbb{R}^\omega = \{\sigma \mid \sigma : \mathbb{N} \to \mathbb{R}\}$ of all streams (infinite sequences) of elements of $\mathbb{R}$, and defining the transition structure $\langle o, t\rangle : \mathbb{R}^\omega \to \mathbb{R} \times \mathbb{R}^\omega$ as $o(\sigma) = \sigma(0)$ and $t(\sigma)(n) = \sigma(n + 1)$). This F-coalgebra is in fact the *final* one, that is, every stream system has a unique homomorphism into it [11]. We may define operations on streams by means of *behavioural differential equations* [12], in which an operation is defined by specifying its initial value and its derivative. Instead of recalling the general definition we only consider the operations of *addition* ($+$), *shuffle product* ($\otimes$) and *shuffle inverse* ($^{-1}$):

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

The inverse operation is only defined on streams σ for which $\sigma_0 \neq 0$. With every real number r we associate a stream $[r] = (r, 0, 0, 0, \ldots)$ (we will abuse notation and denote $[r]$ by r), and we abbreviate $(-1) \otimes \sigma$ 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 can turn $T(\mathbb{R}^\omega)$ into a coalgebra $\mathcal{S} = (T(\mathbb{R}^\omega), \beta)$ by defining the transition structure by induction according to the final coalgebra (for the base case) and the above specification (for the other terms).
4. Coalgebras for the functor $FX = 2 \times X^A$ correspond to deterministic automata with transitions in A. For a coalgebra $\langle o, f\rangle : X \to 2 \times X^A$ and a state $x \in X$, we have $o(x) = 1$ iff x is a *final* or *accepting* state. The function $f(x)$ assigns, to each alphabet letter $a \in A$, the next state or *a-derivative*, denoted x_a. A relation $R \subseteq X \times X$ is a bisimulation if for each $(x, y) \in R$: $o(x) = o(y)$ and for each $a \in A$: $(x_a, y_a) \in R$. The set of formal languages $\mathcal{P}(A^*)$ can be given an F-transition structure $\langle o, f\rangle$ as follows: $o(l) = 1$ iff l contains the empty word, and $f(l)(a) = \{\, w \mid aw \in L \,\}$. This is the final F-coalgebra.

Using a suitable format of behavioural differential equations we can define the operations of regular expressions as follows [10]:

Differential equation	Initial value	Name
$0_a = 0$	$o(0) = 0$	zero
$1_a = 0$	$o(1) = 1$	one
$b_a = \begin{cases} 1 & \text{if } b = a \\ 0 & \text{otherwise} \end{cases}$	$o(b) = 0$	
$(x + y)_a = x_a + y_a$	$o(x + y) = \max(o(x), o(y))$	union
$(x \cdot y)_a = \begin{cases} x_a \cdot y & \text{if } o(x) = 0 \\ x_a \cdot y + y_a & \text{otherwise} \end{cases}$	$o(x \cdot y) = \min(o(x), o(y))$	composition
$(x^*)_a = x_a \cdot x^*$	$o(x^*) = 1$	Kleene star

In a similar way as we have done for the operations on streams, we can construct the set of regular expressions over languages $T(\mathcal{P}(A^*))$ and turn it into a coalgebra $\mathcal{R} = (T(\mathcal{P}(A^*)), \beta)$, defining the transition structure β by induction according to the final coalgebra and the above specification of the operators.

Algebras and distributive laws. In this paragraph we shortly recall some concepts needed for our discussion of bisimulation up to context. These are quite technical, and we do not have the space to cover them in detail; however, a large part of the remainder of this paper can be understood without these preliminaries. An (Eilenberg-Moore) *algebra* for a monad T on Set is a tuple (X, α) consisting of a set X and a map $TX \to X$, satisfying some additional laws (see, e.g., [6]). For an example, recall that in the above Example 1(3), we formed the set of terms $T(\mathbb{R}^\omega)$ over streams of real numbers. In general, given operation and constant symbols (with associated arities), the functor T which constructs the corresponding set of terms over sets of variables X, is (part of) a monad, which we call the *term monad* (for this signature). The construction of regular expressions in Example 1(4) is another such example. Moreover for a set X, a monad gives us an algebra $\alpha : TTX \to TX$. In the case of a term monad this α is called a *term algebra*; it turns a term over terms into a single term.

An (F, T)-*bialgebra* is a triple (X, α, β) consisting of a T-algebra α and an F-coalgebra β. We can extend the coalgebra $\mathcal{S} = \langle T(\mathbb{R}^\omega), \beta \rangle$ from Example 1(3) above to a bialgebra $\langle T(\mathbb{R}^\omega), \alpha, \beta \rangle$ where α is the term algebra. Similarly $\mathcal{R}$ forms a bialgebra together with its associated term algebra. Given a T-algebra α, the set of *contexts* over a relation $R \subseteq X \times X$ is defined as $C_\alpha(R) \subseteq X \times X = \langle \alpha \circ T\pi_1, \alpha \circ T\pi_2 \rangle(TR)$. If T is a term monad, then $C_\alpha(R)$ can be characterized concretely as follows: for two terms $t_1, t_2 \in TX$ we have $(t_1, t_2) \in C_\alpha(R)$ iff we can obtain t_2 by substituting the variables of t_1 for variables related by R.

The interplay between syntax and semantics can be captured by the categorical notion of a *distributive law* of a monad T over a functor $F \times Id$, which is a natural transformation $\lambda : T(F \times Id) \Rightarrow FT \times T$ satisfying some laws (see, e.g., [2,6] for a definition). Intuitively T models syntax and F models behaviour,

and distributive laws of this type can in fact be seen as abstract operational semantics. Indeed, for particular functors F they correspond to concrete formats such as the well-known GSOS format for processes [2,6]. For instance, the definition of the operations in Example 1 above give rise to distributive laws. For a distributive law λ, a bialgebra is called a λ-*bialgebra* if α and β decompose via λ; for lack of space we again refer to [2,6] for a precise definition. For now we note that the coalgebras $\mathcal{S}$ and $\mathcal{R}$ of Example 1, together with their respective term algebras, form λ-bialgebras.

3 Bisimulation-Up-To

We introduce the following definition, which generalizes the notions of *progression* and *bisimulation-up-to* from labelled transition systems [9] to coalgebras.

Definition 1. *Let $\alpha : X \to FX$ be an F-coalgebra. Given relations $R, S \subseteq X \times X$, we say that R* progresses to *S if there exists a transition structure $\gamma : R \to FS$ making the following diagram commute:*

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1^R} & R & \xrightarrow{\pi_2^R} & X \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \alpha} \\ FX & \xleftarrow[F\pi_1^S]{} & FS & \xrightarrow[F\pi_2^S]{} & FX \end{array}$$

If R progresses to $f(R)$ for a function $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ then we say R is a bisimulation up to *f.*

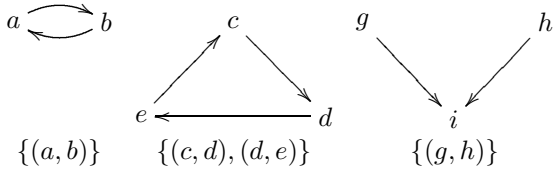
Notice that we have not put any restrictions on the function f in the definition of bisimulation-up-to, and so in general a bisimulation up to f will *not* be a bisimulation, neither will it contain only bisimilar pairs. We continue to introduce several concrete instances of bisimulation-up-to, inspired by the corresponding techniques for labelled transition systems [9]. For now, these instances of bisimulations-up-to do not yet guarantee any pairs to be bisimilar; a discussion of when they do follows afterwards, when we define and study their *soundness*.

Up-to-identity. For a trivial instance of bisimulation-up-to, consider the identity function *id* on relations. A bisimulation up to *id* is simply a bisimulation.

Up-to-equivalence. Consider the function $f(R) = \overline{R}$ which takes a relation R to its equivalence closure. We call a bisimulation up to f a bisimulation *up to equivalence*.

Example 2. Let $\alpha : X \to X + \mathbb{1}$ be a deterministic system with termination, and $R \subseteq X \times X$ a relation. Spelling out the definition, we find that R is a bisimulation up to equivalence if for all $(x, y) \in R$: either $x \downarrow$ and $y \downarrow$, or $x' \; \overline{R} \; y'$. Thus instead of requiring the respective derivatives of each of the pairs

to be related in R again, for a bisimulation up to equivalence, they need only to be related by the reflexive, symmetric and transitive closure of R. Consider the following three deterministic systems and relations (where $x \to y$ iff $x' = y$):



All three of these relations are bisimulations up to equivalence, whereas none of them are actual bisimulations. Consider for instance the relation $\{(a, b)\}$; when we compute the respective derivatives of a and b we obtain $a' = b$ and $b' = a$, but $(b, a) \notin \{(a, b)\}$. However, the pair (b, a) *is* in the least equivalence relation containing $\{(a, b)\}$. □

Up-to-union. For $S \subseteq X \times X$ a relation, consider the function $f(R) = R \cup S$. We call a bisimulation up to f a bisimulation *up to union with* S. In order for a relation R to be a bisimulation up to union with S, the derivatives of R may be related either by R again or by S.

Example 3. For a deterministic automaton $\langle o, f \rangle : X \to 2 \times X^A$, a relation $R \subseteq X \times X$ is a bisimulation up to union with S if for all $(x, y) \in R$: $o(x) = o(y)$ and for any alphabet letter $a \in A$: either $x_a R y_a$ or $x_a S y_a$.

Consider for instance the relation $R = \{(1^*, 1)\}$ on the automaton $\mathcal{R}$ of regular expressions over languages, introduced in Example 1(4). We first note that $o(1^*) = 1 = o(1)$; next let $a \in A$ be an alphabet letter. Then $(1^*)_a = 1_a \cdot 1^* = 0 \cdot 1^*$ and $1_a = 0$. Note that $(0 \cdot 1^*, 0) \notin R$, so R is not a bisimulation. However suppose S is a relation containing some of the basic equivalences between regular expressions, such as $(0 \cdot r, 0) \in S$ for any regular expression r. Then our relation R is a bisimulation up to union with S, since the derivatives of the single pair in R are related by S. Notice that R is also a bisimulation up to union with $\sim$, given that $0 \cdot a$ is bisimilar to 0. □

Up-to-context. The notion of bisimulation up to *context* applies to coalgebras where the state space consists of the elements of an algebra. So let (X, α, β) be an (F, T)-bialgebra (see the last part of Section 2). If a relation $R \subseteq X \times X$ progresses to the set of contexts $C_\alpha(R)$, then we call R a *bisimulation up to context.*

If T is a term monad, such as in the case of the introduced operations on streams or regular expressions, then in practice this technique only becomes interesting when combined with other techniques such as up-to-equivalence or up-to-bisimilarity. A notable exception is when one considers bisimulation up to context on a *final* coalgebra, which is the approach taken in the examples in [2] (e.g., page 126). In that case, combination with up-to-bisimilarity comes for free, since bisimilarity implies equality on final coalgebras [11].

Up-to-union-and-equivalence. Let $f(R) = \overline{R \cup S}$ for a fixed relation S. If R progresses to $f(R)$ then we call R a *bisimulation up to S-union and equivalence*. By taking the equivalence closure of $R \cup S$, derivatives may be related by arbitrary compositions of R and S. If we consider the special case $S = \sim$ we see that this is in fact a generalization of *bisimulation up to bisimilarity*, meaning that the derivatives may be *bisimilar* to elements which *are* related by R. More formally, a bisimulation up to bisimilarity is based on the function $g(R) = \sim \circ R \circ \sim$. Note that for any relation R, we have $g(R) \subseteq f(R)$. Indeed for a bisimulation up to $\sim$-union and equivalence we take arbitrary compositions of $\sim$ and R, including the specific one $\sim \circ R \circ \sim$.

Example 4. For a deterministic automaton $\langle o, f \rangle : X \to 2 \times X^A$, a relation $R \subseteq X \times X$ is a bisimulation up to S-union and equivalence if for all $(x, y) \in R$: $o(x) = o(y)$ and for any alphabet letter $a \in A$: $(x, y) \in \overline{R \cup S}$. Recall the automaton $\mathcal{R}$ of regular expressions from Example 1(4). Consider the implication $l \sim al + b \Rightarrow l \sim a^*b$ for a language $l \in \mathcal{P}(A^*)$ over alphabet symbols $A = \{a, b\}$, intuitively expressing that a^*b is the unique solution of the "equation" $l \sim al + b$. Let $R = \{(l, a^*b) \mid l \sim al + b\}$, and let l be a language such that $l \sim al + b$. First we check that the outputs are equal: $o(l) = o(al + b) = 0 = o(a^*b)$. Next we compute the a- and b-derivatives of l: $l_a \sim (al + b)_a \sim l$ and $l_b \sim (al + b)_b \sim 1$. Now $l_a \sim lR(a^*b)_a$ and $l_b \sim 1 \sim (a^*b)_b$. Thus R is a bisimulation up to $\sim$-union and equivalence. It is not a bisimulation, since $(l_b, (a^*b)_b) \notin R$. □

Up-to-union-context-and-equivalence. If a relation $R \subseteq X \times X$ on the carrier of an (F, T)-bialgebra (X, α, β) progresses to $\overline{C_\alpha(R \cup S)}$, then R 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 by equational reasoning, using $C_\alpha(R \cup S)$ in *multiple* steps.

Consider again the bialgebra $\langle T(\mathbb{R}^\omega), \alpha, \beta \rangle$ consisting of the stream system $\mathcal{S}$ of Example 1(3) and the term algebra α. A bisimulation up to S-union, context and equivalence is a relation $R \subseteq T(\mathbb{R}^\omega) \times T(\mathbb{R}^\omega)$ such that for all $(t_1, t_2) \in R$: $o(t_1) = o(t_2)$ and $(t_1', t_2') \in \overline{C_\alpha(R \cup S)}$. The following is an example on $\mathcal{S}$.

Example 5. Suppose we are given that $\otimes$ is associative and commutative (so $\sigma \otimes \tau \sim \tau \otimes \sigma$, etc.) and that $\sigma + (-\sigma) \sim 0$ (notice that these assumptions actually hold in general [12]). Let $R = \{(\sigma \otimes \sigma^{-1}, 1) \mid \sigma \in T(\mathbb{R}^\omega), \sigma_0 \neq 0\}$. We can now establish that R is a bisimulation up to $\sim$-union, context and equivalence on the coalgebra $\mathcal{S}$. First consider the initial values: $(\sigma \otimes \sigma^{-1})_0 = \sigma_0 \times \sigma_0^{-1} = 0 = \sigma_0 \times (\sigma_0)^{-1} = 1 = 1_0$. Next we relate the derivatives by $\overline{C_\alpha(R \cup \sim)}$:

$$
\begin{aligned}
(\sigma \otimes \sigma^{-1})' = \sigma' \otimes \sigma^{-1} + \sigma \otimes (\sigma^{-1})' &= \sigma' \otimes \sigma^{-1} + \sigma \otimes (-\sigma' \otimes (\sigma^{-1} \otimes \sigma^{-1})) \\
C_\alpha(\sim)^*\ (\sigma' \otimes \sigma^{-1}) &+ (-(\sigma' \otimes \sigma^{-1}) \otimes (\sigma \otimes \sigma^{-1})) \\
C_\alpha(R \cup \sim)\ (\sigma' \otimes \sigma^{-1}) &+ (-(\sigma' \otimes \sigma^{-1}) \otimes 1)\ C_\alpha(\sim)^*\ 0 = 1'
\end{aligned}
$$

where $C_\alpha(\sim)^*$ denotes the transitive closure of $C_\alpha(\sim)$; in the above we apply multiple substitutions of terms for bisimilar terms. Notice that R is not a

bisimulation; here, establishing a bisimulation-up-to is much easier than finding a bisimulation which contains R. □

For the special case $S = \emptyset$, bisimulation up to f is called *bisimulation up to context and equivalence*, or *bisimulation up to congruence*. Finally notice that bisimulation up to $\sim$-union, context and equivalence generalizes the notion of *bisimulation up to context and bisimilarity* (see, e.g., [9]), in a similar way as up-to-union-and-equivalence generalizes up-to-bisimilarity.

4 Soundness

In the above examples, intuitively, establishing that a relation R is a bisimulation-up-to should imply that R indeed contains only bisimilar pairs, i.e., $R \subseteq \sim$. However this depends on the instance of bisimulation-up-to under consideration. For example, one function $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ which, in general, does not have this property is $f(R) = X \times X$, where *any* derivatives are related. But more subtly, for up-to-context, this depends on the bialgebraic structure.

Definition 2. *Let $\alpha : X \to FX$ be a coalgebra, and $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ a function. If $R \subseteq \sim$ for any relation $R \subseteq X \times X$ which is a bisimulation up to f, then we say bisimulation up to f is* sound for α. *If this holds for any coalgebra, then we say bisimulation up to f is* sound.

Bisimulation up to identity is sound, of course. If S is a relation which contains only bisimilar pairs, then bisimulation up to union with S is sound.

Proposition 1. *For any coalgebra $\alpha : X \to FX$, and for any relation $S \subseteq X \times X$: if $S \subseteq \sim$ then bisimulation up to union with S is sound.*

Proof. If R is a bisimulation up to union with S then it is also a bisimulation up to union with $\sim$. For any such R, $R \cup \sim$ is a bisimulation. □

As a consequence of the above proposition, Example 3 contains a full proof that 1^* and 1 are bisimilar, given the knowledge that $0 \cdot 1^*$ is bisimilar to 0. From [2] we obtain a sufficient condition for the soundness of bisimulation up to context:

Proposition 2 ([2], Corollary 4.3.8). *If (X, α, β) is a λ-bialgebra for a distributive law λ of a monad T over a functor $F \times Id$, then bisimulation up to context is sound for (X, β).*

Bisimulation up to bisimilarity is *not* sound in general. This is illustrated by the following example, which is based on an example from [1] (introduced there to show that bisimulation and behavioural equivalence (see Section 5) do not coincide, in general).

Example 6. Define the functor $F : \mathsf{Set} \to \mathsf{Set}$ as $FX = \{(x_1, x_2, x_3) \in X^3 \mid |\{x_1, x_2, x_3\}| \leq 2\}$ and $F(f)(x_1, x_2, x_3) = (f(x_1), f(x_2), f(x_3))$. Consider the F-coalgebra with states $X = \{0, 1, 2, \tilde{0}, \tilde{1}\}$ and transition structure $\{0 \mapsto (0, 1, 0), 1 \mapsto (0, 0, 1), \tilde{0} \mapsto (0, 0, 0), \tilde{1} \mapsto (1, 1, 1), 2 \mapsto (2, 2, 2)\}$. 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 can not 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 $S = \{(\tilde{0},\tilde{1}),(2,2)\}$. We can define a transition structure $\gamma : S \to FS$ as follows:

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

But $0 \sim 2\ S\ 2 \sim 1$ (and $2 \sim S \sim 2$) 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. □

Bisimulation up to equivalence is not sound either, which can be shown by a similar argument. While the above counterexample is based on a somewhat "unrealistic" functor, in [3, Figure 1] there is an example of a weighted automaton, which is easily turned into another counterexample, showing that bisimulation up to bisimilarity (or up to equivalence) is not sound in the case of weighted automata. Fortunately, for a large class of systems (even including certain types of weighted automata) bisimulation up to equivalence *is* sound, namely for those modeled by functors which preserve weak pullbacks; we develop this result in Section 6.

5 Behavioural Equivalence

In this section we recall a notion of equivalence which is in general weaker than bisimulation, but in many cases (depending on the type of system) the two notions coincide. Suppose $\alpha : X \to FX$ is a coalgebra. Two states s,t are called (F-)*behaviourally equivalent* if there exists a homomorphism $f : X \to Y$ into some other coalgebra, such that $f(s) = f(t)$. This notion is pointwise extended to relations $R \subseteq X \times X$, i.e., R is an (F-)*behavioural equivalence* if $R \subseteq \ker(f)$. In the sequel we will base ourselves on a more concrete characterization of behavioural equivalence, which can be seen to be a slight variation of a characterization from [5] (which considers only equivalence relations in this context, but the generalization to arbitrary relations is easy).

Proposition 3 ([5], Lemma 4.12). *Let $\alpha : X \to FX$ be an F-coalgebra. A relation $R \subseteq X \times X$ is a behavioural equivalence iff the following diagram commutes:*

$$R \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} X \xrightarrow{\alpha} FX \xrightarrow{Fq} F(X/\overline{R})$$

where q is the quotient map of $\overline{R}$.

Example 7. Let $\alpha : X \to X + \mathbb{1}$ be a deterministic system with termination, and $R \subseteq X \times X$ a relation with quotient map q; then $Fq = q + id$. According to the above proposition, R is a behavioural equivalence if for all $(x,y) \in R$, either $x \downarrow$ and $y \downarrow$, or $q(x') = q(y')$. The latter case is equivalent to $x'\ \overline{R}\ y'$. □

In [5], behavioural equivalences which are equivalence relations are called *congruences*, after [1], which introduced this coalgebraic notion of congruence. We chose not to use the term "congruence" to avoid confusion with the well-known concept from universal algebra. Behavioural equivalence coincides with the notion of *pre-congruence* from [1]. We proceed to recall from that paper the precise relation with bisimulation.

Proposition 4 ([1]). *Any F-bisimulation is an F-behavioural equivalence. If F preserves weak pullbacks, then the equivalence closure of any F-behavioural equivalence is an F-bisimulation.*

For a fixed coalgebra α the greatest behavioural equivalence exists [1], and it is denoted by $\approx_\alpha$ or simply $\approx$ if α is clear from the context. From the above proposition we immediately obtain that for weak pullback preserving functors, the greatest behavioural equivalence and the greatest bisimulation coincide.

Many interesting functors used to model systems coalgebraically actually do preserve weak pullbacks, including all functors introduced in Example 1. One example of a relevant functor that does *not* preserve weak pullbacks is that of weighted automata over $\mathbb{R}$. Figure 1 of [3] is an example of a weighted automaton containing states which are behaviourally equivalent, but not bisimilar. In fact, the notion of equivalence of weighted automata chosen in [3] is indeed behavioural equivalence, which coincides with so-called *weighted bisimilarity*.

Example 7 on the one hand illustrates how to obtain a proof method for behavioural equivalence for a given functor F, using Proposition 3. On the other hand, it suggests that in this proof method, we can reason up to equivalence. Indeed, while bisimulation up to equivalence is problematic (not sound) in general, we will see in the next section that for behavioural equivalence this comes quite naturally.

6 Soundness via Behavioural Equivalence-Up-To

Given a function $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$, we define behavioural equivalence up to f as a generalization of the characterization given in Proposition 3. The quotient map of $\overline{f(R)}$ is now used to relate derivatives, instead of the quotient map of $\overline{R}$.

Definition 3. *Let $\alpha : X \to FX$ be an F-coalgebra and $R \subseteq X \times X$. Let $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$. R is a* behavioural equivalence up to f *if the following diagram commutes:*

$$R \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} X \xrightarrow{\alpha} FX \xrightarrow{Fq} F(X/\overline{f(R)})$$

where q is the quotient map of $\overline{f(R)}$.

For example, for a deterministic system $\alpha : X \to X + \mathbb{1}$, a relation $R \subseteq X \times X$ is a behavioural equivalence up to f whenever for all $(x, y) \in R$ either $x \downarrow$ and $y \downarrow$ or $x' \ \overline{f(R)} \ y'$. In order to proceed we define soundness of behavioural equivalence-up-to.

Definition 4. *Let $\alpha : X \to FX$ be a coalgebra, and $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$. If $R \subseteq \approx$ for any relation $R \subseteq X \times X$ which is a behavioural equivalence up to f, then we say behavioural equivalence up to f is* sound for α. *If this holds for any coalgebra, then we say behavioural equivalence up to f is* sound.

Behavioural equivalence up to union, equivalence, context etc. are defined in the same way as for bisimulation-up-to.

Lemma 1. *If (X, α, β) is a λ-bialgebra for a distributive law λ of a finitary*[1] *monad T over a functor $F \times Id$, then behavioural equivalence up to context is sound for (X, β).*

We obtain our main result:

Theorem 1. *Suppose $S \subseteq \approx$. Then (F-)behavioural equivalence up to (S-union and) equivalence is sound. Moreover, if (X, α, β) is a λ-bialgebra for a distributive law λ of a finitary monad T over a functor $F \times Id$, then (F-)behavioural equivalence up to (S-union,) context (and equivalence) is sound for (X, β).*

Proof. Let $f(R) = \overline{C_\alpha(R \cup S)}$. If R is a behavioural equivalence up to f, then it is also simply a bisimulation up to f', where $f'(R) = C_\alpha(R \cup S)$. If $S \subseteq \approx$, then R is a behavioural equivalence up to $f''(R) = C_\alpha(R \cup \approx)$ as well. Then $R \cup \approx$ is behavioural equivalence up to context, so $R \subseteq \approx$ by Lemma 1. □

Behavioural equivalence-up-to is related to bisimulation-up-to as follows:

Lemma 2. *Let $f : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$. Then (1) any bisimulation up to f is also a behavioural equivalence up to f, and (2) if F preserves weak pullbacks, then soundness of behavioural equivalence up to f implies soundness of bisimulation up to f.*

From the above lemma and Theorem 1 we immediately obtain the following:

Corollary 1. *If F preserves weak pullbacks, then Theorem 1 holds for bisimulation-up-to as well.*

Consequently, all the examples of bisimulations-up-to in Section 3 contain actual proofs of bisimilarity, based on smaller relations than any other relations needed to establish bisimilarity without these techniques.

7 Conclusions and Future Work

We generalized bisimulation-up-to to the theory of coalgebras. By extending the theory to behavioural equivalence, we established the soundness of bisimulation up to union and equivalence for systems modeled by functors which preserve weak pullbacks. For any coalgebra with an algebraically structured state space

[1] A monad is finitary if its underlying functor T preserves filtered colimits. If T is a term monad, this means there may be infinitely many operations, but each of them must have finite arity.

which forms a λ-bialgebra for a distributive law λ, we have shown that bisimulation up to union, context and equivalence is sound. Future work includes a generalization to other categories, investigation of instances of bisimulation-up-to for concrete types of systems (e.g., [4]), and the integration with the systematic treatment of enhancements of [9].

References

1. Aczel, P., Mendler, N.: A Final Coalgebra Theorem. In: Dybjer, P., Pitts, A.M., Pitt, D.H., Poigné, A., Rydeheard, D.E. (eds.) Category Theory and Computer Science. LNCS, vol. 389, pp. 357–365. Springer, Heidelberg (1989)
2. Bartels, F.: On generalised coinduction and probabilistic specification formats. PhD thesis, CWI, Amsterdam (2004)
3. Bonchi, F., Bonsangue, M., Boreale, M., Rutten, J., Silva, A.: A coalgebraic perspective on linear weighted automata. Inf. Comput. 211, 77–105 (2012)
4. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: POPL (to appear, 2013)
5. Gumm, H.P.: Elements of the general theory of coalgebras. In: LUATCS 1999 Rand Afrikaans University, South Africa (1999)
6. Klin, B.: Bialgebras for structural operational semantics: An introduction. TCS 412(38), 5043–5069 (2011)
7. Lenisa, M.: From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. ENTCS 19, 2–22 (1999)
8. Milner, R.: Calculi for synchrony and asynchrony. TCS 25, 267–310 (1983)
9. Pous, D., Sangiorgi, D.: Enhancements of the bisimulation proof method. In: Advanced Topics in Bisimulation and Coinduction, pp. 233–289. Cambridge University Press (2012)
10. Rutten, J.: Automata and Coinduction (An Exercise in Coalgebra). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 194–218. Springer, Heidelberg (1998)
11. Rutten, J.: Universal coalgebra: a theory of systems. TCS 249(1), 3–80 (2000)
12. Rutten, J.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. TCS 308(1-3), 1–53 (2003)
13. Sangiorgi, D.: On the bisimulation proof method. Math. Struct. Comp. Sci. 8(5), 447–479 (1998)
14. Zhou, X., Li, Y., Li, W., Qiao, H., Shu, Z.: Bisimulation Proof Methods in a Path-Based Specification Language for Polynomial Coalgebras. In: Ueda, K. (ed.) APLAS 2010. LNCS, vol. 6461, pp. 239–254. Springer, Heidelberg (2010)